Advanced Fault-Tolerance Techniques For A Color Digital Camera-On-A-Chip *

Israel Koren, Glenn Chapman† and Zahava Koren

Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, MA 01003
E-mail: koren@ecs.umass.edu
† School of Engineering Science
Simon Fraser University, Burnaby BC, Canada V5A 1S6
E-mail: glennc@cs.sfu.ca

Abstract

Color digital imagers contain Red, Green and Blue subpixels within each color pixel. Defects that develop either at fabrication time or due to environmentally induced errors over time can cause a single color subpixel (e.g., R) to fail, while leaving the remaining colors intact. This paper investigates seven software correction algorithms that interpolate the color of a pixel based on its nearest neighbors. Using several measurements of color error, all seven methods were investigated for a large number of digital images. Interpolations using only information from the single failed color (e.g., R) in the neighbors gave the poorest results. Those using all color measurements and a quadratic interpolation formula, combined with the remaining subpixel colors (e.g., G and B) produced significantly better results. A formula developed using the CIE color coordinates of Tristimulus values (X, Y, Z) yielded the best results.

1: Introduction

Digital cameras on-a-chip are becoming more common (e.g., [6, 7]) and are expected to be used in many industrial and consumer products. Many systems are shifting to the Active Pixel Sensor CMOS detectors, which have the advantage of lower cost, single voltage, lower power consumption, higher level of device integration, and last but not least, the possibility of defect repairs.

In [5], we evaluated the benefits of adding redundant rows and columns to the pixel array. Since a spare row (column) of pixels can only replace a boundary row (column) rather than any faulty row (column), this technique has limited capabilities and is satisfactory only for long duration missions and high image quality requirements. In [4] we suggested two other fault-tolerance techniques, *HC - Hardware Correction*, and *SC - Software Correction*. *HC* is based on splitting the photodiode in each pixel into two half-size photodiodes and duplicating the readout transistors so that one photodiode can be used even if the second one is defective. After testing the entire pixel array under full dark and full light images, the pixels with half intensity can be identified and their value can then be multiplied by two. Relying on the fact that in digital images a pixel value is correlated to that of its immediate neighbors, *SC* uses several weighted averages of the faulty pixel's neighbors to estimate its value. In [4] we analyzed the effects of both methods on pixel errors for grayscale images, in which each pixel is represented by one 8-bit integer between 0 and 255. We demonstrated that in

* This work was supported in part by JPL, under contract 961294, by NSF, under contract MIP-9710130, by the NSERC Canada, and by the Canadian Microelectronics Corp.

most cases the combination of the hardware and software based methods results in a considerably higher image quality than that of the software based technique alone.

In this paper, we focus on fault-tolerance methods for color images using the RGB system in which each pixel is represented by three 8-bit integers, describing the intensities of the Red, Green, and Blue components of the color pixel. The failure of a single color within any color pixel will cause a significant color shift in the image which can be very noticeable (for example, shifting from a color with mixed R and B to one with only B values). This paper describes methods whereby if only one color of a pixel is faulty, the remaining components can be used to improve the estimate.

2: The model

2.1: Fault model and fault-tolerance methods

The hardware correction for the color camera would be a modification of that proposed previously in [2, 4]. In the case of color cameras, typically each pixel consists of Red (R), Green (G) and Blue (B) subpixels. Note that in many digital cameras, often two Green subpixels are used to form a square of $RGBG$ and thus keep the color pixel uniform in size. For HC we again assume a $CMOS$ Active Pixel Sensor for each color subpixel, and split diodes and readout transistors [4]. Each half of the APS diode generates current that charges the gate of the readout transistors. When the row select transistor is activated, current flows through the readouts to the column output in proportion to the charge gathered on the gate. The diodes are then reset to a precharged level after each image.

Our fault model assumes that the pixel array consists of $N \times N$ pixels, and each pixel consists of three subpixels (representing the R, G, B colors) which are statistically independent with regard to faults. Faults occur at each subpixel according to a Poisson process with a rate of λ faults per unit time. Hence, the probability of a given pixel to be fault-free at time t is $e^{-3\lambda t}$ and the probability of the whole array to be fault-free at time t is $e^{-3N^2\lambda t}$. As shown in [4], this probability deteriorates very fast, even for very small failure rates λ, due to the large value of N^2.

Since the probability of both halves of a subpixel being faulty at the same time is very small, most faults will be corrected using HC. If any subpixel remains faulty after applying HC, we can assume that all 26 subpixels surrounding it are defect-free and can be used for software correction. The statistical analysis of the HC for color images is very similar to that appearing in [4] for grayscale images and will not be repeated here. The next subsection deals with the analysis of SC, which is different for color cameras due to the existence of three subpixels, not all of which fail simultaneously.

2.2: The software Correction Methods

In black and white cameras, the only choice for software correction is to do interpolations based on the luminosity (or light intensity) of the nearest pixels. In color images, on the other hand, each pixel carries both the luminosity (L=R+G+B) and the color information, i.e., the relative ratio between the colors. It is clear that in most real life scenes the luminosity of an image changes much faster from pixel to pixel than does the color itself. For example, a typical image will have objects which have a slowly changing or near uniform color, but the luminosity or light level across the object may very quite quickly. Certainly there will be boundaries between objects, and thus sharp color boundaries, but these will be much less common than the luminosity changes. This is the principal that a color TV works on: most of the broadcast signal is related to the luminosity (i.e., the B&W image) while the color signal takes up a much smaller portion of the TV spectrum.

Consider the case of a single failed color (Red in what follows) of an RGB pixel. The important point is that for almost all cameras the RGB optical filters (integrated into the detectors) are such that any color that generates a strong signal of one value (e.g., R) will also generate some additional values of the other colors (G and B) for even pure "Red" values. Indeed, many colors have significant

mixtures of all three values. Hence, if the color value (such as the ratio R to G to B), and the signal from the remaining colors (G and B in this case) are known then the R value can be calculated with high accuracy. What is important is to use a method that separates out the luminosity from the color.

We consider seven forms of software correction (although many more are possible) and denote them by SC^k $(k = 1, 2, ..., 7)$. To define those we make the following notations. We denote the coordinates of the faulty pixel by (0,0), and those of its eight neighbors by (0,1), (1,1), (1,0), (1,-1), (0,-1), (-1,-1), (-1,0), (-1,1). We also denote by $R_{i,j}$, $G_{i,j}$ and $B_{i,j}$ the values (each between 0 and 255) of the red, green, and blue subpixels, respectively, of the pixel whose coordinates are (i, j) $(i, j = -1, 0, 1)$. We assume that $R_{0,0}$ is the only faulty subpixel among all 27 subpixels in the neighborhood. We then denote by $\hat{R}^k_{00}$ the estimated value of the faulty subpixel as obtained by SC^k $(k = 1, 2, ..., 7)$.

The first three methods use only the R values of the neighbors. These are similar to the best interpolations that we could use in the case of $B\&W$ images. In the first method, SC^1, the value of the faulty R is estimated by the average of the R values of its eight immediate neighbors, i.e.,

$$\hat{R}^1_{00} = \frac{1}{8}(R_{01} + R_{11} + R_{10} + R_{1,-1} + R_{0,-1} + R_{-1,-1} + R_{-1,0} + R_{-1,1}) \tag{1}$$

The second method assumes that the faulty R and the R values of its eight neighbors obey a quadratic function

$$R_{x,y} = a_{00} + a_{10}x + a_{01}y + a_{11}xy + a_{20}x^2 + a_{02}y^2 + a_{21}x^2y + a_{12}xy^2 \tag{2}$$

Substituting the given R_{ij}'s for $(i, j) \neq (0, 0)$ we have eight linear equations in the eight unknown coefficients a_{kl}. Since $R_{00} = a_{00}$, we need to solve only for a_{00}, which results in

$$\hat{R}^2_{00} = \frac{1}{2}(R_{01} + R_{10} + R_{0,-1} + R_{-1,0}) - \frac{1}{4}(R_{11} + R_{1,-1} + R_{-1,-1} + R_{-1,1}) \tag{3}$$

The third method attempts to use more information than is included in the eight immediate neighbors, and calculates the average of the R values of the 24 neighbors surrounding the faulty pixel:

$$\hat{R}^3_{00} = \frac{1}{24} \sum_{\substack{-2 \leq i,j \leq 2 \\ i,j \neq 0,0}} R_{ij} \tag{4}$$

The first three methods did not make any use of the relationships between R and the other two colors. Now consider the interpolation methods that use all the available color information. The next two methods look at the ratio between the missing R value and the pixel luminosity $L = R+G+B$, assuming that the ratio $C_{i,j} = R_{i,j}/L_{i,j}$ changes very little over small distances.

The fourth method calculates $\hat{C}_{00}$ as the average of the $C_{i,j}$'s of its eight neighbors, and then estimates R_{00} by

$$\hat{R}^4_{00} = \frac{\hat{C}_{00}(G_{00} + B_{00})}{1 - \hat{C}_{00}} \tag{5}$$

The fifth method estimates C_{00} assuming a quadratic relationship among the nine $C_{i,j}$'s in the neighborhood, and thus, similarly to (3),

$$\hat{C}_{00} = \frac{1}{2}(C_{01} + C_{10} + C_{0,-1} + C_{-1,0}) - \frac{1}{4}(C_{11} + C_{1,-1} + C_{-1,-1} + C_{-1,1}) \tag{6}$$

and $\hat{R}^5_{00}$ is obtained from $\hat{C}_{00}$ using (5).

Rather than using the ratio C, a better method is to make use of actual color measurements which will be employed in the last two methods. To this end, a color coordinate system is required where a color can be specified by some numerical parameters. Many such systems exist, but for the purposes of this work we will use the CIE (Commission Internationale de l' Eclairage) color standard system which has the important characteristic of giving the color in an x,y system, and the luminosity separate from the color [1, 3]. The CIE 1931 system starts by taking the RGB from the camera and relating these to Tristimulus values X, Y and Z which contain both color and luminosity information. The transformation from (R, G, B) to (X, Y, Z) is done through a conversion matrix T,

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = T \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{7}$$

The entries in the matrix T are set by the optical characteristics of the color filters in the particular camera being used. For the following examples we will use the standard case which sets Red at 700nm filter, Green at 546.1nm and Blue at 435.8nm [3], resulting in

$$T = \begin{pmatrix} T_{rx} & T_{gx} & T_{bx} \\ T_{ry} & T_{gy} & T_{by} \\ T_{rz} & T_{gz} & T_{bz} \end{pmatrix} = \begin{pmatrix} 0.7347 & 0.2653 & 0.0000 \\ 0.2738 & 0.7174 & 0.0088 \\ 0.1666 & 0.0089 & 0.8245 \end{pmatrix} \tag{8}$$

In the true CIE color coordinates, these Tristimulus values are further reduced to a dimensionless Chromaticity coordinate system x,y and z that does not include the luminosity. However, our tests showed that better interpolations were obtained with the Tristimulus values X, Y and Z, so they were used in this work.

Assume that for the $(0, 0)$ pixel, G and B are available while X, Y and Z can be obtained by interpolating the X, Y and Z values of the neighbors. Dividing each line in (7) by its respective interpolated Tristimulus value creates a linear transformation for X and Y based on (8) and (7) which yields

$$\frac{T_{rx}R + T_{gx}G + T_{bx}B}{X} = \frac{T_{ry}R + T_{gy}G + T_{by}B}{Y}$$

and thus,

$$R_{ij} = \frac{G_{ij}\left(\frac{T_{gy}}{Y_{ij}} - \frac{T_{gx}}{X_{ij}}\right) + B_{ij}\left(\frac{T_{by}}{Y_{ij}} - \frac{T_{bx}}{X_{ij}}\right)}{\frac{T_{rx}}{X_{ij}} - \frac{T_{ry}}{Y_{ij}}} \tag{9}$$

In the sixth method, X_{00} and Y_{00} are estimated as simple averages of X_{ij} and Y_{ij}, respectively, of the eight immediate neighbors, and then substituted in (9) to obtain $\hat{R}^6_{00}$.

In the seventh and last method, we assume that X, Y and Z are quadratic functions of the pixel coordinates i and j, and thus, similarly to (3),

$$\hat{X}_{00} = \frac{1}{2}\left(X_{01} + X_{10} + X_{0,-1} + X_{-1,0}\right) - \frac{1}{4}\left(X_{11} + X_{1,-1} + X_{-1,-1} + X_{-1,1}\right) \tag{10}$$

$\hat{Y}_{00}$ is obtained similarly, and then substituted in (9) to get $\hat{R}^7_{00}$.

Clearly, similar equations to (9) can be obtained for the B and G colors. One important point to note is that for cameras with the $RGBG$ arrangement, the Green color has redundancy within a color pixel. Thus, for the raw data, a software correction of G similar to that discussed in [4] for the split pixel hardware correction is probably a better solution.

2.3: Performance measures

We used five different measures to compare the goodness of fit of the seven SC methods $\hat{R}^1, ..., \hat{R}^7$ described above to a given color image. In each case, we analyzed the whole image and calculated the absolute errors incurred when using the estimate rather than the actual value of R. Denoting by i, j the pixel coordinates, $1 \leq i, j \leq N$ (where $N \times N$ is the size of the pixel array), we denote, for a specific SC method $\hat{R}^k$,

$$E_{ij}^k = |R_{ij} - \hat{R}_{ij}^k|$$

The first measure, M_1, calculates the average of E_{ij}^k over the whole image:

$$M_1 = \frac{1}{N^2} \sum_{1 \leq i,j \leq N} E_{ij}^k$$

The second measure, M_2, calculates the average relative error:

$$M_2 = \frac{1}{N^2} \sum_{1 \leq i,j \leq N} \frac{E_{ij}^k}{R_{ij}}$$

M_3 calculates the average error, relative not to the R value but to the total luminosity of the pixel:

$$M_3 = \frac{1}{N^2} \sum_{1 \leq i,j \leq N} \frac{E_{ij}^k}{R_{ij} + G_{ij} + B_{ij}}$$

M_4 and M_5 are based on our observation that the same absolute difference in the value of R is more noticeable to the eye for lighter pixels than for darker pixels, and the measure should reflect this. Since lighter pixels are characterized by higher values, we divide the error by the complement of the pixel value or the luminosity value, i.e.,

$$M_4 = \frac{1}{N^2} \sum_{1 \leq i,j \leq N} \frac{E_{ij}^k}{255 - R_{ij}}$$

and

$$M_5 = \frac{1}{N^2} \sum_{1 \leq i,j \leq N} \frac{E_{ij}^k}{765 - (R_{ij} + G_{ij} + B_{ij})}$$

Clearly, if all faulty pixels are corrected perfectly, each M_i will be equal to zero.

We analyzed many digital color images in which we calculated the five measures for the seven SC methods. In almost all cases, M_1, M_4 and M_5 were consistent in the way the methods were ordered from best to worst. As demonstrated in the next section, using M_2 and M_3 resulted in a different ordering among the methods, sometimes almost reversed to that generated by the other three. This leads us to believe that either of the three measures - M_1, M_4 or M_5 can be used when selecting an SC method. Some of our numerical results are presented in the next section.

3: Numerical Results

To demonstrate the differences among the different correction methods we experimented with a large number of digital color images, including landscapes, groups of people, and pictures of earth taken from space by NASA.

We based the ordering of the methods on the measure M_1 - the average absolute error over the whole image, although M_4 and M_5 resulted in the same ordering.

Some general statements which apply to almost all the images we have analyzed are:
(a) Methods which include all three colors' information are better than those that use only the red subpixels.
(b) Using X, Y and Z is better than using the ratios R/L, G/L and B/L.
(c) Using only eight immediate neighbors is better than using 24 neighbors, which may be too far away from the faulty pixel to provide any useful information.
(d) Using interpolations which assume quadratic functions are better than simple averages.

Figure 1 shows the analysis results of an image of the Golden Gate bridge. Method $\hat{R}^7$ which fits a quadratic function to X, Y and Z is the best with regard to the measures M_1, M_4 and M_5, while $\hat{R}^3$ is the worst. Figure 2 shows the frequency distribution of the error size for three out of

	M_1	M_2	M_3	M_4	M_5
$\hat{R}^7$	0.8795	0.7677	0.1078	0.0050	0.0021
$\hat{R}^5$	0.9835	0.6684	0.0914	0.0056	0.0023
$\hat{R}^2$	1.2177	0.2460	0.0344	0.0073	0.0030
$\hat{R}^6$	1.4789	0.7310	0.1015	0.0085	0.0035
$\hat{R}^4$	1.7026	0.6342	0.0851	0.0100	0.0041
$\hat{R}^1$	3.0886	0.1944	0.0256	0.0198	0.0078
$\hat{R}^3$	4.6422	0.5744	0.0753	0.0300	0.0119

Figure 1. Golden Gate Bridge - Analysis results.

the seven methods, and its conclusions are similar to those obtained from Figure 1.

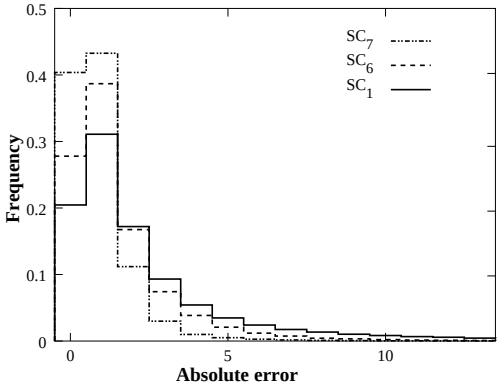


Figure 2. Distribution of errors for three correction schemes – Golden Gate Bridge image.

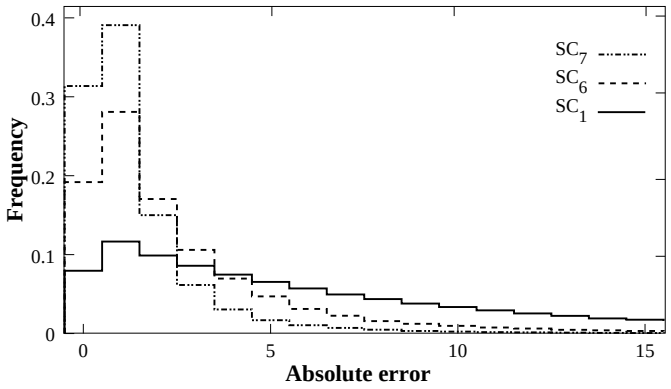


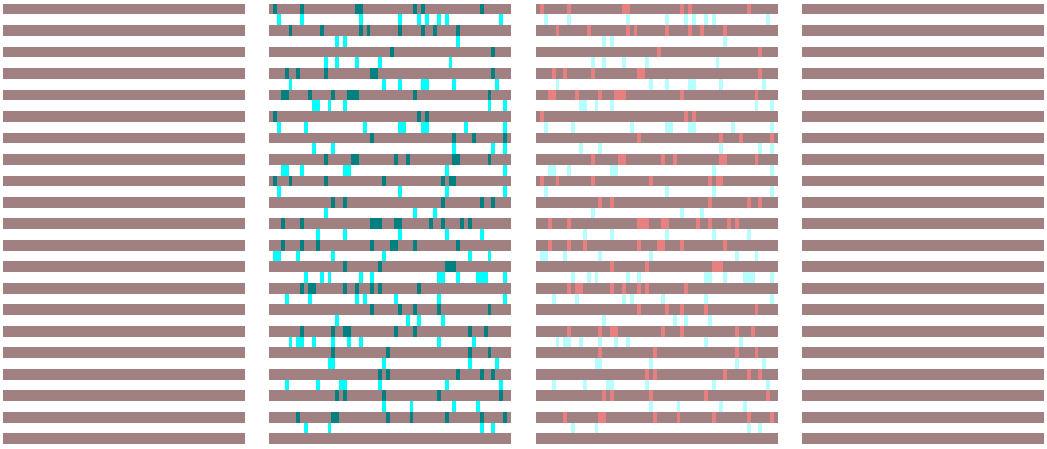
Figure 3. Distribution of errors for three correction schemes – Flowers image.

As a comparison, similar results are presented in Figures 3 and 4 for the shown picture of flowers. Although the order of methods is not exactly the same as for the first picture, the best two and the worst two are the same.

To demonstrate the visual effects of subpixel errors and of the different SC methods we used an image of horizontal bars in which we inserted 10% random defects in the red subpixel and then corrected them using either $\hat{R}^1$ or $\hat{R}^7$. The four different images appear in Figure 5 (in grayscale). As can be seen, $\hat{R}^1$ which uses only the red subpixels of the neighbors does not get rid of all the errors, while $\hat{R}^7$ which uses all three colors results in an image which is visually identically to the original.

	M_1	M_2	M_3	M_4	M_5
$\hat{R}^7$	1.0923	0.4090	0.1706	0.0303	0.0046
$\hat{R}^5$	1.8076	0.3711	0.1535	0.0376	0.0063
$\hat{R}^6$	2.0570	0.3965	0.1649	0.0415	0.0074
$\hat{R}^4$	3.2570	0.3505	0.1442	0.0515	0.0107
$\hat{R}^2$	4.4379	0.1479	0.0600	0.0929	0.0178
$\hat{R}^1$	10.5967	0.1033	0.0419	0.3265	0.0502
$\hat{R}^3$	16.8580	0.3186	0.1298	0.6140	0.0883

Figure 4. Flowers - Analysis results.



Original image **10% errors in R** **Corrected with $\hat{R}^1$** **Corrected with $\hat{R}^7$**

Figure 5. Horizontal bars.

In addition to the analysis of real pictures, simulations with common pixel color and intensity distributions have been done to study what are the hard to fit type patterns. Figure 6 shows plots of the R values in a 3×3 color pixel set for a very difficult to interpolate condition. In this figure, a color step crosses the x=1 edge, going from an R:G:B ratio of (0.4,1.0,0.9) for X=0,-1 to a ratio of (1.0,1.0,0.5). The luminosity also changes, with R = 102 (x=1), R=25 (x=0,-1) but with a sudden spike of R=255 at the center (0,0) point where the R subpixel is dead. The $\hat{R}^1$ interpolation (traditional averaging) gives a very poor result (R=54) while the CIE-based $\hat{R}^6$ yields R=171 and $\hat{R}^7$ returns the true value of 255.

Hence, even with very sudden changes in both color and luminosity, $\hat{R}^7$ returns exactly the correct value. The only places where $\hat{R}^7$ appears to give significant deviations are when the color shifts rapidly or randomly across all the 3×3 pixels, combined with sudden luminosity changes. That is probably a very rare condition in actual images.

4: Conclusion

We have investigated in this paper seven software correction algorithms that interpolate a faulty color subpixel based on its nearest neighbors. We concluded that correction schemes which use a quadratic interpolation formula based on the CIE color coordinates of the Tristimulus values (X, Y, Z) yielded the best results. These fault correction schemes, based on using the color informa-

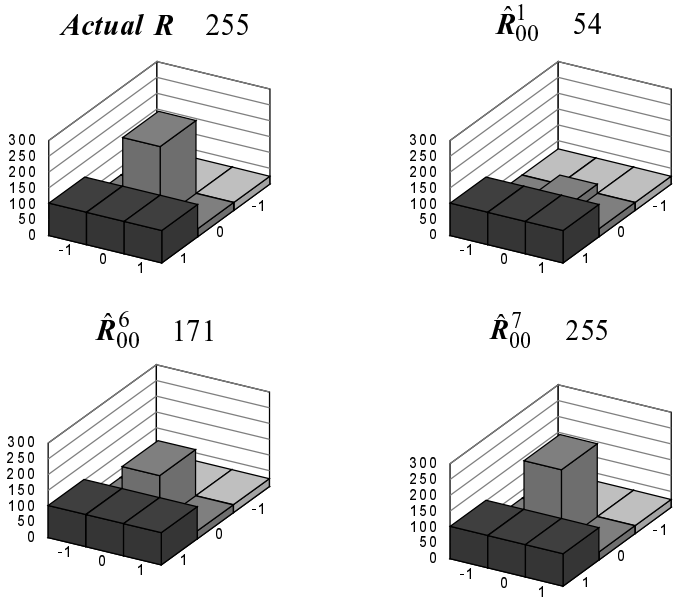


Figure 6. Correcting the R value in a 3×3 pixel set.

tion, provided significantly better correction than those based on just the interpolating the missing simgle color data.

References

[1] F.W. Billmeyer, Jr. and M. Saltzman, *Principles of Color Technology,* Second Edition, John Wiley & Sons, New York 1981.

[2] G. Chapman and Y. Audet, "Creating 35 mm Camera Active Pixel Sensors," *Proc. of the 1999 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, pp. 22-30, November 1999.

[3] F.W. Clulow, *Colour: Its Principles and Their Applications,* Morgan and Morgan, New York, 1972.

[4] I. Koren, G. Chapman, and Z. Koren, "A Self-Correcting Active Pixel Camera," *Proc. of the 2000 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, November 2000.

[5] I. Koren and Z. Koren, "Incorporating Fault-Tolerance into a Digital Camera-On-A-Chip," *Proc. of the 1999 Microelectronics Reliability and Qualification Workshop*, pp. 1-3, Pasadena, Oct. 1999.

[6] B. Pain *et al.*, "A Low-Power Digital Camera-on-a-Chip Implemented in CMOS Active Pixel Approach," *Proc. of the 12th VLSI Design Conference*, pp. 1-6, January 1999.

[7] S-Y. Ma and L-G. Chen, "A single-Chip CMOS APS Camera with Direct Frame Difference Output," *IEEE Journal of Solid-state Circuits,* Vol. 34, Oct. 1999, pp. 1415-1418.

[8] S-L. Liu, P. Chen, C-H. Chen and J-G. Hwu, "Analog Maximum, Median and Minimum Circuit," *Proc. of the 1997 IEEE Intern. Sympo. on Circuits and Systems,* Vol. 1, pp. 257-260, June 1997.

[9] M. Sasaki, T. Inuoe, Y. Shirai and F. Uneo, "Fuzzy Multiple-Input Maximum and Minimum Circuits in Current Mode and Their Analysis Using Bounded-Difference Equations," *IEEE Trans. on Computers*, Vol. 39, pp. 768-774, June 1990.