IMPULSE NOISE REMOVAL FROM IMAGES GENERATED BY A SINGLE CMOS SENSOR WITH A COLOR FILTER ARRAY

Frank Mertz, Michael S. Moore and Sanjit K. Mitra* [†]

Department of Electrical & Computer Engineering
University of California
Santa Barbara, CA 93106-9560
mertz@ind.rwth-aachen.de, {msmoore,mitra}@ece.ucsb.edu

ABSTRACT

CMOS sensors of digital color cameras sometimes produce “bad pixels” that appear as positive impulse noise in captured images. Because the occurrence and the magnitudes of the impulses vary over time, an algorithm is needed to deal with these defects when and where they occur. This paper describes the demands on the noise removal method, details various algorithms that were implemented to address this problem, and finally provides some performance results. The most effective algorithms consist of separate routines for impulse detection and replacement, i.e. the approximation of detected defects. In addition to implementations of some common methods, modified to work with the color filter array, a new method was developed for this application. The proposed algorithm adapts the detection sensitivity to the current detail level of the image.

1. INTRODUCTION

Digital color cameras using a single sensor array system cover individual pixels in the sensor with red, green, or blue optical filters. The arrangement of color filters is usually a mosaic pattern. The most common color filter array (CFA) is the Bayer CFA pattern as shown in Figure 1 [1]. There are twice as many green filters as red or blue, because the green component signals contribute more to the luminance signal than the red or blue component signals.

The output of the single array system is separated into three different arrays. The full color image is generated by an appropriate color interpolation method, which fills in the missing pixels in each of the three arrays. However, the CMOS sensor chips sometimes contain a number of “bad pixels” which result from the production process and are often unavoidable. These pixels generate exclusively positive impulse noise of random height in the acquired image. It is thus necessary to detect and mask the impulses using image-processing algorithms before a color interpolation algorithm is applied to generate the full color image.

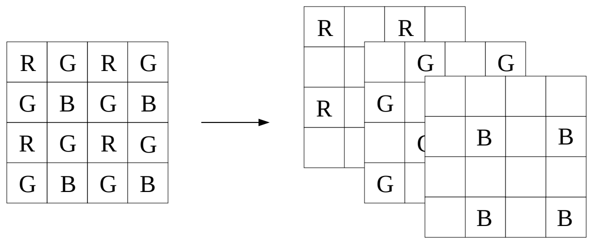


Figure 1: The Bayer pattern CFA must be separated into three color-planes and the missing colors interpolated to create a full color image.

The positions of the corrupted pixels in an image generated by a defective sensor are fixed. For this reason, the impulse pattern is sometimes referred to as fixed pattern noise. However, the impulse intensities depend on several factors. For example, the intensity of a bad pixel is temperature-dependent. Corrupted pixels that are not initially visible become visible as the sensor heats up. In addition, the corrupted pixel intensity levels depend on the base level of illumination. As illumination increases, the magnitude of an impulse generally decreases.

These facts make it impractical to measure and store the impulse locations of each chip during a test phase and then use this data later for correction. The defects may be spatially static, but they are temporally dynamic. Therefore, the impulse noise problem requires an algorithm that is able to deal with corrupted pixels as they occur.

[*] Now with the Institute of Communication Systems and Data Processing (IND), Aachen University of Technology (RWTH), Germany.

[†] This work was supported in part by a University of California MICRO grant with matching support from Lucent Technologies, National Semiconductor Corporation, Tektronix Corporation, and Xerox Corporation.

2. IMPULSE NOISE REMOVAL ALGORITHMS

There are many existing algorithms for removing impulse noise from images [2, 3, 4]. In general, the algorithms were designed assuming a type of impulse noise, such as salt-and-pepper noise [2], uniformly distributed impulse noise [3], or impulses resulting from flipped bits in a binary transmission channel [4]. However, the impulses resulting from CMOS sensor defects do not conform to any of these models. An algorithm to remove fixed pattern noise should:

1. Remove only positive impulses. The CMOS defects do not introduce negative impulses.
2. Keep track of the color of the pixel being filtered. The color components of an image are not always correlated. Therefore, the local neighborhood of a pixel should only include pixels that are the same color.
3. Be small and fast to support video applications.

The following four algorithms have been implemented and tested on images generated by a commercial CMOS sensor:

1. Median Filter [2],
2. MaxiMin Filter [2],
3. SD-ROM Filter [3], and
4. Adaptive SD-ROM Filter.

The first three algorithms have been modified to take into account the characteristics of the color filter array. The fourth algorithm is a new algorithm and was developed exclusively for this problem.

The following subsections describe the algorithms mentioned above in detail. The pixel considered for evaluation and replacement if necessary is referred by $x(\mathbf{n})$, with $\mathbf{n} = (n_1, n_2)$ the coordinate of the pixel's position within the image. A set of N neighboring pixels of $x(\mathbf{n})$ that have the same color is given by $\mathbf{r}(\mathbf{n}) = [r_1(\mathbf{n}), r_2(\mathbf{n}), \ldots, r_N(\mathbf{n})]$ and defined as shown in Figure 2.

A special treatment of the green pixels, having additional closer neighbors in the diagonal directions, generally leads to slightly better results. However, the improvements proved to be quite small. Therefore the green pixels are treated like red and blue to make the algorithms less complex. For the same reason, the algorithms generally work with only four of the eight neighbors.

2.1. Median Filter

A common algorithm for reducing impulse noise in an image is the median filter [2]. This filter replaces the center pixel x by the median of the pixel set containing x and the neighbors r_1,r_2,r_3 and r_4. The median of a set of pixels is defined as the middle element of the sorted version of this set.

r_5		r_1		r_6
r_2		x		r_3
r_7		r_4		r_8

Figure 2: Considered pixel x and it's neighbors of the same color

This algorithm does not attempt to classify x as either corrupted or uncorrupted. Rather, every pixel in the image is altered, removing impulse noise very effectively but causing a strong blurring effect. The resulting blur is generally unacceptable in this application.

2.2. MaxiMin (Maximum of Minima) Filter

This algorithm is described in [2] and was modified for this special application (Bayer pattern and only positive impulse noise). The current pixel x is compared to its neighbors r_1,r_2,r_3 and r_4. The pixel is replaced by the biggest of its neighbors, only if it is the largest in its neighborhood. This is accomplished by first determining the minima:

$$m_1 = \min(x, r_1) \qquad m_3 = \min(x, r_3)$$
$$m_2 = \min(x, r_2) \qquad m_4 = \min(x, r_4).$$

From these minima, the maximum value

$$m = \max(m_1, m_2, m_3, m_4),$$

is chosen to replace the center pixel x.

The algorithm works very effectively, causing no visible blurring and making few visible mistakes. However, in some cases clearly visible impulses were missed by this algorithm. In addition this filter is not a robust filter. It can not handle more than one bad pixel within the sample window.

2.3. SD-ROM Filter

Originally the SD-ROM algorithm (signal dependent rank-ordered mean) [3] was designed to remove impulses in highly corrupted images, e.g. with up to 40% corruption. In this application, the corruption rate is very much smaller, only around 1%. This small rate makes it possible to get very good results in finding the corrupted pixels, but it also makes false detection errors (replacing non-corrupted pixels) more important.

The SD-ROM algorithm is a decision-based algorithm. First, it classifies the center pixel x as either an impulse or an uncorrupted value. Only the pixels that are classified as impulses are replaced by an estimated value.

To decide whether the current pixel x is an impulse, the group of its N neighbors $\mathbf{r} = [r_1, r_2, \ldots, r_N]$ is first sorted in descending order, so that $r'_1 \geq r'_2 \geq \ldots \geq r'_N$. Assuming $N = 4$, the center pixel x is then compared to the two biggest neighbors r'_1 and r'_2. If at least one of the conditions $x - r'_1 > t_1$ and $x - r'_2 > t_2$ is true, x is considered an impulse. From experiments, the best values for the thresholds were computed as $t_1 = 12,\ t_2 = 36$.

If the algorithm decides the current pixel x is corrupted, the pixel is replaced by an estimate. The filter uses the rank-ordered mean (ROM) m, computed from the sorted field of its neighbors. The rank-ordered mean is the middle element of $\mathbf{r}$, or if the number of neighbors N is even, it is the average of the two middle elements:

$$m = \begin{cases} \dfrac{r_{\frac{N}{2}} + r_{\frac{N}{2}+1}}{2} & \text{if N even,} \\ r_{\frac{N+1}{2}} & \text{if N odd.} \end{cases}$$

For $N = 4$, $m = \frac{r'_2 + r'_3}{2}$.

In all cases, a recursive version that replaces x immediately provided nearly as good results as a non-recursive version. For reasons of simplicity and memory conservation the recursive form should be used. The results presented in Section 3 justify this decision.

A larger value of N (e.g. $N = 8$) increases the complexity of the algorithm and does not provide significantly better results.

2.4. Adaptive SD-ROM Filter

The SD-ROM algorithm described in Section 2.3 sometimes incorrectly detects bad pixels, especially in areas with high detail and changing colors like written text. Making the threshold t_1 adaptive and dependent on the current detail level of the image has shown a clear improvement.

By increasing t_1 in high detail areas, the algorithm becomes less sensitive to changes, thereby reducing the number of mistakenly replaced pixels in these regions. A small increase in the number of missed impulses can be tolerated in a highly detailed area where corrupted pixels are less likely to be seen. In an image area that has an almost constant color, t_1 should be reduced, allowing the impulse detector to find even smaller impulses than the regular SD-ROM algorithm.

The adaptation adjusts the threshold t_1 to the average distances between pixels in the current area of the image. Fig. 3 displays the principle of this approach. The impulse detection works exactly like in the regular SD-ROM algorithm, using the four neighbors $r_1, \ldots, r_4$. After evaluating and replacing x where necessary, a new threshold t_1 is calculated for the next step. The algorithm must store the last three absolute differences between the upper and left neigh-

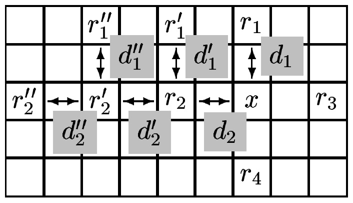


Figure 3: Adaptation of threshold value t_1

bors and their center pixel in a set

$$\mathbf{d} = (d''_1, d''_2, d'_1, d'_2, d_1, d_2)\ ,$$

with its elements calculated as follows:

$$\begin{aligned} d''_1 &= \mid r''_1 - r'_2 \mid & \qquad d''_2 &= \mid r''_2 - r'_2 \mid \\ d'_1 &= \mid r'_1 - r_2 \mid & d'_2 &= \mid r'_2 - r_2 \mid \\ d_1 &= \mid r_1 - x \mid & d_2 &= \mid r_2 - x \mid \end{aligned}$$

The next threshold is then computed as

$$t_1 = t_0 + \frac{d''_1 + d''_2 + d'_1 + d'_2 + d_1 + d_2}{6}\ ,$$

with t_0 set to a constant minimum threshold value. From experiments, $t_0 = 9$ seems to be optimal.

This algorithm works recursively to assure that corrupted pixels are excluded from the adaptation process. That is also the reason for choosing only the upper and left neighbors for the computation of t_1, because they have already been evaluated and replaced when necessary.

3. EXPERIMENTAL RESULTS

The results obtained with the algorithms are given in this section, but first the setup that was used to measure the performances is explained in detail.

3.1. Evaluation Setup

To evaluate the performance of the filters, several still images were captured using commercial CMOS sensors. Several sensors were available and ranged in the number of impulses from a few (~ 20) to many (> 2000). The content of the still images was also varied to include both relatively simple scenes and very detailed scenes.

Immediately after each image was captured, another image was captured with a lens cap covering the camera lens. Covering the lens will result in an absolutely black image if the sensor has no bad pixels. However, bad pixels cause positive impulses that appear as white dots in the black frame, similar to stars in a night sky. We called these images constellation images. The constellation images were

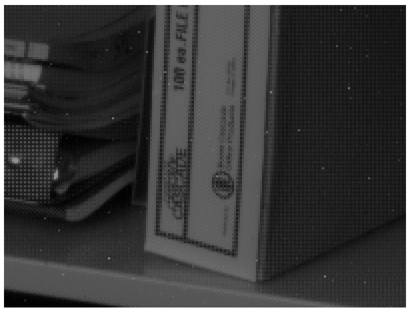
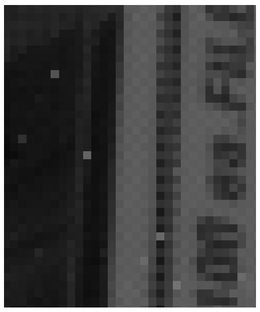
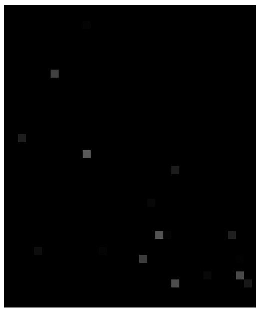
(a)  (b)  (c) 

Figure 4: CMOS sensor output: (a) test, (b) zoomed, and (c) constellation image

captured immediately after the test images to minimize sensor temperature effects. Figure 4 contains zoomed portions of both a test image and the matching constellation image.

The constellation images were used to evaluate the performance of the filtering algorithms in two ways. First, the impulse locations were found by thresholding the constellation image. The pixels in the filtered images could then be separated into four categories: impulses that were correctly removed, image pixels that were correctly classified as non-impulses, impulses that were missed and not removed, and image pixels that were incorrectly classified as impulses and replaced. A good filter removes impulses without changing uncorrupted pixels. Second, the constellation image was used to 'fix' the test images. Essentially, only the impulses detected in the constellation were replaced. The resulting image approximates an ideal filter output. To measure the performance of the various filters, the peak signal-to-noise ratio (PSNR) was calculated on the differences between the filter outputs and the ideal output. The PSNR is defined by

$$PSNR = 10\log_{10}\left(\frac{\sum_{l=1}^{L}\sum_{c=1}^{C}255^2}{\sum_{l=1}^{L}\sum_{c=1}^{C}[I(l,c)-F(l,c)]^2}\right)$$

for images with 8 bits per pixel, where $I(l,c)$ are the pixels of the ideal filter output and $F(l,c)$ is the filtered corrupted image, respectively. l and c are the lines and columns of the image with dimension $(L \times C)$.

3.2. Algorithm Performances

Table 1 summarizes some of the experimental results. The filter performances range over a fairly wide spectrum. The median filter removes the most impulses but also makes the most mistakes due to the non-classification of pixels (blurring effect). The other filters make fewer mistakes, but at the cost of fewer detected impulses. In general, a careful balance must be struck between removing impulses, even those with low intensities, and preserving the image.

The MaxiMin and SD-ROM algorithms performed very well according to this balance. The newly developed threshold adaptation method for the SD-ROM algorithm shows a significant increase in detected impulses and even a small decrease in mistakenly replaced uncorrupted pixels, when comparing the recursive implementations. Example images are shown in Figure 5.

Algorithm	recursive impl.	perc. of imp.found	errors	PSNR [dB]
Median	no	92.1 %	140940	30.1
	yes	91.8 %	131018	28.7
MaxiMin	no	77.9 %	21685	38.4
	yes	78.5 %	21754	38.5
SD-ROM	no	29.8 %	273	36.2
	yes	30.1 %	307	35.8
Adaptive SD-ROM	yes	46.5 %	277	36.6

Table 1: Performance results of the algorithms (total number of impulses in test image: 2362)

4. CONCLUDING REMARKS

Although the removal of impulse noise is a well-studied problem, standard noise models do not fit the CMOS fixed pattern noise problem very well. This fact was apparent in the poor performance of the standard median filter compared to the three algorithms that were modified or developed for this specific problem. Although the results do not support the selection of a single best algorithm for removing bad pixels from the sensor output, two algorithms, the

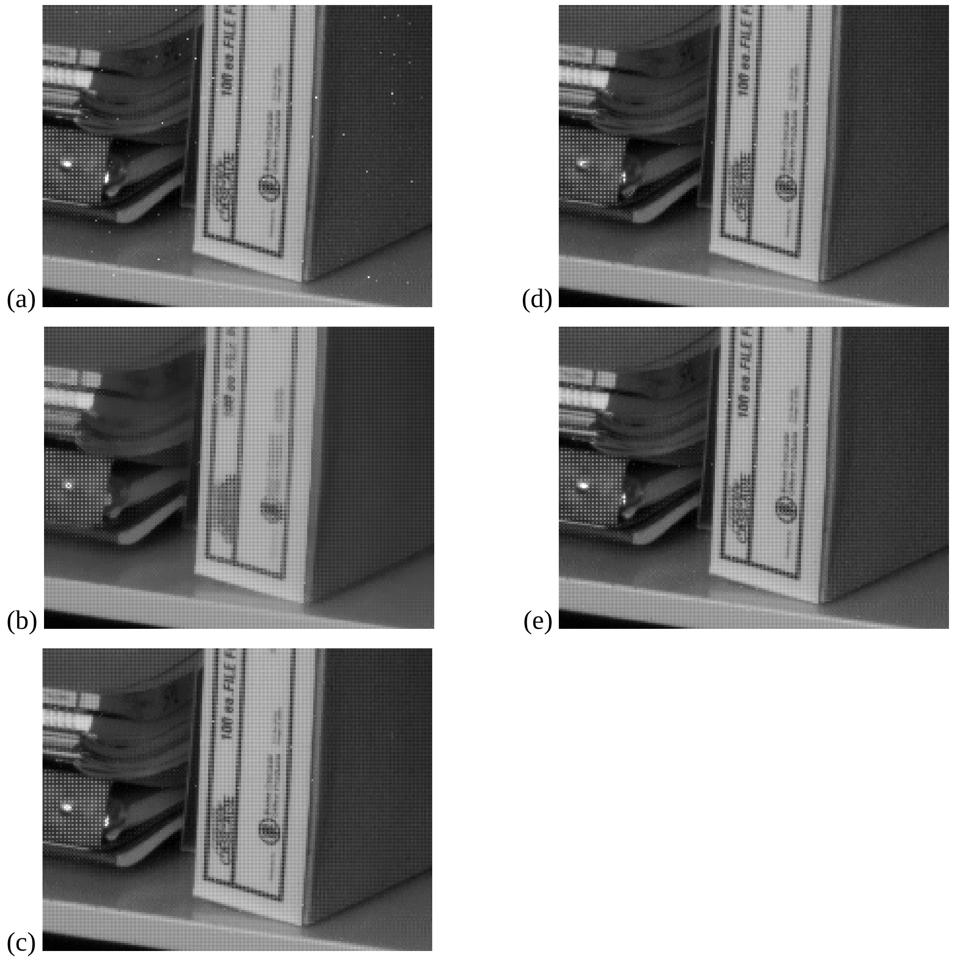


Figure 5: Performance shown on a test image: (a) corrupted image, (b) filtered with median algorithm, (c) filtered with MaxiMin algorithm, (d) filtered with SD-ROM algorithm, and (e) filtered with adaptive SD-ROM algorithm.

MaxiMin filter and the adaptive SD-ROM filter, performed very well. The PSNR results favor the MaxiMin filter, but the subjective quality of the resulting images are essentially the same. A decision between these two filters depends upon a trade-off between complexity and robustness. The MaxiMin algorithm is a simple filter that can remove even very small postive impulses. However, the algorithm can not correct clusters of bad pixels. If some additional computational complexity can be tolerated, the adaptive SD-ROM filter is a good alternative. The filter makes much fewer mistakes and can detect multiple impulses in close proximity.

5. REFERENCES

[1] B. Bayer. US Patent No. 3,971,065, (1976).

[2] W. E. Pratt, *Digital Image Processing*. Wiley-Interscience, New York, NY, 2nd ed., 1991.

[3] E. Abreu, M. Lightstone, S. K. Mitra, and K. Arakawa, “A new efficient approach for the removal of impulse noise from highly corrupted images,” *IEEE Trans. on Image Processing*, vol. 5, pp. 1012–1025, June 1996.

[4] J. Astola and P. Kuosmanen, *Fundamentals of Nonlinear Digital Filtering*. CRC Press, New York, NY, 1997.