Image Sensor Color Calibration Using the Zynq-7000 All Programmable SoC

by Gabor Szedo
Staff Video Design Engineer
Xilinx Inc.
gabor.szedo@xilinx.com

Steve Elzinga
Video IP Design Engineer
Xilinx Inc.
steve.elzinga@xilinx.com

Greg Jewett
Video Marketing Manager
Xilinx Inc.
greg.jewett@xilinx.com

Xilinx image- and video-processing cores and kits provide the perfect prototyping platform for camera developers.

Image sensors are used in a wide range of applications, from cell phones and video surveillance products to automobiles and missile systems. Almost all of these applications require white-balance correction (also referred to as color correction) in order to produce images with colors that appear correct to the human eye regardless of the type of illumination—daylight, incandescent, fluorescent and so on.

Implementing automatic white-balance correction in a programmable logic device such as a Xilinx® FPGA or Zynq™-7000 All Programmable SoC is likely to be a new challenge for many developers who have used ASIC or ASSP devices previously. Let's look at how software running on an embedded processor, such as an ARM9 processing system on the Zynq-7000 All Programmable SoC, can control custom image- and video-processing logic to perform real-time pixel-level color/white-balance correction.

To set the stage for how this is done, it's helpful to first examine some basic concepts of color perception and camera calibration.

CAMERA CALIBRATION

The measured color and intensity of reflections from a small, uniform surface element with no inherent light emission or opacity depend on three functions: the spectral power distribution of the illuminant, $I(\lambda)$; the spectral reflective properties of the surface material, $R(\lambda)$; and the spectral sensitivities of the imager, $S(\lambda)$.

The signal power measured by a detector can be expressed as:

$$P=\int_0^\infty I(\lambda)\mathrm{R}(\lambda)\mathrm{S}(\lambda)d\lambda$$

In order to get a color image, the human eye, as well as photographic and video equipment, uses multiple adjacent sensors with different spectral responses. Human vision relies on three types of light-sensitive cone cells to formulate color perception. In developing a color model based on human perception, the International Commission on Illumination (CIE) has defined a set of three color-matching functions, $\bar{x}(\lambda)$, $\bar{y}(\lambda)$ and $\bar{z}(\lambda)$. These can be thought of as the spectral sensitivity curves of three linear light detectors that yield the CIE XYZ tristimulus values P_x, P_y, and P_z, known collectively as the "CIE standard observer."

Digital image sensors predominantly use two methods to measure tristimulus values: a color filter array overlay above inherently monochromatic photodiodes; and stacked photodiodes that measure the absorption depth of photons, which is proportional to wavelength λ.

However, neither of these methods creates spectral responses similar to those of the human eye. As a result, color measurements between different photo detection and reproduction equipment will differ, as will measurements between image sensors and human observers when photographing the same scene—the same $(I\lambda)$ and $(R\lambda)$.

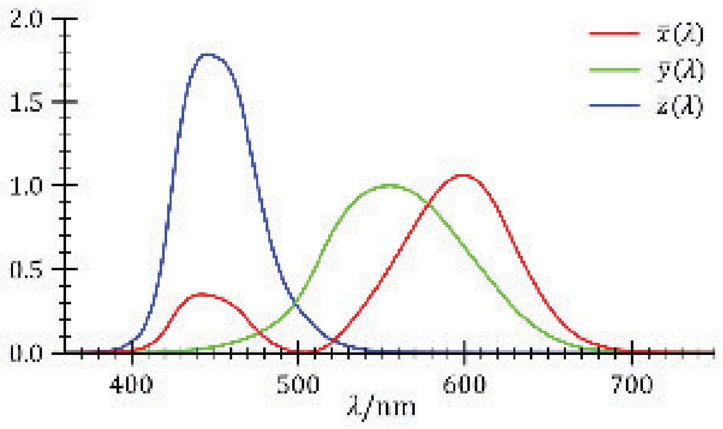
Thus, the purpose of camera calibration is to transform and correct the tristimulus values that a camera or image sensor measures, such that the spectral responses match those of the CIE standard observer.

WHITE BALANCE

You may view any object under various lighting conditions—for example, illuminated by natural sunlight, the light of a fire, fluorescent or incandescent bulbs. In all of these situations, human vision perceives the object as having the same color, a phenomenon called "chromatic adaptation" or "color constancy." However, a camera with no adjustment or automatic compensation for illuminants may register the color as varying. When a camera corrects for this situation, it is referred to as white-balance correction.

According to the top equation at the right of Figure 1, describing spectra of the illuminants, the reflective properties of objects in a scene and the spectral sensitivity of the detector all contribute to the resulting color measurement. Therefore, even with the same detectors, measurement results will mix information from innate object colors and the spectrum of the illuminant. White balancing, or the separation of innate reflective properties $R(\lambda)$ from the spectrum of the illuminant $I(\lambda)$, is possible only if:

- Some heuristics, e.g. the spatial frequency limits on the illuminant, or object colors are known a priori. For example, when photographing a scene with natural sunlight, it is expected that the spec-



$$P_x = \int_0^{\infty} I(\lambda)R(\lambda)\bar{x}(\lambda)\mathrm{d}\lambda$$

$$P_y = \int_0^{\infty} I(\lambda)R(\lambda)\bar{y}(\lambda)\mathrm{d}\lambda$$

$$P_z = \int_0^{\infty} I(\lambda)R(\lambda)\bar{z}(\lambda)\mathrm{d}\lambda$$

Figure 1 – Spectral responses of the "standard observer"

tral properties of the illuminant will remain constant over the entire image. Conversely, when an image is projected onto a white screen, spectral properties of the illuminant change dramatically from pixel to pixel, while the reflective properties of the scene (the canvas) remain constant. When both illuminant and reflective properties change abruptly, it is very difficult to isolate the scene's objects and illuminants.

- Detector sensitivity $S(\lambda)$ and the illuminant spectrum $I(\lambda)$ do not have zeros in the range of spectrum observed. You cannot gain any information about the reflective properties of objects outside the illuminant spectrum. For example, when a scene is illuminated by a monochromatic red source, a blue object will look just as black as a green one.

PRIOR METHODS

In digital imaging systems, the problem of camera calibration for a known illuminant can be represented as a discrete, three-dimensional vector function:

$$\underline{x}'=F(\underline{x})$$

where $F(\underline{x})$ is the mapping vector function and $\underline{x}$ is the discrete (typically 8-, 10- or 12-bit) vector of R,G,B principal color components. Based on whether you are going to perform mapping linearly and whether color components are corrected independently, the mapping function can be categorized as shown in Table 1.

THE VON KRIES HYPOTHESIS

The simplest, and most widely used method for camera calibration is based on the von Kries Hypothesis [1], which aims to transform colors to the LMS color space, then performs correction using only three multipliers on a per-channel basis. The hypothesis rests on the assumption that color constancy in the human visual system can be achieved by individually adapting the gains of the three cone responses; the gains will depend on the sensory context, that is, the color history and surround. Cone responses from two radiant spectra, f_1 and f_2, can be matched by an appropriate choice of diagonal adaptation matrices D_1 and D_2 such that $D_1\ S\ f_1 = D_2\ S\ f_2$, where S is the cone sensitivity matrix. In the LMS (long-, medium-, short-wave sensitive cone-response space),

$$D = D_1^{-1}D_2 = \begin{bmatrix} L_2/L_1 & 0 & 0 \\ 0 & M_2/M_1 & 0 \\ 0 & 0 & S_2/S_1 \end{bmatrix}$$

The advantage of this method is its relative simplicity and easy implementation with three parallel multipliers as part of either a digital image sensor or the image sensor pipeline (ISP):

$$\begin{bmatrix} L' \\ M' \\ S' \end{bmatrix} \begin{bmatrix} k_L & 0 & 0 \\ 0 & k_M & 0 \\ 0 & 0 & k_S \end{bmatrix} \begin{bmatrix} L \\ M \\ S \end{bmatrix}$$

In a practical implementation, instead of using the LMS space, the RGB color space is used to adjust channel gains such that one color, typically white, is represented by equal R,G,B values. However, adjusting the perceived cone responses or R,G,B values for one color does not guarantee that other colors are represented faithfully.

	Linear	Nonlinear
Independent	von Kries	Component correction
Dependent	Color-correction matrix	Full lookup table

Table 1 – Camera calibration methods

COMPONENT CORRECTION

For any particular color component, the von Kries Hypothesis can only represent linear relationships between input and output. Assuming similar data representation (e.g. 8, 10 or 12 bits per component), unless k is 1.0, some of the output dynamic range is unused or some of the input values correspond to values that need to be clipped/clamped. Instead of multipliers, you can represent any function defining input/output mapping using small, component-based lookup tables. This way you can address sensor/display nonlinearity and gamma correction in one block. In an FPGA image-processing pipeline implementation, you can use the Xilinx Gamma Correction IP block to perform this operation.

FULL LOOKUP TABLE

Camera calibration assigns an expected value to all possible camera input tristimulus values. A brute-force approach to the problem is to use a large lookup table containing expected values for all possible input RGB values. This solution has two drawbacks. The first is memory size. For 10-bit components, the table is 2^{30} word (4 Gbytes) deep and 30 bits wide. The second problem is initialization values. Typically only a few dozen to a few hundred camera input/expected-value pairs are established via calibration measurements. The rest of the sparse lookup-table values have to be interpolated. This interpolation task is not trivial, as the heterogeneous component input-to-output functions are neither monotone nor smooth. Figure 2a presents the measured vs. expected-value pairs for R,G,B input (rows) and output (columns) values.

A visual evaluation of empirical results interpolated (Figure 2b) did not show significant quality improvement over a gamma-corrected, color-correction matrix-based solution. Most image- or video-processing systems are constrained on accessible bandwidth to external memory. The large size of the lookup table, which mandates external memory use; the significant bandwidth demand the per-pixel accesses pose; and the static nature of lookup-table contents (difficult to reprogram on a frame-by-frame basis) limit practical use of a full LUT-based solution in embedded video- and image-processing applications.

COLOR-CORRECTION MATRIX

The calibration method we describe in this article demonstrates how you can use a 3x3-matrix multiplier to perform a coordinate transformation aiming to orthogonalize measured red, green and blue components. The advantage of this method over the von Kries approach is that all three color channels are involved in the calibration process. For example, you can incorporate information from the red and blue channels when adjusting green-channel gains. Also, this solution lends itself well for camera calibration and white-balance correction to be performed simultaneously using the same module, updating matrix coefficients to match changing illuminants smoothly on a frame-by-frame basis.

The two simplest algorithms for white-balance correction—the Gray World and the White Point algorithms—use the RGB color space.

The Gray World algorithm [2] is based on the heuristics that although different objects in a scene have different, distinct colors, the average of scene colors (average of red, green and blue values) should result in a neutral, gray color. Consequently, the differences in R,G,B color values

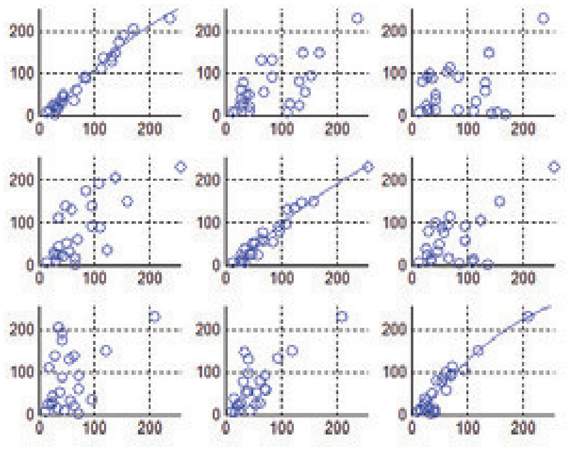


Figure 2a – R,G,B measured vs. expected mapping values

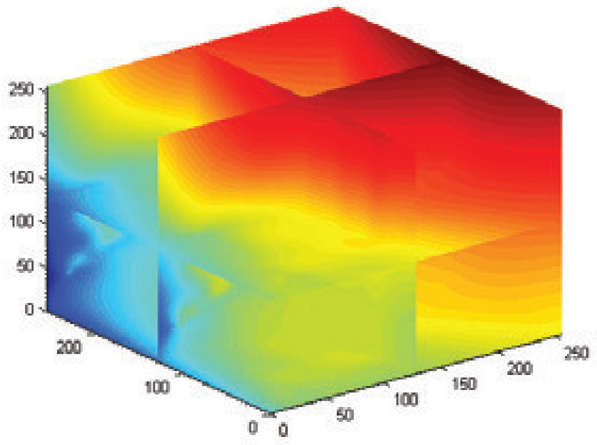


Figure 2b – R component output as a function of R,G,B inputs

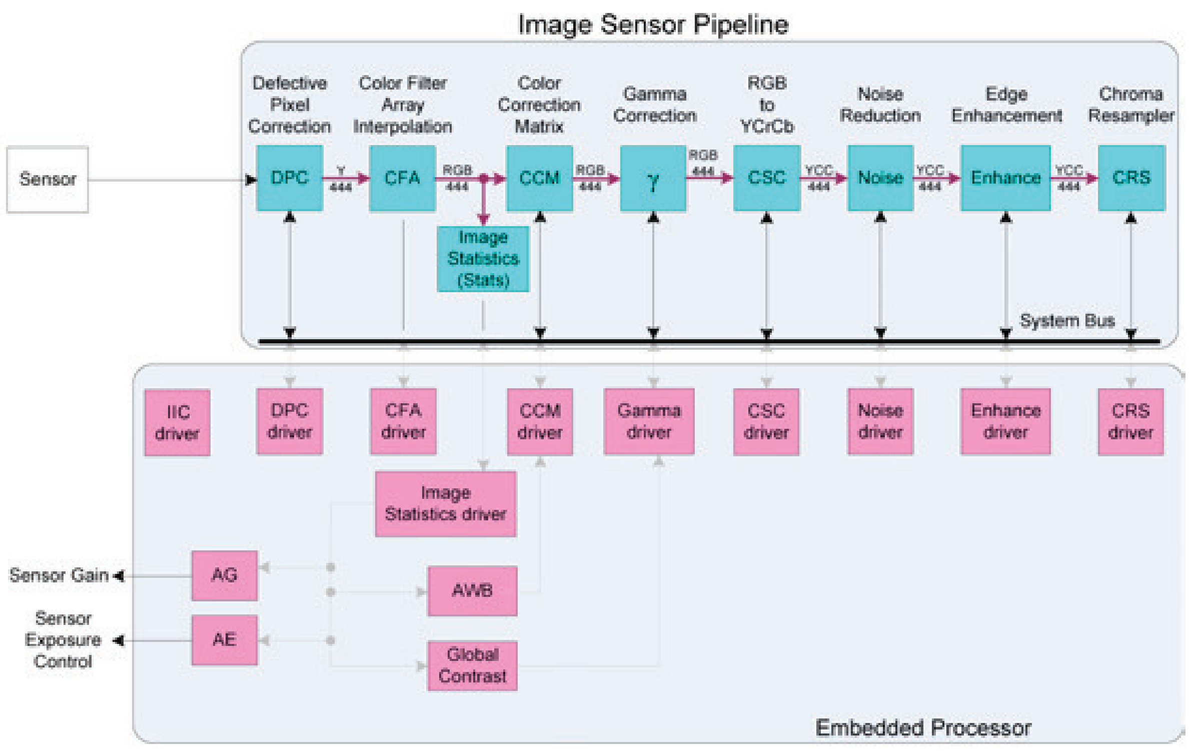


Figure 3 – Typical image sensor pipeline

averaged over a frame provide information about the illuminant color, and correction should transform colors such that the resulting color averages are identical. The Gray World algorithm is relatively easy to implement. However, it introduces large errors in which inherent scene colors may be removed or altered in the presence of large, vivid objects.

The White Point [2] algorithm is based on the assumption that the lightest pixels in an image must be white or light gray. The difference in red, green and blue channel maxima provides information about the illuminant color, and correction should transform colors such that the resulting color maxima are identical. However, to find the white point, it's necessary to rank pixels by luminance values. In addition, you may also have to perform spatiotemporal filtering of the ordered list to suppress noise artifacts and aggregate ranked results into a single, white color triplet. The advantage of using the White Point algorithm is easy implementation. The downside is that it too can introduce large errors and may remove inherent scene colors. Also, the method is easily compromised by saturated pixels.

More-refined methods take advantage of color-space conversions, where hue can be easily isolated from color saturation and luminance, reducing three-dimensional color correction to a one-dimensional problem.

For example, color gamut mapping builds a two-dimensional histogram in the YCC, YUV, L*a*b* or Luv color spaces, and fits a convex hull around the base of the histogram. The UV or (Cr, Cb) averages are calculated and used to correct colors, such that the resulting color UV, or CbCr histograms are centered on the neutral, or gray point in the YUV, YCC, Luv or Lab space. The advantage of these methods is better color performance. The disadvantage is that implementation may require floating-point arithmetic.

All of the methods described above may suffer from artifacts due to incorrect exposure settings or extreme dynamic ranges in scene illumination. For example, saturated pixels in an image that's illuminated by a bright light source with inherent hue, such as a candlelit picture with the flame in focus, may lead to fully saturated, white pixels present on the image.

OTHER WAYS TO IMPROVE WHITE-BALANCE RESULTS

Separating foreground and background is another approach to color correction. The autofocus logic, coupled to multizone metering, in digital cameras allows spatial distinction of pixels in focus around the center and the background around the edges. The assumption is that the objects photographed, with only a few dominant colors, are in focus at the center of the image. Objects in the distance are closer to the edge, where the Gray World hypothesis prevails.

Another technique centers on shape detection. Face or skin-color detection helps cameras identify image content with expected hues. In this case, white-balance correction can be limited to pixels with known, expected hues. Color correction will take place to move the colors of these pixels closer to the expected colors. The disadvantage of this method is the costly segmentation and recognition logic.

Most commercial applications combine multiple methods, using a strategy of adapting to image contents and photographic environment. [2]

ISPs FOR CAMERA CALIBRATION AND COLOR CORRECTION

Our implementation uses a typical image sensor pipeline, illustrated in Figure 3. We built the hardware components of the ISP (the blue blocks) with Xilinx image-processing cores using configurable logic. Meanwhile, we designed the camera calibration and white-balancing algorithms as C code (pink blocks) running on one of the embedded ARM processors. This same ARM processor runs embedded Linux to provide a user interface to a host PC.

The portion of the ISP relevant to white balancing and camera calibration is the feedback loop, including:

- The image statistics module, which gathers zone-based statistical data on a frame-by-frame basis;
- The embedded drivers and the application software, which analyzes the statistical information and programs the color-correction module on a frame-by-frame basis;
- The color-correction module, which performs color transformations on a pixel-by-pixel basis.

We implemented the ISP as part of the Zynq Video and Imaging Kit (ZVIK) 1080P60 Camera Image Processing Reference Design.

DETAILED ALGORITHM DESCRIPTION

In order to calibrate the colors of our sensor, we used an off-the-shelf color-viewing booth (X-Rite Macbeth Judge II), or light box, which has four standard illuminants with known spectra: simulated daylight, cool-white fluorescent, warm fluorescent and incandescent. We also used an off-the-shelf color target (an X-Rite ColorChecker 24 Patch Classic) with color patches of known reflective properties and expected RGB and sRGB values.

To begin the process of implementing the camera-calibration algorithm, we first placed the color target in the light booth, flat against the gray background of the light box. We made sure to position the color target such that illumination from all light sources was as even as possible.

Next we captured the images taken by the sensor to be calibrated, with the all illuminants, with no color correction (using "bypass" color-correction settings: identity matrix loaded to the color-correction matrix).

We then used MATLAB® scripts available from Xilinx to assist with compensating for barrel (geometric) lens distortion and lens shading (light intensity dropping off toward the corners). The MATLAB script allows us to identify control points on the recorded images, then warps the image to compensate for barrel distortion. The rest of the script estimates horizontal and vertical light drop-off using the background around the registered ColorChecker target.

In order to attenuate measurement noise, we identified rectangular zones within the color patches. Within these zones, we averaged (R,G,B) pixel data to represent each color patch with an RGB triplet. A MATLAB script with a GUI helps identify the patch centers and calculates averaged RGB triplets corresponding to the expected RGB values of each color patch (R_e, G_e, B_e).

We implemented the simulated annealing optimization method to identify color-correction coefficients and offsets. The measured uncalibrated (R,G,B) color triplets are transformed to corrected (R',G',B') triplets using the Color Correction module of Figure 3.

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} R_{offs} \\ G_{offs} \\ B_{offs} \end{bmatrix}$$

The simulated annealing algorithm minimizes an error function returning a scalar. In the following discussion (R_k, G_k, B_k) reference a subset or superset of measured color-patch pixel values. The user is free to limit the number of patches included in the optimization (subset), or include a particular patch multiple

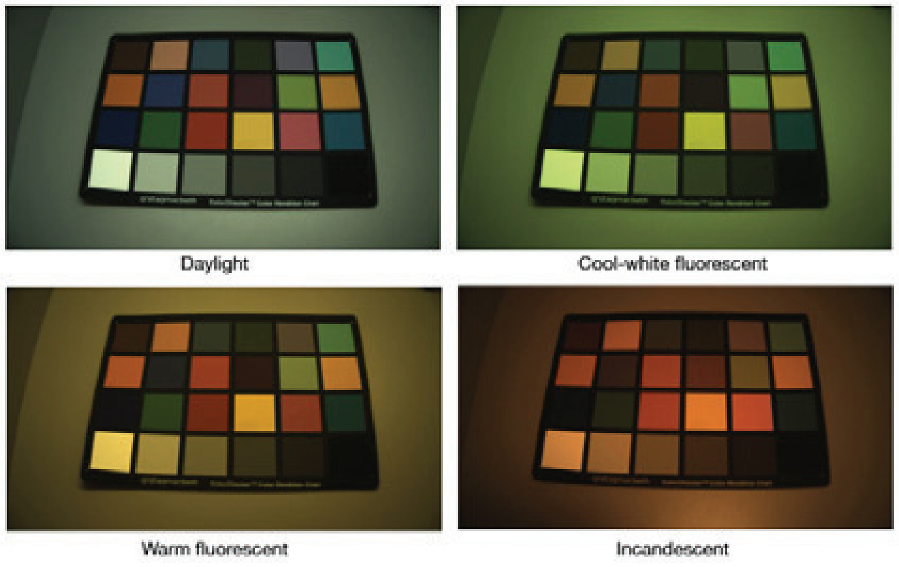


Figure 4 – Sensor images with different illuminants before lens correction

times, thereby increasing its relative weight during the optimization process. The integer n represents the number of color patches selected for inclusion in the optimization. If all patches are included exactly once in the optimization process, for the X-Rite ColorChecker 24 Patch Classic, n=24.

As the optimization algorithm has freedom to set 12 variables (CCM coefficients and offsets) only, typically no exact solution exists that maps all measured values to precisely the expected color patch values. However, the algorithm seeks to minimize an error function to provide optimal error distribution over the range of patches used.

We set up error functions that calculate one of the following:

- The sum of squared differences between expected and transformed triplets in the RGB color space:

$$E=\sum_{k=0}^{n} (R_k'-Re_k)^2+(G_k'-Ge_k)^2+B_k'-Be_k)^2$$

- The sum of absolute differences between expected and transformed triplets in the RGB color space:

$$E=\sum_{k=0}^{n} |R_k'-Re_k|+|G_k'-Ge_k|+|B_k'-Be_k|$$

- The sum of squared differences between expected and transformed triplets in the YUV color space:

$$E=\sum_{k=0}^{n} (U_k'-Ue_k)^2+(V_k'-Ve_k)^2$$

- Or absolute differences between expected and transformed triplets in the YUV color space:

$$E=\sum_{k=0}^{n} |U_k'-Ue_k|+|V_k'-Ve_k|$$

where U' and V' correspond to R'G'B' values transformed to the YUV color space. Similarly, error functions can be set up to the L*u*v* or L*a*b* color spaces. You can use any of the above error functions in the simulated annealing minimization.

WHITE BALANCING

Using the camera-calibration method above, we established four sets of color-correction coefficients and offsets, CCMk, k={1,2,3,4}, that result in optimal color representation assuming that the illuminant is correctly identified. The white-balancing algorithm, implemented in software running on the embedded processor, has to perform the following operations on a frame-by-frame basis. Using statistical information, it estimates the illuminant weights (W_k). Weights are low-pass filtered to compensate for sudden scene changes, resulting in illuminant probabilities (p_k). The color-correction matrix module is programmed with the combined CCM_k values according to weights p_k.

The advantage of this method is that a linear combination of calibration CCM_k values will limit color artifacts in case scene colors and illuminant colors are not properly separated. In the case of underwater photography, for example, where a strong blue tinge is present, a simple white-balancing algorithm such as Gray World would compensate to remove all blue, severely distorting the innate colors of the scene.

For all illuminants k={1,2,3,4} with different scene setups in the light booth, we also recorded the two-dimensional YUV histograms of the scenes by binning pixel values by chrominance, and weighing each pixel by its luminance value (luminance-weighed chrominance histogram). This method de-prioritizes dark pixels, or those in which a small difference in R,G,B values results in large noise in the chrominance domain.

Using a mask, we eliminated histogram bins which pertain to vivid colors that cannot possibly originate from a neutral (gray or white) object illuminated by a typical illuminant (Figure 6). A typical mask contains nonzero values only around the neutral (white) point, where most illuminants are located. We hard-coded the masked two-dimensional histogram values H_k (x,y), as well as the CCM_k values, into the white-

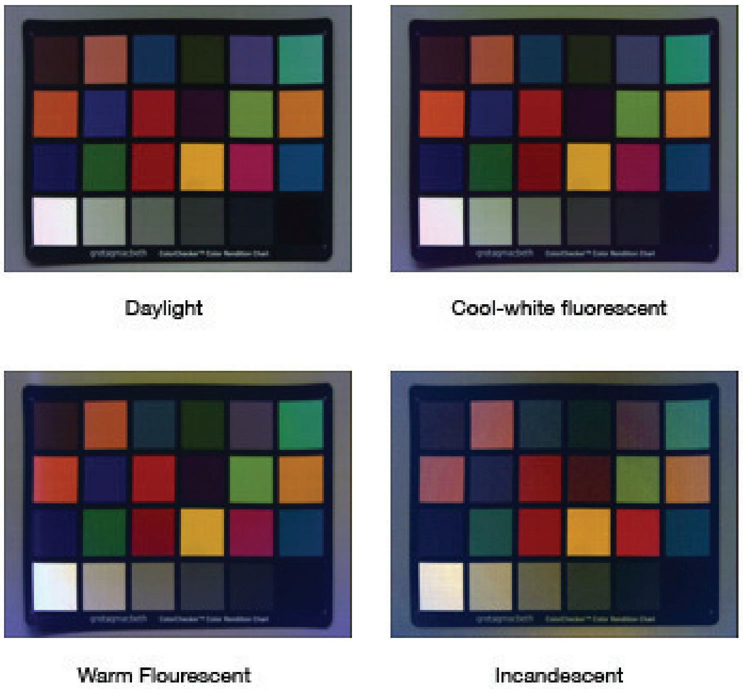


Figure 5 – Color-calibrated, lens-corrected images with different illuminants

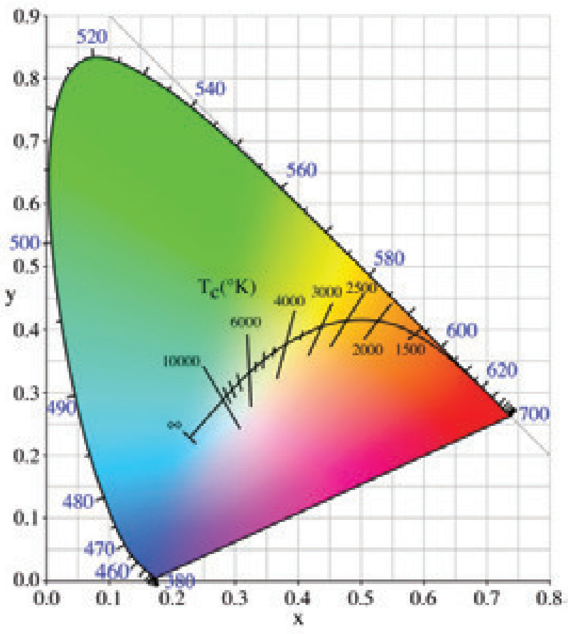


Figure 6 – Illuminants with different temperatures in CIE color space

balancing application running on the embedded processor.

During real-time operation, the white-balancing application collects similar two-dimensional, luminance-weighed chrominance histograms. The measured two-dimensional histograms are also masked, and the sum of absolute differences or sum of squared differences is calculated among the four stored histograms and the measured one:

$$D_k=\sum_{k=0}^{15}\sum_{k=0}^{15}\left(H_k(x,y)-H(x,y)\right)^2$$

where $H_k(x,y)$ are the precalculated reference two-dimensional histograms pertaining to known illuminants $\{k=1,2,3,4\}$, and $H(x,y)$ is the real-time histogram measurement.

Based on the measured histogram differences D_k, normalized similarity values are calculated using:

$$w_i=\frac{1/D_i}{\sum_{k=1}^{4}1/D_k}$$

To avoid abrupt frame-by-frame tone changes, we smoothed normalized similarity values over time. We used a simple low-pass IIR filter, implementing

$$p_i=cw_i+(1-c)p_{i-1}$$

where $0 < c < 1$ controls the impulse response of the IIR filter. The smaller the values of c, the smoother the transitions. The larger the value, the quicker the filter responds to changes in lighting conditions.

Finally, we programmed the color-correction module of the ISP (Figure 3) with a linear combination of the precalculated color-correction coefficients and offsets (CCM_k):

$$CCM=\sum_{k=1}^{4}p_kCCM_k$$

Real-time white-balance implementation results (Figure 7) from a scene illuminated by both natural daylight and fluorescent light show significant improvement in perceived image quality and color representation.

The Zynq Video and Imaging Kit, along with MATLAB scripts available from Xilinx, complement and provide an implementation example for the algorithms we have presented.

Real-time color balancing is becoming increasingly challenging as resolutions and frame rates of industrial, consumer and automotive video applications improve. The algorithm we have described illustrates how software running on an embedded processor, such as the ARM9 cores of the Zynq processing platform, can control custom image- and video-processing logic performing pixel-level color correction.

References

1. *H.Y. Chong, S.J. Gortler and T. Zickler, "The von Kries Hypothesis and Basis for Color Constancy," Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2007.*

2. *S. Bianco, F. Gasparini and R. Schettini, "Combining Strategies for White Balance," Proceedings of SPIE (2007), Volume 39, pages 65020D-65020D-9*

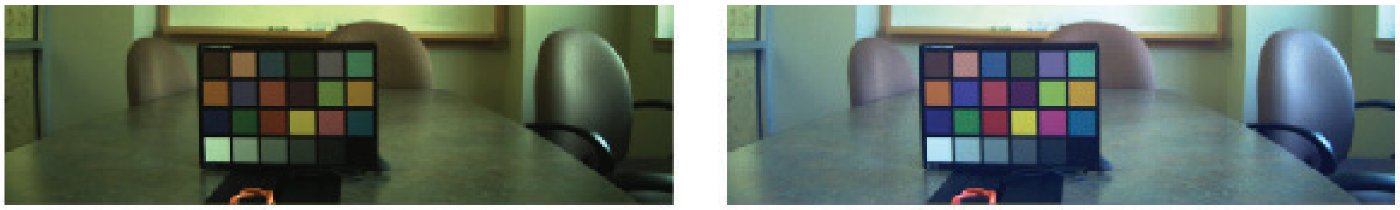


Figure 7 – Scene captured with no correction (left) and with white-balance correction