Bad Pixel Location Algorithm for Cell Phone Cameras

Sergio Goma, Milivoje Aleksic
AMD, 75 Tiverton Court, Markham, ON, Canada, L3R-9S3

ABSTRACT

As CMOS imaging technology advances, sensor to sensor differences increase, creating an increasing need for individual, per sensor, calibration. Traditionally, the cell-phone market has a low tolerance for complex per unit calibration. This paper proposes an algorithm that eliminates the need for a complex test environment and does not require a manufacturing based calibration on a per phone basis. The algorithm locates "bad pixels", pixels with light response characteristics out of the mean range of the values specified by the manufacturer in terms of light response. It uses several images captured from a sensor without using a mechanical shutter or predefined scenes. The implementation that follows uses two blocks: a dynamic detection block (local area based) and a static correction block (location table based). The dynamic block fills the location table of the static block using clustering techniques. The result of the algorithm is a list of coordinates containing the location of the found 'bad pixels'. An example is given of how this method can be applied to several different cell-phone CMOS sensors.

Keywords: bad-pixel, white pixels, defective pixels, fixed-pattern noise, pixel response non-uniformity, cell-phone camera;

1. INTRODUCTION

One of the most important specifications of an image sensor is its "cosmetic quality": a sensor's image should be flawless. Unfortunately, the technology is not perfect. Due to processing imperfections, random variations, etc., a finite number of pixels in a sensor array will be defective (hard faults) or yield a signal that deviates visibly from the nominal toleration of a pixel gray value (soft faults). Such faults appear as white or black (or grey) points in the image. For a human observer, these tend to be much more annoying than other image imperfections such as temporal noise, mild fixed patterns, or imperfect reproduction of color or gray values. Usually these pixels are denominated as 'bad pixels' whether 'stuck-at' (hard-faults) or 'hot pixels' (soft-faults). The various methods described in the literature that address the problem can be classified into two main categories:

1. dynamic methods, which identify 'bad pixels' in a captured image using assumptions that the value of any pixels cannot deviate more than an estimated value of the neighbors.
2. static methods, which use a pre-defined 'location' table containing the location of 'bad pixels'.

The process of correcting for bad-pixels is typically a two-stage process consisting of:

stage 1: bad-pixel detection (or location), where a faulty pixel is identified using static or dynamic methods
stage 2: bad-pixel correction, where the faulty pixel is replaced with a value that is at least 'visually acceptable'[5];

There are some applications (eg. astronomy) that deal with issues that use a similar methodology, however in those cases, the pixels become 'hot-pixels' for only a limited number of frames (ie. cosmic ray induced errors). These applications are outside the main use cases for cell-phone camera sensors, but do offer sophisticated methods for bad-pixel location. A review of some of these methods is offered in[6]. A main assumption in these methods is that pixel defects do not cluster**.** . The algorithm presented in this paper locates clusters of bad-pixels, a case that presents difficulties for most detection algorithms but deals only with persistent 'bad pixels'.

Dynamic and static methods each have advantages and disadvantages. The main disadvantage of dynamic methods is the aliasing problem especially in the context of new CMOS pixels architectures that share transistors with adjacent pixels, which increases the occurrence of clustered faulty pixels.

Digital Photography III, edited by Russel A. Martin, Jeffrey M. DiCarlo, Nitin Sampat, Proc. of
SPIE-IS&T Electronic Imaging, SPIE Vol. 6502, 65020H, © 2007 SPIE-IS&T · 0277-786X/07/$18

One of the disadvantages of static methods is that the 'location table' needs to be filled prior to use. This can be done during sensor test procedures. While the quality of the sensor is evaluated, the bad-pixel location table could be programmed into a memory in the sensor. However, if the sensor does not have a memory to collect this information - and most of current CMOS sensors used in cell-phone camera do not - a calibration procedure needs to be performed by either the cell-phone integrator or the end-user. In either case, certain conditions need to be met for the bad pixel location data to be collected. Specifically, at least a 'black' image (an image obtained in absence of light) needs to be acquired. An automated procedure can be used if a mechanical shutter is available, however, in a cell-phone camera this is not usually the case, making this category of bad-pixel location methods hard to implement. The proposed algorithm offers a solution to this problem as it can be run at power-on and does not need specific conditions (ie. mechanical shutter or dark frame).

This paper is structured as follows:
Section 2, "Problem Definition" provides brief descriptions of fixed pattern noise and photo response non-uniformities. These are used to build fig. 1, which summarizes the pixel response. The algorithm is then developed to address the categories of faulty pixels as depicted in fig.1. Section 3, "Implementation", details some implementation options and Section 4 presents results.

2. PROBLEM DEFINITION

To describe the proposed algorithm, some definitions are reviewed below[4]. Those include fixed pattern noise (FPN) and photo response non-uniformity (PRNU). In case of FPN, for the purpose of this algorithm, only stuck-at faults are considered. For a more comprehensive discussion of CMOS FPN see[3]. This algorithm does not try to remove FPN, but to identify the 'visible' bad-pixels that are static and thus behave like stuck-at faults (under different conditions).

For pixel response characterization, the equivalent exposure is defined as the product of sensor gain and sensor integration time. Since the proposed algorithm is not particularly sensitive to random noise, equivalent exposure is used in order to characterize sensor response (i.e. as long as the equivalent exposure value is the same, any valid combination analog gain/exposure can be used with similar results). It is noted though, that sensor behavior is dependant on temperature as well as equivalent exposure. In some use scenarios, the algorithm can be triggered by a major change in temperature.

2.1. Fixed Pattern Noise (FPN)

FPN is a fixed pixel-to-pixel offset and is due to a combination of variations in the image sensor amplifiers and dark current. It is fixed with respect to the illumination; however, it is temperature, integration time, and gain dependent. The FPN is a measured as the pixel-to-pixel variation in absence of illumination, thus FPN includes only the non-uniformity evident in the absence of illumination. This may also be referred as Dark Signal Non-Uniformity (DSNU). The ideal sensor would produce the same output voltage for each pixel under dark conditions. In reality this is rarely the case, as there are several sources of non-uniformity. The contribution of FPN in the output signal is approximately constant under varying illumination and is usually specified as an absolute value. An extreme case of FPN are pixels that are stuck-at a value, and the most frequent are ones visible in a captured image as 'white' pixels. It should be noted that for stuck-at pixels, there is no dependency on exposure, gain, etc.

2.2. Photo Response Non-Uniformity (PRNU)

The PRNU is a measure of the pixel-to-pixel variation in output under uniform illumination. The ideal sensor would produce the same output signal for each pixel under uniform illumination. Again, this is rarely the case, as there are several sources of non-uniformity. The magnitude of PRNU in the output signal varies linearly with the energy of illumination. PRNU is specified as a percentage of the average output signal, which is representative of the operating level of exposure. Often the amount of non-uniformity that can be tolerated in image sensor applications is less for adjacent pixels than between two pixels farther apart in the array. The local neighborhood PRNU is specified as the greatest difference between the output voltage of any two active pixels within an 8 pixel neighborhood, under uniform illumination, expressed as a percentage of the average over all pixels in the array. In the case of an area sensor, the local neighborhood implies a square array of pixels. When explicitly specified in a datasheet, the maximum local neighborhood PRNU for a sensor will be significantly less than the maximum PRNU specified for the entire array. PRNU is specified as an RMS quantity that is measured statistically. $PPNU_{RMS}$ is based on the standard deviation of the histogram of the output signal, under uniform illumination, from all active pixels in the array. The measurement techniques should be such designed to remove temporal noise from the histogram to reveal the spatial characteristics of the PRNU. Any pixel with output deviating from the mean of the histogram by more than a specified threshold is classified as a blemish. For sensors with cosmetic defect grades, pixels identified as blemishes are excluded from the $PRNU_{RMS}$ measurement and the measurement is performed with illumination set between 50% and 80% of Saturation Equivalent Exposure[4]. Thus, PRNU, is a pixel-to-pixel variation in the sensor responsitivity. In relation to FPN, FPN is an offset, PRNU is a gain. It arises due to variations in pixels. For example, a sensor quoted as having a 3.5um by 3.5um pixel will actually have a variation in pixel size across the array. Measured at a given integration time, temperature, and gain under uniform lighting conditions (typically at half of the saturation level - 512DN in 10bit mode), this area variation will contribute to PRNU.

2.3. Pixel Defects Classification

Considering the above, given the structure of the pixel, several types of pixel defects can be easily distinguished (see Figure 1):

1. pixels stuck to a value (those defects are always there and their behavior is independent of the sensor temperature or equivalent exposure):
    a. “white” pixels which are stuck at 255, thus they’ll appear white even with no exposure;
    b. “black” pixels which are stuck at 0, thus they’ll appear black on any exposure (for the experiments conducted, no such pixels were detected), so this type of defects is not addressed by the presented algorithm;
2. pixels that have variations in response from the mean more than a specified value (those defects become more apparent when the exposure/temperature/analog gain increases):
    a. pixels that have different black offset, which are accounted for in the specified FPN of the sensor;
    b. pixels that have different light/dark current sensitivity and are accounted for in the specified PRNU of the sensor. Of these pixels, some tend to saturate sooner than the others, and, they become evident in a longer exposure or on a higher gain setting, appearing as “white pixels”

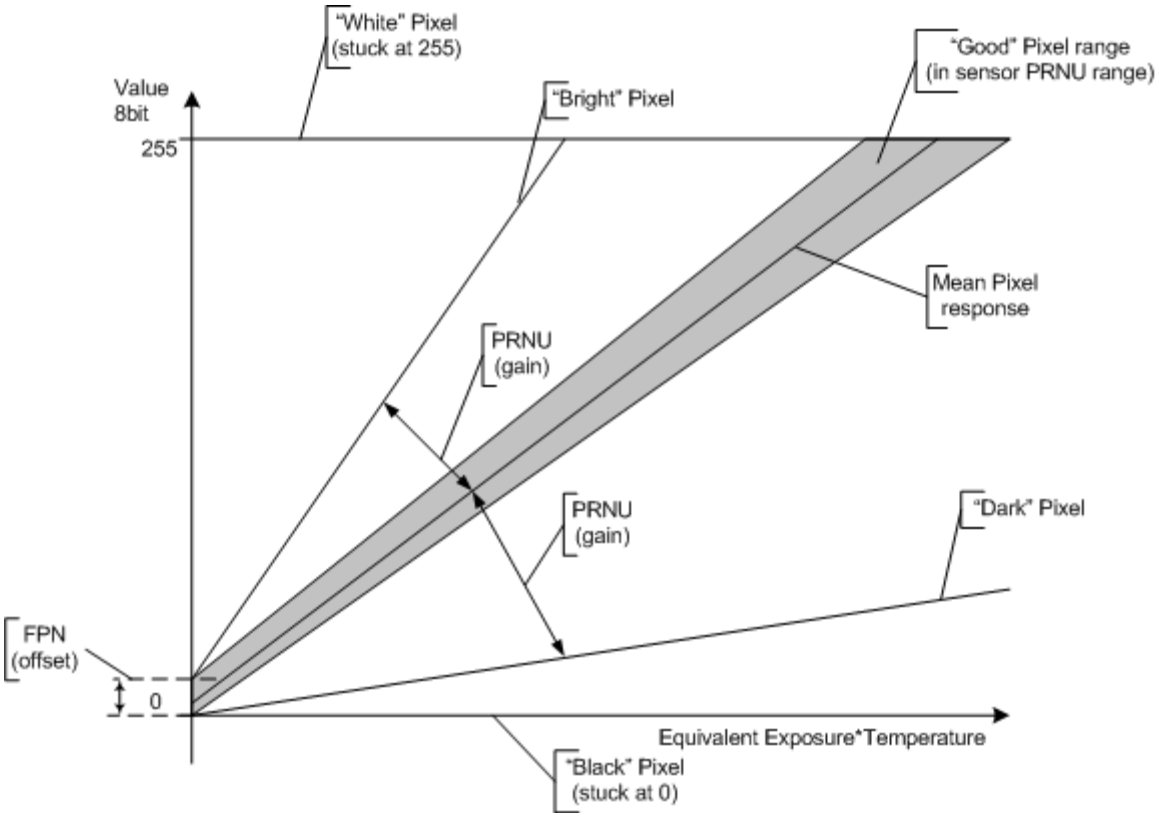


Figure 1 Static Pixel Defects

2.4. Proposed Algorithm

The proposed algorithm approaches the initial problem by first detecting bad pixels that are stuck-at (some of the FPN) then detecting bad pixels due to PRNU. When those two steps are done, a map with the location of the bad pixels over the sensor array is built; at this point the correction of the detected bad-pixel needs to be done, which is not described here. The proposed algorithm is particularly suited to a imaging pipe that does on-the-fly pixel processing and contains at least three blocks:

1. a RAW collection histogram, which collects pixels without color discrimination;
2. a location table that contains coordinates and values;
3. a bad-pixel detection block, which is essentially a local filter (any or similar to ones described in[1256] can be used).

The main assumption used in the detection block is that the difference between a bad-pixel and a normal one is based on the point spread function (PSF) of the pixel, i.e. PSF of a bad pixel usually smaller than of normal sources, and their edges are usually sharper. Normal sources with very sharp PSFs cause aliasing in most bad-pixel detection methods and the situation gets worse with clustering of bad-pixels, which is typically caused by transistors shared among neighboring pixels.

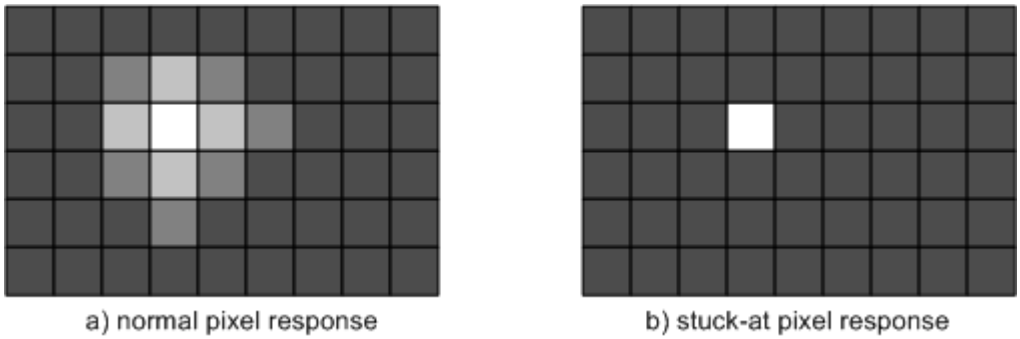


Figure 2 bad-pixel vs normal pixel area response

Consequently, a profile would look like this:

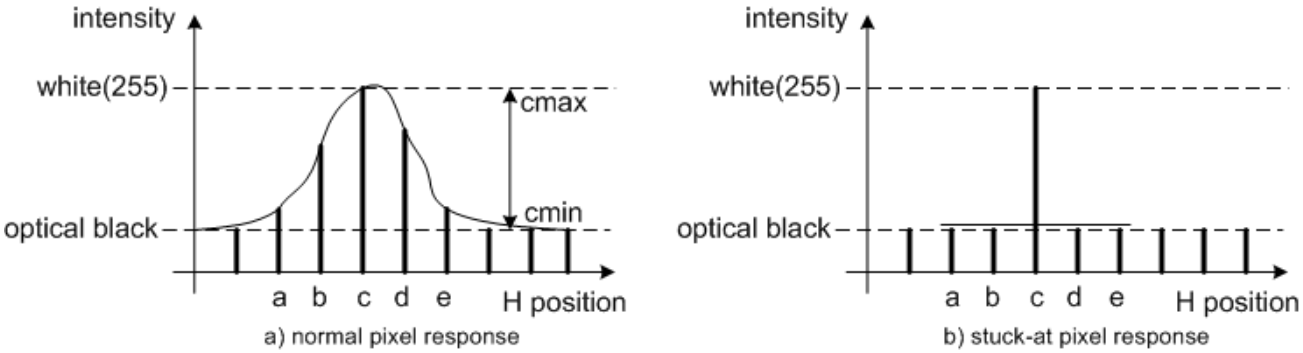


Figure 3 bad-pixel vs normal pixel 1D profile

To prevent aliasing, the algorithm uses the detection block to fill a location table and, using consequent steps, validates whether or not the pixels found are actually faulty. The sequence of events is as follows:

1. Use bad-pixel detection block to detect Bad Pixels due to stuck-at faults and also bad-pixels due to FPN that have a much higher than usual offset:
    a. Set sensor exposure to min to collect a 'Dark Exposure' frame (use a higher that 1x analog gain to enhance the FPN pixels that have a unusual high offset;
    b. Collect Bad Pixel Table;
    c. Analyze Bad Pixel Table and histogram (the histogram should look as a) in Figure 4);
2. Use bad-pixel detect block to detect Bad pixels due to PRNU (repeat 2 for a number of frames until the result becomes consistent):
    a. Set sensor exposure such that there are a number of 'White pixels' in bins close to max but image is underexposed such that there are empty bins in the histogram between the image content and the top of the histogram (see b) in Figure 4);
    b. Collect Bad Pixel Table;
    c. Analyze Bad Pixel Table and histogram;

d. Increase exposure and repeat b-c (to avoid aliasing with elements from the scene, the histogram needs to have empty bins between the image area and the white-pixels area).

e. Validate 'White pixels' for the number of 'White pixels' that have same location and are found at the top bins of the histogram (ie. have same behavior for both exposures). At the same time, Histogram Mean changes according to exposure change. Also, some empty bins have to be in both histograms between 'Histogram mean' and 'White pixels bins' (see b) and c) in Figure 4).

3. Combine results from 1 and 2 and update the Bad Pixels Table (pixels found in step 1 have to be found in step 2 as well);

At 3, the location table is filled with valid bad-pixels and at this point the detection block can be disabled, using the location table to identify the bad-pixels that are corrected using one of the many available methods.

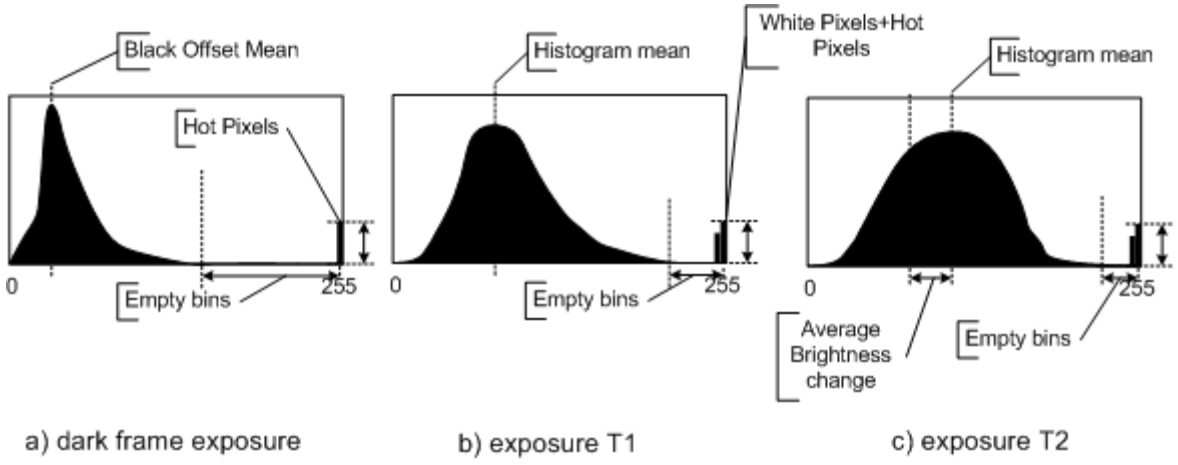


Figure 4 Histograms of the captured frames

Two important characteristics of the algorithm are noted:

1. usage of the histogram of all image reduces the aliasing of the local detection algorithm by making use of the empty bins found in the histogram in the presence of pixels whose value varies unnaturally from the mean;
2. usage of two different exposures of the same scene where clipping pixels are noted in location as well as in value further reduces the aliasing probability;

3. IMPLEMENTATION

A suggested implementation example follows. The diagram illustrates a generic case of bad-pixel processing that is modified to accommodate the proposed algorithm. The modification consists in adding a write coordinates and value from the bad-pixel detection block to the static location table. Thus the components are:

1. a static bad-pixel correction block, which is a table based bad-pixel correction block. This block can be enabled to correct based on the location (2D coordinates) read from the table (active mode) or can be disabled in which case will be bypassed;
2. a bad-pixel detection block: this is the usual local 1D/2D filter used to detect a presumable bad-pixel. This block can be also used in an active mode when detects bad-pixels and writes them into the static table (writing the coordinate of the pixels and its value) or in passive mode, when writing to the location table is disabled;
3. a histogram collection block, which can be collecting a histogram of RAW data

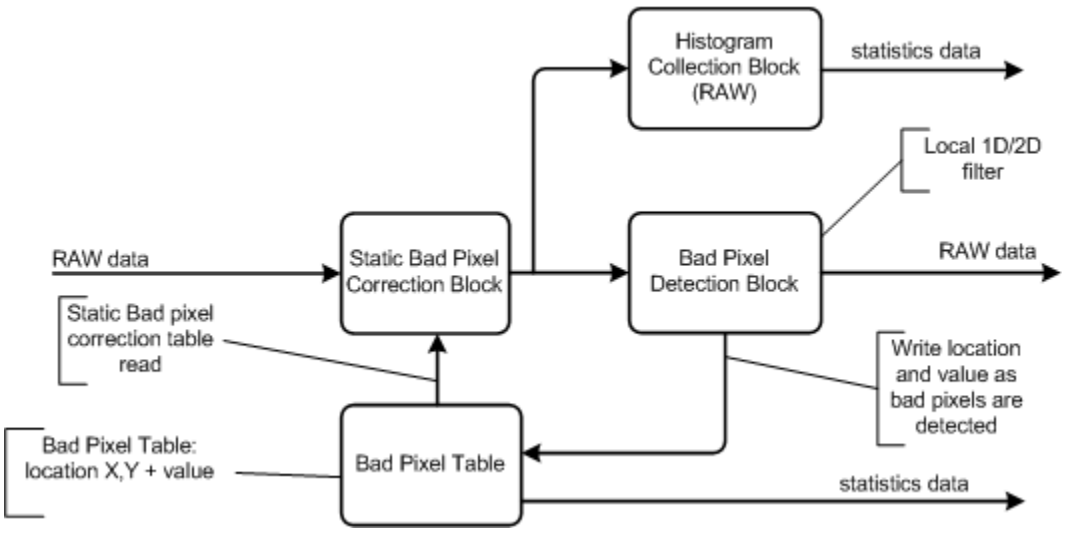


Figure 5 Bad Pixel Correction Block Diagram

This implementation has two distinct operation modes:

1. the 'detection' mode which is the mode where the proposed algorithm is used to fill and validate the content of the location table. This mode is used on power-on or triggered on demand and is active for a limited number of frames.
2. the 'correction' mode which is the normal functioning mode of the block.

The bad-pixel location table used in this implementation is shown in the following figure:

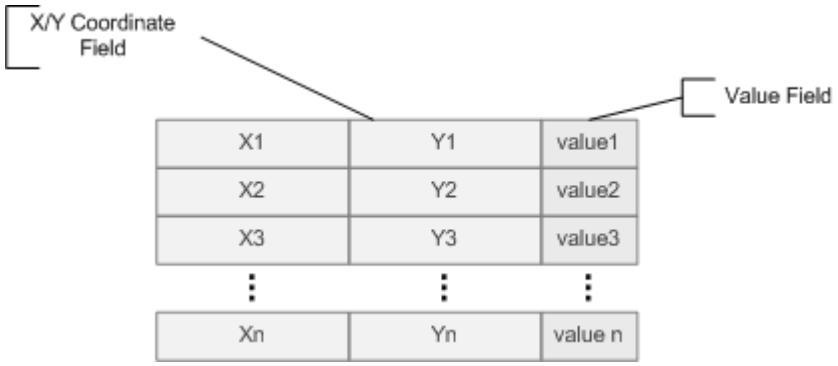


Figure 6 Bad Pixel Table Memory Fields

The memory used for Bad Pixel table has the following fields:

- X coordinate of pixel;
- Y coordinate of pixel;
- pixel value (this field is used in table collection phase to correlate results with the collected histogram);

4. RESULTS

To evaluate the performance of the proposed algorithm, the following methodology was used:

1. a 6000k light source was used, centre of the scene illumination was 10000lux to emulate a bright situation (in case a corner situation is tested, where a 'dark exposure' is captured from a very bright scene). The tests were repeated for 500lux with similar results.
2. three frames were collected using a common scene using an appropriate set of exposure/gain;
3. using the same settings, three dark frames were captured;
4. same algorithm was run in both cases and the list of defective pixels was cross-referenced.
5. For the purpose of presenting these images, a digital gain of 4 was applied to all (the histograms have no gain, so they are from the original images).

A 3 mega-pixel cell-phone camera CMOS sensor was used, running at 15 frames/second (2048 pixels and 1536 lines) with a SNR ~48dB. For this particular sensor, the minimum integration time was 1 line or aprox. 42us.

Step 1: capture the 0 exposure frame:

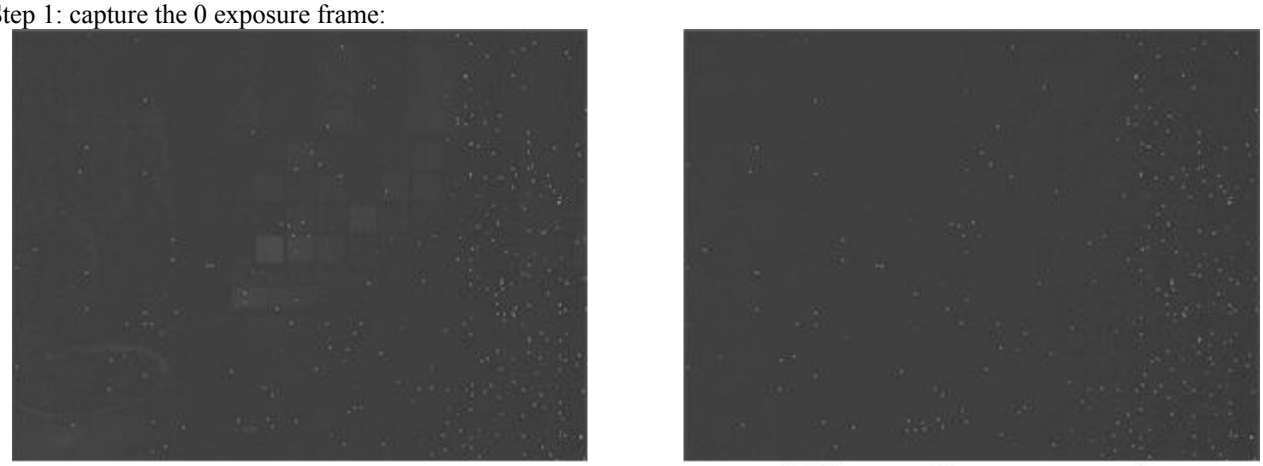


Figure 7 Step 1: collect a 'Dark Exposure'. Images from scene (left) and with sensor covered (right)

Because the illumination was strong, even with minimum allowed exposure (which in this particular case was 1 line ~42us) some information from the scene can be seen. For this sensor, in this step, analog gain was set to 6x.
The associated histograms are:

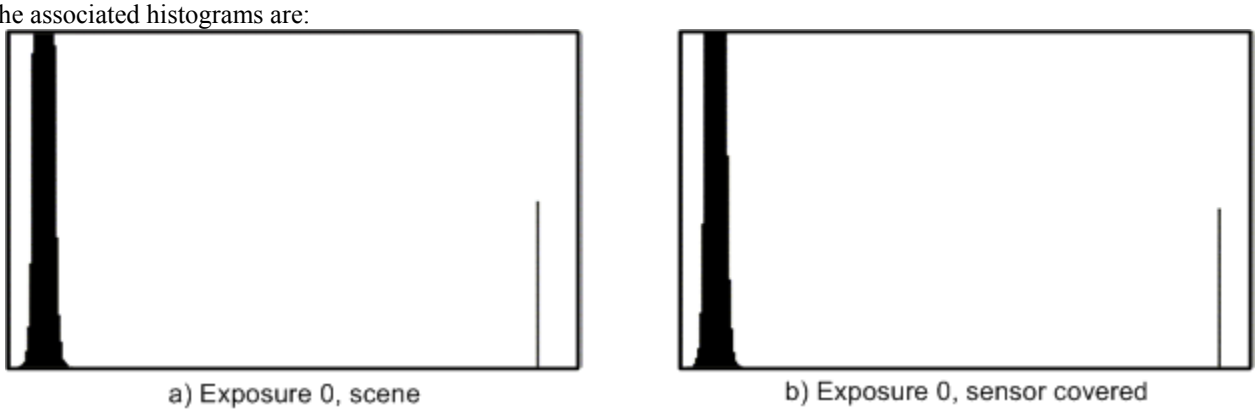
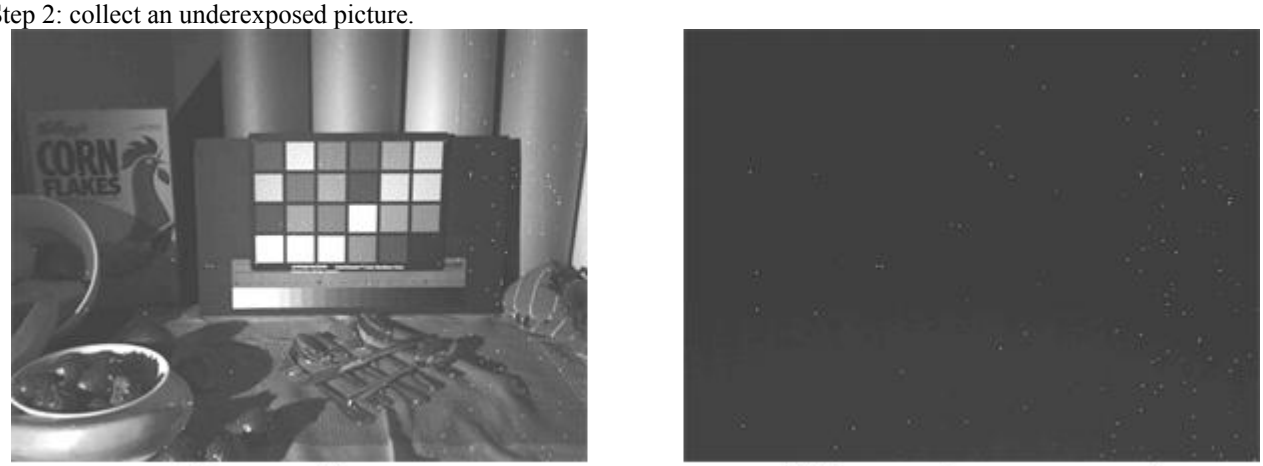


Figure 8 Step1: Histograms from scene (left) and with sensor covered (right)

In this step, as was expected, the histograms are very similar. The stuck-at 1 or 'white pixels' can be seen in the right most bin of the histogram.

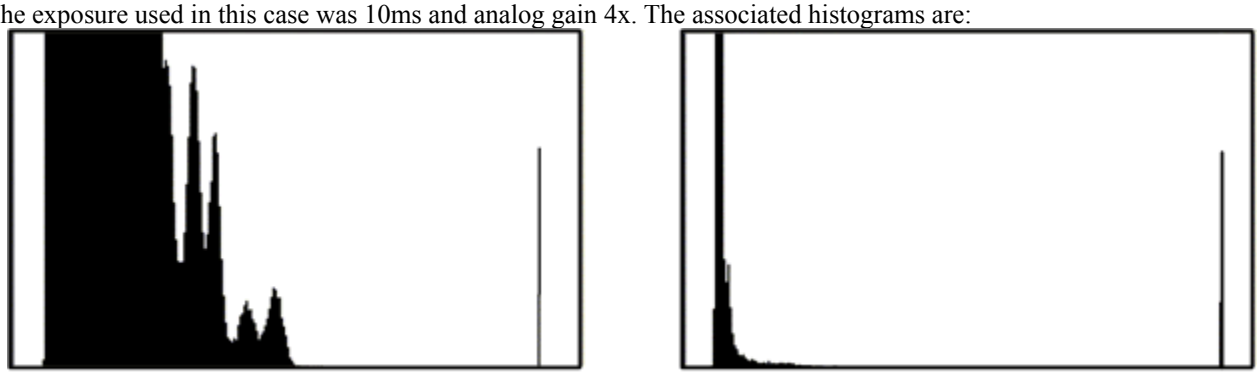
Step 2: collect an underexposed picture.



a) Exposure t1, scene b) Exposure t1, sensor covered

Figure 9 Step2: Exposure T1: collect and underexposed picture. Images from scene (left) and with sensor covered (right)

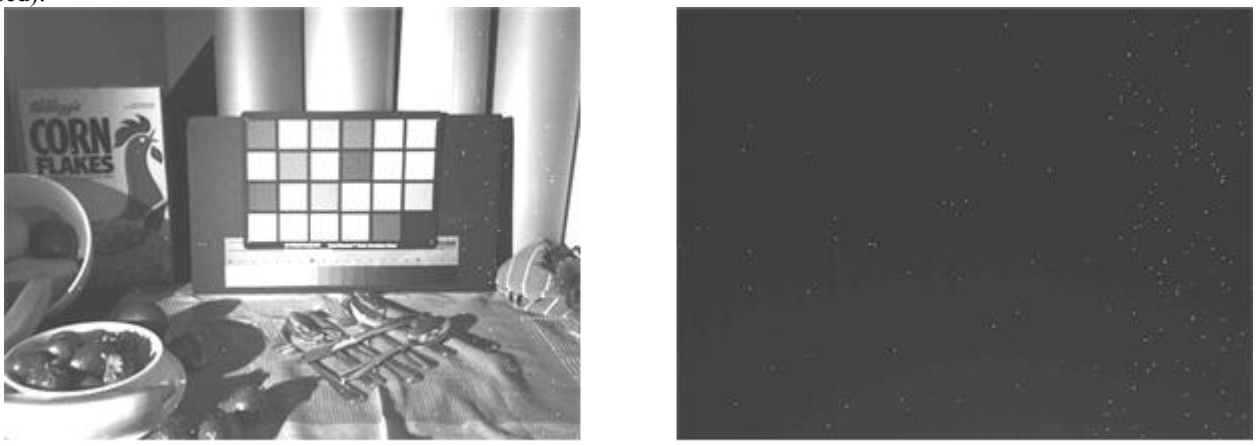
The exposure used in this case was 10ms and analog gain 4x. The associated histograms are:



a) Exposure t1, scene b) Exposure t1, sensor covered

Figure 10 Step2: Histograms from scene (left) and with sensor covered (right)

Here, the histograms show the expected difference, and still the white-pixels can be differentiated.
Step3: collect an underexposed picture with a increased exposure compared to step 2 (for this step exposure t2=2*t1 was used):



a) Exposure t2, scene b) Exposure t2, sensor covered

Figure 11 Step3: Exposure T2: collect another underexposed picture with T2>T1. Scene (left) and with sensor covered (right)

The associated histograms follow:

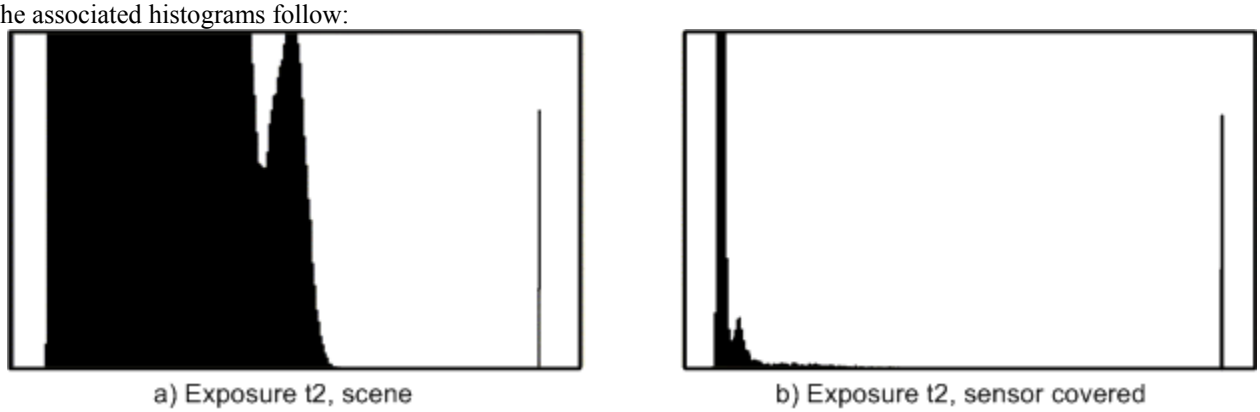


Figure 12 Step3: Histograms from scene (left) and with sensor covered (right)

As expected, the image part of the histogram shows the increased exposure, but since there are still empty bins in the histogram (the image is underexposed), there is a low probability of aliasing for the pixels collected in the top bins of the histogram.

In the final step, results from all three steps are cross-referenced and possible aliasing is removed. In the tested cases, the algorithm proved to find 100% of the pixels that were stuck-at, clustered or not, and also the 'bright pixels' for the explored equivalent exposure. There were some detected pixels that were removed by the algorithm in steps 2 and 3, and in this particular experiment, those pixels proved to manifest as intermittent faults.

4. CONCLUSIONS

An algorithm that addresses the problem of static bad-pixel detection using a location table was presented. Also, an implementation suitable for cell-phone camera usage that does not require full-frame storage or the presence of a mechanical shutter was suggested. The results show that this type of algorithm can be used during a user power-on calibration, thus eliminating the need for per-phone, in factory, bad-pixel calibration.

REFERENCES

1. James E. Rhoads, "Cosmic-Ray Rejection by Linear Filtering of Single Images", Pub. ASP, **vol. 112,** pp703-710, 2000.
2. Mark Cresitello-Dittmar, Thomas L. Aldcroft, David Morris, "On the Fly Bad Pixel Detection for the Chandra X-ray Observatory's Aspect Camera", Pub. ASP, **vol. 238,** pp438-442, 2001.
3. Abbas El Gamal, Boyd Fowlera, Hao Minb, Xinqiao Liua, "Modeling and Estimation of FPN Components in CMOS Image Sensors" , Proc. SPIE, **vol. 3301**, pp. 168-177, 1998.
4. Dalsa Inc., *1998-1999 Databook*, pp303-034, Dalsa, Waterloo, 1998.
5. Guy Meynants, Bart Dierickx ,"A Circuit for the Correction of Pixel Defects in Image Sensors", ESSCIRC '98. Proceedings of the 24th European Solid-State Circuits Conference, pp312-315, 1998.
6. Catherine L. Farage and Kevin A. Pimbblet," Evaluation of Cosmic Ray Rejection Algorithms on Single-Shot Exposures", Publications of the Astronomical Society of Australia, **vol. 22**, pp249-256, 2005