COLOR PROCESSING IN DIGITAL CAMERAS

IN SECONDS, A DIGITAL CAMERA PERFORMS FULL-COLOR RENDERING THAT INCLUDES COLOR FILTER ARRAY INTERPOLATION, COLOR CALIBRATION, ANTI-ALIASING, INFRARED REJECTION, AND WHITE-POINT CORRECTION. THIS ARTICLE DESCRIBES THE DESIGN DECISIONS THAT MAKE THIS PROCESSING POSSIBLE.

Jim Adams
Ken Parulski
Kevin Spaulding
Eastman Kodak Company

•••••• Digital cameras are fun! You press the button, and seconds later you see a full-color image on the camera's LCD or your computer's monitor. But in those few seconds, the camera has gone through an elaborate color-processing chain to create that full-color rendering. To develop the process that produced your image, even more has gone on in the heads of the camera design engineers—long before you pressed the button.

Dreaming up a design

If you have never designed a digital camera before, you might naturally start by using three or four image sensors to capture a color image of a scene.[1] Your first block diagram of the camera's optical path might look like Figure 1.

In Figure 1, the taking lens forms an image of the scene on each detector. This design includes a place for a bank of filters that operates on all the optical paths. It also includes a place for three separate banks of spectrally selective filters to enable each sensor to detect a particular color channel—for example, red, green, or blue.

In principle, this arrangement would work and, in fact, some professional digital cameras use this design. The problem is that this is a very expensive approach. Generally, the sensor is the most expensive component of the digital camera; it can easily account for 10% to 25% of the total cost. So, if you wish to make a digital camera priced for the typical consumer, you must settle for a design with as few sensors as possible.

Before Oz: Starting in a black-and-white world

The solution for most digital camera designers is to use a single sensor and to cover that sensor's surface with a mosaic of colored filters. This kind of sensor is called a color filter array, or CFA (see Figure 2). In essence, each pixel has its own spectrally selective filter to peer through. Figure 3 shows two popular CFA patterns used today. Many more CFA designs exist,[2] and still more are being invented.

Figure 4 is a typical image taken with a camera outfitted with the RGB CFA pattern in Figure 3. The first thing to note is that the image appears to be black and white! The next thing to note is the checkerboard pattern throughout the image. This is due to the CFA. Looking at the face in Figure 4, the bright pixels correspond to the red CFA filters—so, they are "red pixels." The dark pixels correspond to the green and blue CFA filters—"green" and "blue" pixels. The red pixels are brighter because skin tones reflect more red light than

0272-1732/98/$10.00 © 1998 IEEE

green or blue light. Each pixel is surrounded by pixels of the other two colors.

Our final observation about this image is that the pixels are generally much smaller than the image's significant details. For example, it takes a large number of pixels to represent the face. Given the last two observations, you should be able to estimate the "missing" color values for each pixel by examining the color values of neighboring pixels. This process is called CFA interpolation.

Follow the yellow brick road

Let's look at the processes of CFA interpolation and color calibration. This will help us address the more fundamental questions of which CFA pattern to use for a given application and what to consider if you decide to design your own CFA pattern.

The first step in most CFA interpolation methods is to produce a fully populated luminance color plane. This luminance color plane (or luminance record) will contain most of the spatial information in your CFA image. If generated properly, it will look like a reasonable gray-scale rendering of your image. With a CMYG CFA pattern, you might create a luminance record using the equation

$$V = \frac{C + M + Y + G}{4}$$

where V is the luminance value. With a RGB CFA pattern, you might define the green channel to be the luminance record and simply work to find the missing green values. You can do this because the spectral response of a green filter looks like the spectral response of the eye's luminance channel.

Because the luminance record contains important spatial information, you need to preserve as much spatial detail as possible during the computation process. One simple method of determining the missing luminance data without preserving spatial detail is simple convolution. For example, consider the CMYG pattern from Figure 3. If you wish to calculate a luminance value for each four-way intersection in the array, you could average the pixel values of the four pixels surrounding each intersection. This could be done very quickly in hardware or firmware and is, in fact, how many video cameras get the job done. The problem is that this averaging blurs sharp edges, producing small, periodic errors in the luminance estimates. Subsequent color processing can amplify these errors and translate them into colored fringes at sharp edges. In a video sequence, however, these fringes might go unnoticed. If the errors are too big

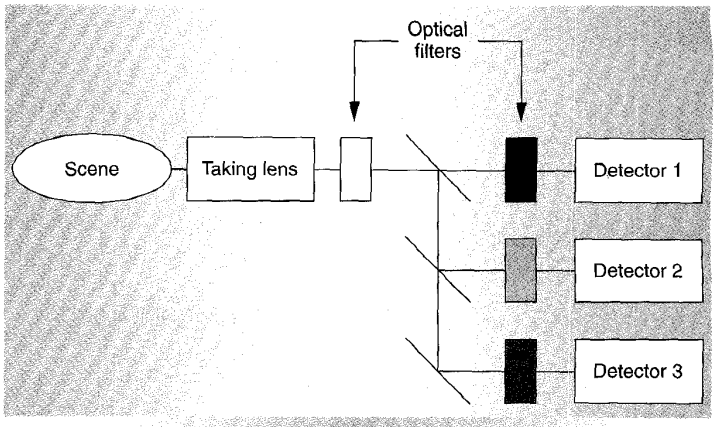

Figure 1. Initial optical-path block diagram for a digital camera.

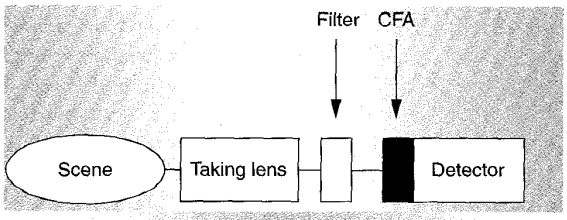

Figure 2. Single-sensor optical-path block diagram for a digital camera.

G	R	G	R	G	R	
B	G	B	G	B	G	
G	R	G	R	G	R	
B	G	B	G	B	G	
G	R	G	R	G	R	
B	G	B	G	B	G	

M	G	M	G	M	G	
C	Y	C	Y	C	Y	
M	G	M	G	M	G	
C	Y	C	Y	C	Y	
M	G	M	G	M	G	
C	Y	C	Y	C	Y	

Figure 3. Two popular color filter arrays. R stands for red, G for green, B for blue, M for magenta, C for cyan, and Y for yellow.

Figure 4. A typical CFA image captured with the RGB CFA in Figure 3.

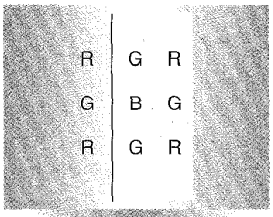

Figure 5. CFA pixel neighborhood with a vertical edge.

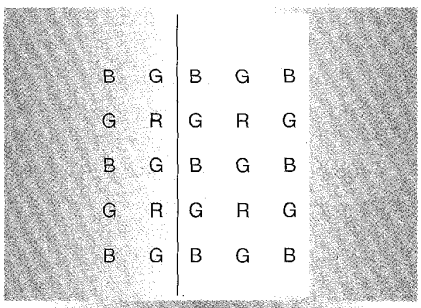

Figure 6. Expanded CFA pixel neighborhood with a vertical edge.

to ignore, you can always sharpen the luminance channel after interpolation. If you do sharpen the luminance channel, you must take care not to sharpen too aggressively, or you will simply exchange luminance prediction errors for sharpening artifacts in your final image.

If you are working with an RGB pattern (see Figure 3), you can do the same type of fast processing by averaging the four green pixels around each red and blue pixel to produce your missing green values. This process will also blur edges, however. So, a better scheme, at least for digital still cameras, is to employ a nonlinear, adaptive strategy.[3] For each pixel, you perform edge detection using neighboring pixels and attempt to average only along edges, rather than across them.

Consider Figure 5. The pixels to the left of the edge have larger pixel values, and the pixels to the right have smaller pixel values. To estimate the missing green value of the blue pixel, you have three obvious choices. You could average all four adjacent green pixels, just the two horizontally adjacent pixels, or just the two vertically adjacent pixels. Clearly, you want to use only pixels to the right of the edge, so you choose the two vertically adjacent pixels. But how do you make that choice algorithmically? You could simply calculate the absolute difference between the horizontally adjacent greens and compare it with the absolute difference between the vertically adjacent greens. The direction with the smaller value would get the nod. If you wished, you could compare both differences against some predetermined threshold value. If both differences are less than the threshold, you would conclude there are no edges in the region and then use a four-way average.

Of course, there are some problems with this technique. What if the edge is diagonally oriented? And, what if you are looking at an image of a picket fence where the slats and the gaps between slats are each one pixel wide? In both cases, the horizontal and vertical differences will be similar, though, in the case of the picket fence, you most definitely want to average along the slats and not across! In the case of the diagonal edge you seem to end up with a poor answer no matter what you do. You cannot even take diagonal differences and make diagonal averages because there are no green pixels along the diagonals!

One solution to some of these problems is to simply use a larger region around the pixel in question and press more pixels into service. Consider expanding the 3×3-pixel neighborhood of Figure 5 into the 5×5-pixel neighborhood of Figure 6.

Now, here's the trick. You assume that the green and blue pixel values are perfectly correlated to within a simple offset for this larger pixel neighborhood, that is,

$$B = G + k$$

where k is the offset for the neighborhood. In practice, this turns out to be a pretty decent assumption for most portions of an image. As a result, instead of only having two horizontal green pixels and two vertical green pixels to work with, you can now use all five horizontal green and blue pixels running through the center of the neighborhood, as well as all five vertical green and blue pixels running through the center of the neighborhood.[4] (If you are calculating the missing green value for a red pixel, simply replace every mention of the color blue with the color red in the preceding discussion.)

Clearly, you can keep enlarging the size of the pixel neighborhood and increasing the complexity of the edge detectors and missing-value predictors. It is up to the digital camera design engineer to decide when the added cost of improving this luminance calculation outweighs the corresponding image quality improvement.

Once you have fully populated the luminance record, you need to create two chrominance records. Usually, creating the chrominance records is much easier than creating the luminance record. Because there is very little spatial information in chrominance records, you can

ignore all edge location considerations and simply work with the smallest pixel neighborhoods that contain values from all color channels. The two chrominance channels are classically know as C_R (the red channel) and C_B (the blue channel). For the CMYG system of Figure 3, you can define these chrominance channels as

$$C_R = \frac{(M+Y)-(G+C)}{2}$$

$$C_B = \frac{(M+C)-(G+Y)}{2}$$

Again, at each four-way intersection of the CFA is all the information you need to calculate both C_R and C_B. If you are calculating luminance values for the center of the pixel rather than at the four-way intersection, you can use a simple two-dimensional linear interpolation to calculate chrominance values that are coincident with the luminance estimates.

For the RGB CFA of Figure 3, you can define C_R as red minus green and C_B as blue minus green. At blue and red pixels you already have a green (luminance) estimate, so calculation of C_B and C_R is trivial. To get the remaining chrominance value at these pixels, you simply average the four diagonally adjacent chrominance values (see Figure 7a).

Depending on where you are in the CFA, the C_Rs and C_Bs might be swapped, but the calculation is the same. In the case of green pixels (see Figure 7b), you average the two horizontal neighbors and the two vertical neighbors to estimate the two missing chrominance values. Again, the locations of the C_Rs and the C_Bs might be swapped, but the calculation is the same.

Once you have a luminance and two chrominance values for each pixel location, you have a full-color rendering of the image! If you want to view this image on a CRT or some other device that works with RGB signals, you can simply convert your V, C_R, and C_B data into RGB.

Seeing the world in living color (Toto, I don't think we're in Kansas anymore...)

Let's assume you convert your new full-color image to RGB and then display it on your CRT. You will probably not be very happy with what you see. First, you need to adjust the image to have the right tone scale for the gamma of the monitor. After that, there will still be problems. The image's colors will be very desaturated, and some colors might have the wrong color name—for example, yellows might look orange. So, clearly, you are not done. The problem is that there is no guarantee that the spectral sensitivity characteristics of the RGB channels generated by the CFA interpolation are an appropriate match for the spectral emission characteristics of your CRT phosphors.[5]

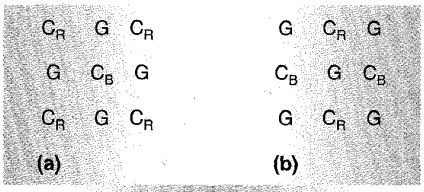

Figure 7. Blue pixel (a) and green pixel (b) chrominance neighborhoods for the RGB CFA of Figure 3.

Fortunately, your friendly neighborhood color scientist comes to the rescue! Working explicitly with the human visual system's internal and highly nonlinear image-processing path is a quick way to getting buried in hopeless complexity—and we still don't know exactly how everything works. Thus, color scientists use a very simple linear model of color perception that works extremely well in most applications. This model describes color in terms of what are called tristimulus values:

$$X = k\sum_{\lambda} \Phi(\lambda)\bar{x}(\lambda)\Delta\lambda$$

$$Y = k\sum_{\lambda} \Phi(\lambda)\bar{y}(\lambda)\Delta\lambda$$

$$Z = k\sum_{\lambda} \Phi(\lambda)\bar{z}(\lambda)\Delta\lambda$$

The terms $\bar{x}$, $\bar{y}$, and $\bar{z}$ are called color-matching functions, and they represent the sensitivities of the human visual system's three color channels. Most color science books tabulate these functions,[6,7] and we will consider them fixed (see Figure 8, next page). The term $\Phi(\lambda)$ is the stimulus function, which takes on one of three forms:

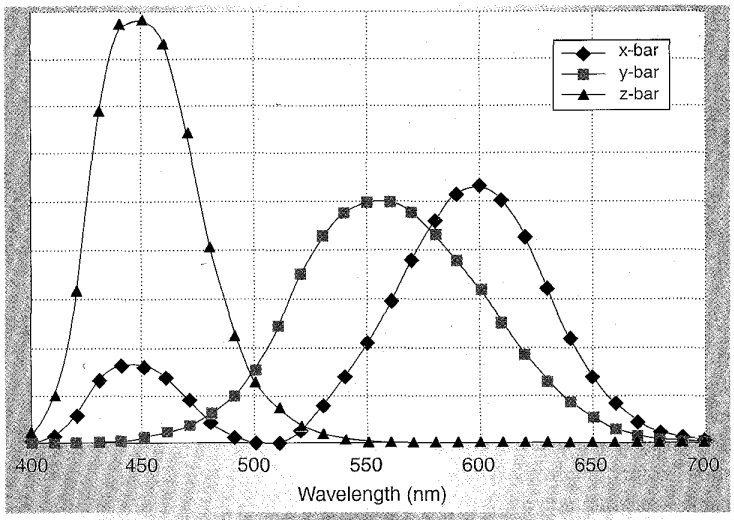

Figure 8. Color-matching functions.

For opaque objects, $\Phi(\lambda) = S(\lambda)R(\lambda)$.
For transparent objects, $\Phi(\lambda) = S(\lambda)T(\lambda)$.
For light sources, $\Phi(\lambda) = S(\lambda)$.

In these equations, $S(\lambda)$ is the spectral power distribution of the light source, $R(\lambda)$ is the spectral reflectance of an object, and $T(\lambda)$ is the spectral transmittance of an object. $\Delta\lambda$ is the wavelength resolution of the color-matching and stimulus functions, and k is a normalizing constant. Finally, X, Y, and Z are the tristimulus values. If two objects have the same sets of tristimulus values, we would say they have identical colors—that is, they appear to be the same color to your visual system.

So, to fix your CFA-interpolated image colors, you perform a color calibration based on the system just described. First, you gather the appropriate data. You begin by capturing an image of a color test chart. This test chart is nothing more than a collection of several small patches, each patch corresponding to a different color. Next, you perform CFA interpolation on your image to create a full-color RGB rendition. After that, you need to make sure that the pixel values scale linearly with scene exposure. (Sometimes systems use nonlinear mappings of pixel code values for data compression and storage considerations. You want to make sure all that is turned off.)

Now, you display a full-color image of the color test chart on the monitor. You next use a device called a colorimeter to directly measure the tristimulus values of the color patches on your video screen. (A colorimeter makes all the appropriate measurements and calculations for you. How convenient!) With the XYZ values of the monitor image in hand, you use the colorimeter to measure the tristimulus values of the original color test chart.

You now have all the data you need to perform your color calibration. Next you need to determine how to modify your interpolated image so that when you display it on the CRT, the colors on the screen will match the colors on the test chart. The most popular mathematical route for doing this is with 3×3 matrix multiplies.

Our goal will be to calculate a 3×3 matrix, $\overline{M}_{INTP \to CAL}$, that will convert the RGB values from your interpolated image (which we'll call R_{INTP}) into color-calibrated RGB values (which we'll call R_{CAL}), ready for use with your CRT.

You begin by temporarily splitting this 3×3 matrix multiplication

$$\begin{pmatrix} R_{CAL} \\ G_{CAL} \\ B_{CAL} \end{pmatrix} = \overline{M}_{INTP \to CAL} \begin{pmatrix} R_{INTP} \\ G_{INTP} \\ B_{INTP} \end{pmatrix}$$

into two matrix multiplications:

$$\begin{pmatrix} R_{CAL} \\ G_{CAL} \\ B_{CAL} \end{pmatrix} = \overline{M}^{-1}_{INTP \to CRT} \begin{pmatrix} X_{TEST} \\ Y_{TEST} \\ Z_{TEST} \end{pmatrix},$$

$$\begin{pmatrix} X_{CRT} \\ Y_{CRT} \\ Z_{CRT} \end{pmatrix} = \overline{M}_{INTP \to CRT} \begin{pmatrix} R_{INTP} \\ G_{INTP} \\ B_{INTP} \end{pmatrix}$$

You first determine $\overline{M}_{INTP \to CAL}$ from the second of the matrix equations. Knowing the XYZ values measured from the CRT and the interpolated RGB values, you can determine the matrix coefficients through linear regression. (If you are lucky enough to have a color scientist at your disposal, he or she can analytically calculate this matrix from the white-point and phosphor chromaticities of your monitor.)

Now, using the first matrix equation and

the matrix inverse of your newly determined $\overline{M}_{INTP \rightarrow CRT}$, you calculate the color-calibrated RGB values corresponding to the *XYZ* values you measured from the test chart. The result is a set of color-calibrated RGB values that correspond to the original CFA-interpolated RGB values. Using linear regression once again, you can calculate your desired matrix, $\overline{M}_{INTP \rightarrow CAL}$.

This entire color calibration is based on 3×3 matrices. However, there is nothing to say you cannot use larger matrices. In some cases, larger matrices may yield better accuracy. For example, instead of the simple 3×3 $\overline{M}_{INTP \rightarrow CAL}$, you might want to calculate a 3×6 matrix:

$$\begin{pmatrix} R_{CAL} \\ G_{CAL} \\ B_{CAL} \end{pmatrix} = \overline{M}_{INTP \rightarrow CAL} \begin{pmatrix} R_{INTP} \\ G_{INTP} \\ B_{INTP} \\ R^2_{INTP} \\ G^2_{INTP} \\ B^2_{INTP} \end{pmatrix}$$

You can add as many terms as you wish into the calculation. For greater flexibility, you can use three-dimensional lookup tables to implement the calculation.[8] However, this does add to the complexity.

Revisiting the CFA

Wow! We have come a long way from our discussion of CFAs. Now that you have seen how to manipulate CFA data and what some of the pitfalls are, we can talk about picking (or designing) the right CFA for your system.

Several factors influence the CFA's design. First, the individual CFA filters are usually layers of transmissive organic or pigment dyes. Manufacturing concerns usually limit the choice of dyes. So if you decide for some reason that you want a light purple pixel in your array, you'd better check whether your detector manufacturer can meet this specification. Ensuring that the dyes have the right mechanical properties—such as ease of application, durability, and resistance to humidity and other atmospheric stresses—is a challenging task. This makes it difficult, at best, to fine-tune the spectral responsivities.

Given a basic set of CFA dyes, such as cyan, magenta, and yellow, you must still choose what colors to create. For example, you can make cyan, magenta, and yellow pixels. Or you can make red by layering magenta and yellow, green by layering cyan and yellow, and blue by layering cyan and magenta. If you make these RGB pixels, light must now go through two layers of dye before reaching the pixel's silicon. As a result, your system is less sensitive to light than if you used cyan, magenta, and yellow pixels. So, light sensitivity is an important trade-off.

Having decided on your colors, you then decide how to arrange them within the array. There are several things to think about. First, how will your luminance channel be interpolated? You would like to use a method that produces as little blurring of the luminance record as possible. If you use a linear convolution, try to make the required convolution kernel as small as possible in size—for example, 2×2, 1×3, or 1×2. If you are using an adaptive strategy, do you wish to consider luminance data in each direction—diagonally as well as horizontally and vertically? Of course, you have to ask how much computing power you will have at your disposal and how much time you are allowed to complete the calculations on each image. All these considerations will feed into the design process for your CFA pattern. By the way, as you make decisions, you will have to check constantly with your sensor manufacturer, because not every arrangement of colors may be practical.

Horses of different colors: Color-processing sideshows

While the CFA and the color calibration are the players on the center stage of a digital camera's color-processing chain, there are many important sideshows that can make or break an engineering job. Three of the most important are antialiasing, infrared rejection, and white-point correction.

Aliasing

This phenomenon occurs when you try to capture a spatial frequency that is finer than what the spacing between pixels in the sensor can support. Examples of real-world high-frequency patterns that can give a digital camera heartburn are fine decorative patterns in clothing (like tweeds) and picket fences in the distance. (At this point we are talking about the

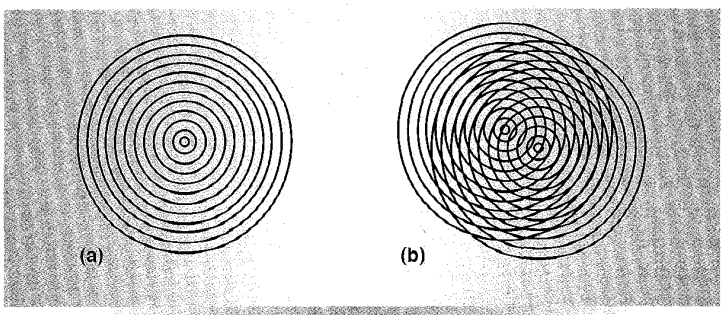

Figure 9. Bull's-eye (a) and two overlapping bull's-eyes with moiré patterns (b).

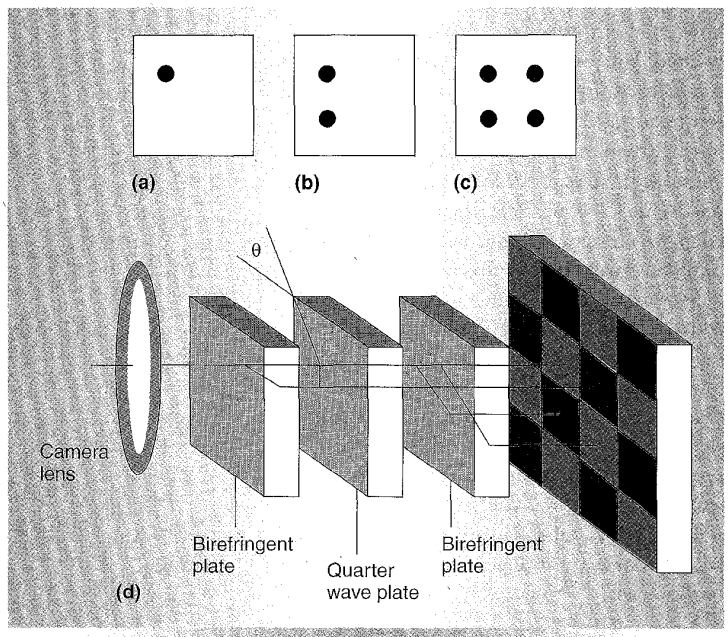

Figure 10. Example of a one-to-four beam-splitting, antialiasing filter. A single beam (a) before the filter is split into two beams (b) by the first birefringent layer and then into four beams (c) by the second birefringent layer. The full light path (d) in profile.

image projected onto the CFA, not the original scene.) If you do not take preventative actions, low-frequency moiré patterns will show up in the region containing the high-frequency information. Figure 9 shows a computer-generated version of this phenomenon.

One important problem with aliasing is that it messes with the CFA interpolation algorithm and can cause catastrophic failures with the interpolation process. This is especially problematic when the aliasing occurs only in one or two of the color channels. The solution is to incorporate an antialiasing filter into the optical chain. You place this, not too surprisingly, in the position marked *filter* in Figure 2. The primary purpose of the antialiasing filter is to exclude, usually by blurring, the high spatial frequencies that would show up as moiré patterns in the final image. You can exploit several optical phenomena to create this kind of filter. Two of the more common are polarization and phase delay.

With an antialiasing filter based on polarization,[9] you orient two or more pieces of birefringent material, such as quartz or calcite, to split an incoming light beam into several outgoing beams, directing each beam to a different pixel. The classic example is the one-to-four split. Figure 10a shows the location of the original beam of light before striking the first piece of birefringent material. Figure 10b shows that in passing through the first piece of birefringent material, one beam of light is split into two beams of light. (Physicists would call these the ordinary and extraordinary rays, but we'll refrain from an optics lecture.) The second piece of birefringent material splits each of these two beams again (Figure 10c) to produce four beams of light.

Figure 10d is a side view of this process. Usually, for a CFA such as the RGB pattern of Figure 3, you adjust the thickness of each layer and the spacing between the layers so that the four beams of light fall on four adjacent pixels. This provides a reasonable amount of high-frequency blur to reduce most of the aliasing without blurring the entire image unacceptably. There are many ways to make these kinds of polarization-based antialiasing filters, usually by using different numbers of layers and different kinds of birefringent materials. One of the biggest drawbacks of this approach is that optical-grade birefringent material is expensive. Hence, it would be a costly addition to your parts list.

Antialiasing filters based on phase delay are a different kind of beast. Here, you etch a pattern into the surface of a single piece of optical material to cause some light in the image to pass through a slightly thicker portion of the filter than the rest.[10] As a result, the light that passes through more filter material suffers a phase delay and begins to interfere with the light taking the other path. By adjusting the amount of the phase delay, you can cause higher spatial frequencies to disappear from

your image long before the lower spatial frequencies do.

Figure 11 shows such an antialiasing filter; it has tiny disks etched into the surface. The disks are randomly placed and have random diameters. It is possible to vary the heights of the disks, too. Also, there's nothing sacred about disks; you can use virtually any pattern. An advantage of phase-delay filters is that they are relatively inexpensive to make. A disadvantage is that they can act like diffraction gratings if you do not choose your pattern parameters wisely. A diffraction grating in your optical path will almost guarantee an unusable image!

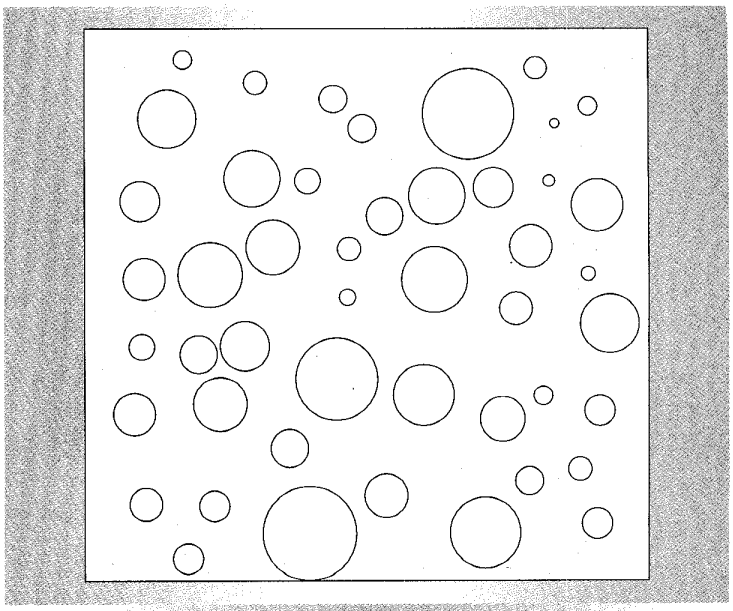

Figure 11. Typical phase-delay antialiasing filter.

Infrared rejection

IR rejection is important to have in your optical path, because most CFA dyes transmit light beyond 700 nm. While you will not *see* any light beyond 700 nm or so, your silicon detector may be very sensitive at these wavelengths. If the human visual system disregards the information at these wavelengths, it is a good bet that your digital camera should, too, if you want to represent the world as your eye sees it. IR filters fall into two general categories, depending on whether they use reflection or absorption.

If you coat an optical surface with several layers of carefully chosen dielectric materials, you can create what is known as a hot mirror. (The optical surface could be a surface on your camera lens or a separate slab of optical material placed somewhere in the optical path before your CFA.) A hot mirror will transmit any light below its cutoff wavelength (usually in the 600- to 700-nm range) and reflect the higher wavelengths. It is called a hot mirror because this reflected, infrared energy is also heat energy. (Yes, a cold mirror does the exact opposite, transmitting the IR and reflecting the visual, but it really does not have an application in digital cameras.) Placing a hot mirror in your optical path will help prevent weird color shifts when taking a photo of materials with high reflectances in the infrared, such as some flowers and many black dyes used in clothing.

You can achieve the same result with a filter made of an optical material that absorbs infrared radiation. These kinds of filters typically are light blue or light green in color. They work just fine unless they absorb too much heat all at once. If they are exposed to too much IR without proper cooling, they can crack or shatter. For this reason, some digital camera manufacturers prefer a hot mirror solution.

White-point correction

This is something we take for granted in our human visual systems. Suppose you take a white piece of paper from the fluorescent lighting of your office to the daylight of the parking lot, and then home to the incandescent lighting of your living room. You will continually see the white paper as being white. If you take pictures of that same piece of paper with your digital camera under the three lighting conditions, none of these images will look truly white. Also, the pictures taken under fluorescent and incandescent lights look, accordingly, greenish and yellowish compared to the daylight picture.

Your human visual system constantly adapts to the current source of light and makes adjustments so that objects you know to be white continue to appear white. You have to build the same type of mechanism into your digital-camera image-processing path, or whites (and all the rest of the colors) will never appear right except under the exact lighting conditions used

Figure 12. Kodak Professional DCS 315 digital camera.

during your color calibration operation.

There are two general strategies for making white-point corrections in digital cameras. The simple one is to perform a fixed white-point correction. To do this, you simply capture an image of a white or light gray color patch under each kind of illuminant you wish to support. A typical set might be daylight, fluorescent, tungsten (incandescent), and flash. (Most flashes approximate daylight, so this last entry might be redundant.) You determine the average code value for each of your color channels from each image, assuming you have first transformed the data to be linear with scene exposure. You then calculate the ratio between each color channel and the reference color channel (usually the green one) and store the ratio in a table for future use. A user capturing an image specifies to the image-processing software which illuminant the image was captured under. The software retrieves the appropriate set of color channel ratios from the table and applies them to the image.

As a numeric example, let's assume that you captured a gray chart under tungsten illumination and got the average code values of 153 for red, 120 for green, and 100 for blue. You store the ratios red/green (1.28) and blue/green (0.83) in your table under tungsten. Now you capture a picture of that white piece of paper under incandescent lighting. Let's say a typical pixel in that white sheet of paper comes out to be 210 for red, 166 for green, and 132 for blue. You divide the red value by your stored red/green value and the blue value by your stored blue/green value. The result is 164 for red, 166 for green, and 159 for blue. These numbers aren't perfect because the lighting in your house is not identical to that in the calibration lab. However, the paper will certainly look a whole lot whiter after white-point correction than it did before! You would apply this same set of calculations for each pixel in the image, dividing each red value by 1.28 and each blue value by 0.83.

Sometimes, however, a fixed white-point correction does not meet your needs. Maybe you do not know what illuminant was used, or maybe no illuminant in the table provides a close enough match. Or maybe you don't want the camera user to have to bother making this specification. In that case, you can perform an automatic white-point correction. This can be a very complex calculation that begins to resemble a neural network in its extreme implementation. (How do you think the human visual system does it?)

One of the simplest and most useful assumptions to make at this point is the Gray World Theory: The set of all the visual scenes in the world integrate to gray. The most direct (and not very good) way to apply this theory to make an automatic white-point correction is to take your image and calculate its average code value for each color channel. From these average code values, you can calculate ratios between each color channel and a reference color channel. (Again, green is the most common choice for the reference channel.) You then use these ratios to correct your image, just as you used the ratios stored in your table for fixed white-point correction. The problem, of course, is that group shot you just took of your office buddies with everyone wearing bright green because it's St. Patrick's Day! The chances of that image averaging to gray are about nil. So what do you do? Here you start significantly expanding the complexity of your white-point correction algorithm.

You can start by identifying and throwing out brightly colored pixels from your averaging calculation. The hard part about this is defining the boundary between the colors you want to keep and the ones you want to exclude. You might also want to make some

educated guesses about what the illuminant was at the time of capture. If the scene is relatively dim, it is more likely an indoor (fluorescent or incandescent) shot. A bright image is probably an outdoor (daylight) shot. If the average channel values seem to be on the reddish side, maybe the image was taken under incandescent lighting. Of course, if the flash went off and you recorded that fact when you stored your image data, you know what the illuminant was! You can take all these guesses and inferences and assign probabilities to them and use these probabilities as weights to create a composite set of color correction ratios from your table of fixed illuminant color correction values.

Here is one last thought to ponder on white-point correction. If you can determine that your scene was shot during sunset—or that the scene *was* the sunset—you might not want to make much white-point correction, if any. Leaving the image looking "warm" could be exactly what the photographer wanted. After all, isn't that what the human visual system does?

Putting together the colors of the rainbow

We've gone through the major steps in color processing that go into a finished digital-camera image. So how can a real camera orchestrate all these steps? One interesting example to examine is the Kodak Professional DCS 315 digital camera, shown in Figure 12.[11] It gives the user two ways to perform color processing: in the camera or using the provided software on the user's computer after the unprocessed images have been downloaded.

If the user chooses to have the camera process the color, the internal firmware performs an adaptive CFA interpolation, automatic white-point and exposure correction, and color calibration. It finishes by producing a JPEG-compressed image.

A user who decides to have the host computer perform color processing has more options. Instead of automatic white-point correction, the user may choose fixed white-point correction and select the appropriate illuminant. Alternatively, the user can even perform a click-balance by pointing to a white or gray region in the image and letting the software calculate the appropriate correction factors from that area. The user may also forgo the JPEG compression step.

At this point, we can emphatically restate that digital cameras are fun! Next time you press the button and see the results right away, think of all the processing that went into producing that great-looking image. With future research, digital-camera color processing will produce even better images. New CFA patterns and better image-processing algorithms will let digital cameras reproduce colors more accurately while reducing the visibility of noise. And the computation time to produce the image will shrink still further. Why, even the Wizard of Oz himself would be proud of a trick like that! MICRO

References

1. "Digital Photography," *Digital Consumer Electronics Handbook,* R. Jurgen, ed., McGraw-Hill, New York, 1997.
2. K.A. Parulski, "Color Filter Arrays and Processing Alternatives for One-Chip Cameras," *IEEE Trans. Electron Devices,* Vol. ED-32, No. 8, Aug. 1985, pp. 1381-1389.
3. J.E. Adams, Jr., "Interactions Between Color Plane Interpolation and Other Image Processing Functions in Electronic Photography," *Proc. SPIE,* Vol. 2416, SPIE—Int'l Soc. for Optical Engineering, Bellingham, Wash., 1995, pp. 144-155.
4. J.E. Adams, Jr., "Design of Practical Color Filter Array Interpolation Algorithms for Digital Cameras," *Proc. SPIE,* Vol. 3028, SPIE, 1997, pp. 117-125.
5. E.J. Giorgianni and T.E. Madden, *Digital Color Management: Encoding Solutions,* Video Images, Chapter 4, Addison-Wesley, Reading, Mass., 1998.
6. R.W.G. Hunt, *The Reproduction of Color,* Fourth Ed., Appendix 2, Fountain Press, Tolworth, Surrey, UK, 1987.
7. G. Wyszecki and W.S. Stiles, *Color Science: Concepts and Methods, Quantitative Data and Formulae,* Second Ed., Appendix of Extended Tables and Illustrations, John Wiley & Sons, New York, 1982.
8. J.M. Kasson et al., "Performing Color Space Conversions with Three Dimensional Linear Interpolation," *J. Electronic Imaging,* Vol. 4, No. 3, 1995, pp. 225-250.
9. J. Greivenkamp, "Color Dependent Optical Prefilter for the Suppression of Aliasing