AN IMAGE PROCESSING PIPELINE WITH DIGITAL COMPENSATION OF LOW COST OPTICS FOR MOBILE TELEPHONY

Nikolaos Bellas

Embedded System Research
Motorola, Inc.
Schaumburg, IL

Arnold Yanof

Freescale, Inc.
Phoenix, AZ

ABSTRACT

As digital imaging becomes more prevalent in consumer products, the industry strives to reduce the cost and the complexity of imaging solutions and, at the same time, to improve the color quality and resolution of the images. The proliferation of imaging in mobile phones, digital cameras, webcams, toys, etc. creates the need for a low cost, small footprint image sensor. Nowadays, sensor optics account for approximately half the cost of an image sensor system for a camera phone, and their cost does not scale down as fast as the cost of semiconductors.
In this paper, we describe the algorithms and the hardware implementation of a novel color processing chain that uses image processing techniques to compensate for the spatial variations in image attributes and quality due to low cost optics. If left uncorrected, these variations produce undesirable visual effects and lead to unacceptable image quality. Besides the correction of artifacts due to lenses, the image processing chain performs a sequence of corrections for real time dead pixel replacement, color correction, filtering, color space transformations for subsequent compression, etc.

1. INTRODUCTION

Image sensors are widely used in high volume, space constrained applications such as mobile imaging, toys, barcode scanners, automotive applications, etc. There is a growing interest for CMOS based image sensors because of customer demand for miniaturized, low power, and high integration imaging systems. Moore's law on scaling of CMOS semiconductor technology ensures that the cost of the pixel array and accompanying image processors and DSPs will continue to drop. This leaves the optical lenses and packages the dominant cost contributor of an image sensor system.
The advances in semiconductor integration has prompted image sensor designers to transfer additional functionalities to the image processors in order to improve the quality and to amend inefficiencies at the front end of the system. As an example, the packaging of sensors in mobile devices requires a small z-height optical system in which the distance between the main lens and the active pixel array creates blurring effects and chromatic aberrations near the edges of the sensor (Figure 1). We will focus on this imaging error later in the paper.
Another example is the incomplete manufacturing testing of the pixel array that may leave behind pixels that are stuck at a particular value independent of the light intensity of the scene (called dark and hot pixels). Again, image processing can be used to detect and correct dead pixels in the active pixel array in real time. By making it unnecessary to discard sensor chips that contain limited numbers of scattered defects, the method would increase effective production yields and thereby lower the costs of individual image sensors.
In this paper, a complete image processing algorithm and its hardware implementation is described. Moreover, the paper explains how the correction stages are combined with a traditional pipelined image processing chain used to enhance the quality of the picture. This chain can be used in a mobile telephony device, for example. We present an important and realistic paradigm on how back-end image processing can substitute manufacturing inefficiencies, thus driving the effects of Moore's law into the whole system
.
The paper makes the following contributions:

- It presents novel algorithms to solve the problems of roll off corrections due to low cost optics and of real time dead pixel detection and replacement, and
- it describes how these algorithms can be mapped into a modular ASIC design for a high-speed, low power hardware implementation.

The rest of the paper is organized as follows: section 2

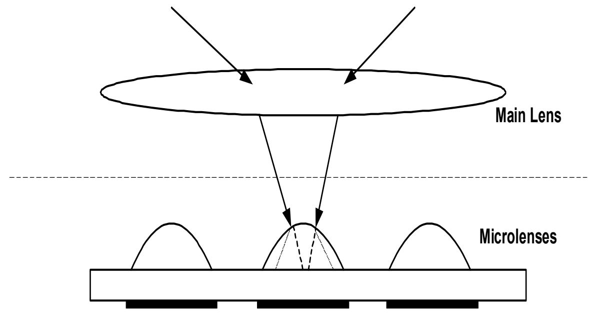

Figure 1 Each pixel is enclosed in microlenses to focus the incident light on photo sensitive portions of the pixel and improve the effective fill factor of the image sensor. If the z-height is small, the image appears blurred at the edges.

details the complete image processing chain. Section 3 presents the pipelined image processor that interfaces to a VGA sensor and implements the algorithms, and section 4 outlines previous work in the area and concludes the paper.

2. IMAGE PROCESSING ALGORITHMS

The color processing pipeline of Figure 2 receives a stream of sensor pixels in Bayer format, and produces a stream of pixels in the YCbCr color space. In some cases, certain values of the sensor integration time cause the appearance of rolling horizontal flickering bars in the image. The processing chain has a flickering detection and correction mechanism which is based on the Fourier transformation of image rows of successive frames. A flickering detection triggers a gradual modification of the sensor integration time to eliminate the flickering effect.

The algorithm computes statistical information on the pixel values including the average R, G, and B values in each frame and the distribution of these values using histogram analysis. The statistical information is used for dynamic updates of the integration time and white balance values in the sensor.

Before most of the image processing can be conducted, dead pixel values must be removed. Traditionally, dead pixel detection and correction is achieved by storing the locations of the dead pixels during sensor manufacturing test. During sensor initialization, these locations are stored in the image processor and the dead pixels are replaced by neighboring, non-dead pixels while the frame is read out. One of the novel algorithmic aspects of our imaging chain is the detection of pixels that are substantially different from surrounding pixels without losing sharpness and spatial details. The algorithm eliminates the need to store the location of the dead pixels a-priori since it does not aim to detect every dead pixel in the image, but to detect the dead pixels that cause obvious visual errors for a particular scene.

While it is easy to devise heuristics to detect suspicious pixels with a high contrast to their surrounding pixels, problems may arise when a mathematical formula is applied to replace them, especially in sites with high spatial frequency. For brevity, we will only detail the detection and replacement of pixels in a red location in Figure 3.

The large separation of red pixels (compared to green pixels) forces us to use surrounding green pixels as well to detect the brightness of the site. If the red pixel is much brighter than its neighbours, it is being substituted by the maximum value of nearby red and green pixels, and not by their average or median value. This is necessary in order to retain highlights in locales with rich spatial detail. Moderately bright red pixels use a smaller threshold to compare against neighboring pixels, but they are only substituted in case of a flat or dark field. It is permissible to detect dark pixels by testing against only surrounding red pixels, since fine details are not adversely affected by brightening dark red pixels. Note that unlike the bright red pixels, the substitution is the mean of the surrounding red and green ones.

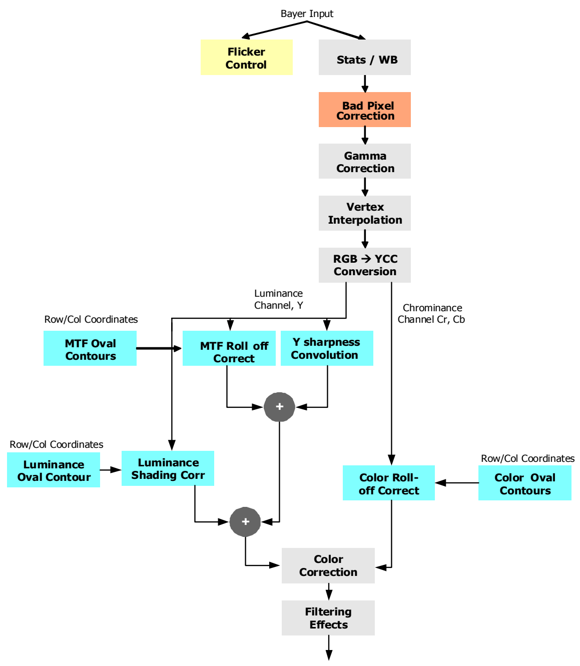

Figure 2 The image processing chain, including flicker control, dead pixel detection and replacement and roll-off corrections

The dead pixel detection and replacement for green pixels is simpler, and utilizes only green surrounding pixels. The process for the blue pixels is similar to the process for the red pixels, with extra precautions for areas in which dark objects are immersed in very bright backgrounds.

Dead pixels have to be replaced before the subsequent filtering and roll off correction stages to avoid errors to be magnified.

Pixels near the corners and the edges receive light at a larger incident angle, which is also more diffused and causes loss of acuity (Figure 4). Moreover, the large incidence angle causes color variations near the edge of the sensor.

The algorithm corrects for roll off in image data by determining for each pixel a roll off contour in which the pixel resides which in turn, depends upon the pixel coordinates on the image plane. The (x,y) coordinates of the location of each pixel are converted to a radial distance from the center of the image, which is used to map the location to the roll-off contour (Figure 5) and to access look-up tables that contain gain parameters.

The non-linear roll off correction is used in three different circumstances: to correct unwanted variability in luminance (Figure 4), loss of image sharpness, and color distortion because the color components RGB do not focus at exactly the same point at the edges of the image.

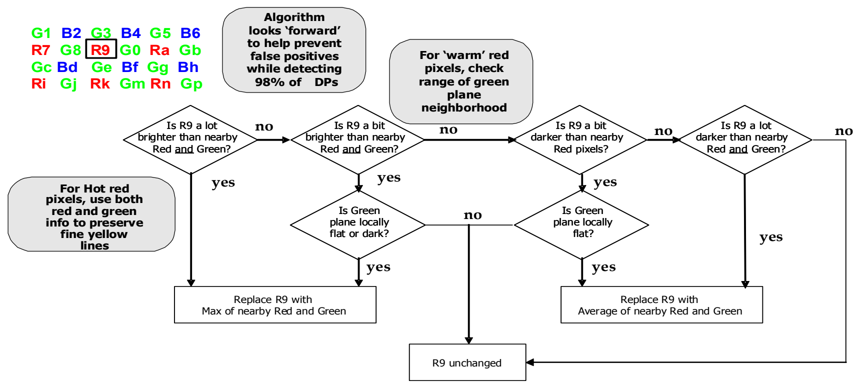
G1 B2 G3 B4 G5 B6
R7 G8 R9 G0 Ra Gb
Gc Bd Ge Bf Gg Bh
Ri Gj Rk Gm Rn Gp
Algorithm looks 'forward' to help prevent false positives while detecting 98% of DPs
For 'warm' red pixels, check range of green plane neighborhood
For Hot red pixels, use both red and green info to preserve fine yellow lines
Is R9 a lot brighter than nearby Red and Green?
Is R9 a bit brighter than nearby Red and Green?
Is R9 a bit darker than nearby Red pixels?
Is R9 a lot darker than nearby Red and Green?
Is Green plane locally flat or dark?
Is Green plane locally flat?
yes
no
Replace R9 with Max of nearby Red and Green
Replace R9 with Average of nearby Red and Green
R9 unchanged

Figure 3 The algorithm for dead pixel detection and replacement of red pixels

The non-linear luminance shading correction is applied in the same Y input stream as the edge enhancement/MTF correction. The shading correction is multiplicative and can increase the luminance value by a factor of 3-4x near the image edge:

$$Roll\_LUT = f1(R); \quad /* \text{ a non-linear function f1(R) } */$$

$$Y_1 = Roll\_LUT * Y_{inc} \quad /* \text{ is used to store the luminance gain } */$$

The luminance stream Y passes through a baseline 3x3 high pass filter to perform edge enhancement, and, simultaneously to restore loss of sharpness through the location dependent MTF correction filter. The equations are the following:

$$E = conv(Y_{in}, m)$$

$$MTF\_LUT = f2(R)$$

$$Y2 = BaseLineEdgeTerm + RollOffEdgeTerm =$$

$$\frac{E * Base}{8} + \frac{E * (MTF\_LUT * RollOff)/2}{128}$$

$$Yout = Y1 + Y2$$

where m is the convolution matrix:

$$m = \frac{\begin{pmatrix} -1 & -2 & -1 \\ -2 & +12 & -2 \\ -1 & -2 & -1 \end{pmatrix}}{16}$$

and *Base* and *RollOff* are user defined normalization factors. The *f1* and *f2* are non-linear, monotonically non-decreasing functions of the radial distance *R* of the pixel from the optical center and are implemented in hardware using lookup tables (LUTs). The chroma pixels do not pass through edge enhancement, but only through roll off correction. The chroma correction is additive to the baseline chroma magnitude, so that total corrective effect is much smaller than in the luminance roll off case, although the image errors can be more objectionable than in the Y pixels. A final color correction and optional filtering effects stage conclude the image processing pipeline.

3. HARDWARE IMPLEMENTATION

The imaging pipeline is implemented as a synthesizable image sensor companion chip. Therefore, it has to meet stringent real-time performance requirements and operate within a low power budget. The design can process Bayer data from a VGA (640x480) image sensor at 30 frames/sec using a max clock frequency of 33 MHz for low power operation. In Figure 6, each of the three multi-cycle pipe stages execute parts of the processing chain. The low clock

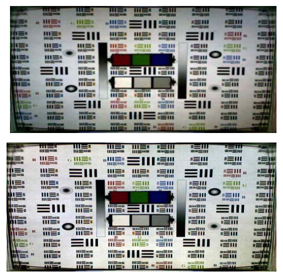

Figure 4 Roll-off luminance correction is needed to compensate the shaded areas at the edges due to small z-height

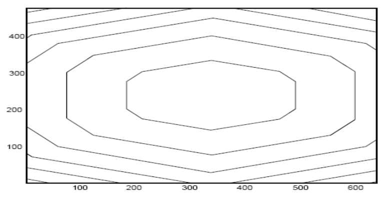

Figure 5 An oval contour made of linear segments simplify the real time conversion of Cartesian coordinates to radial distance and covers a large set of commercially available lenses. The luminance gain increases as the radial distance from the center increases.

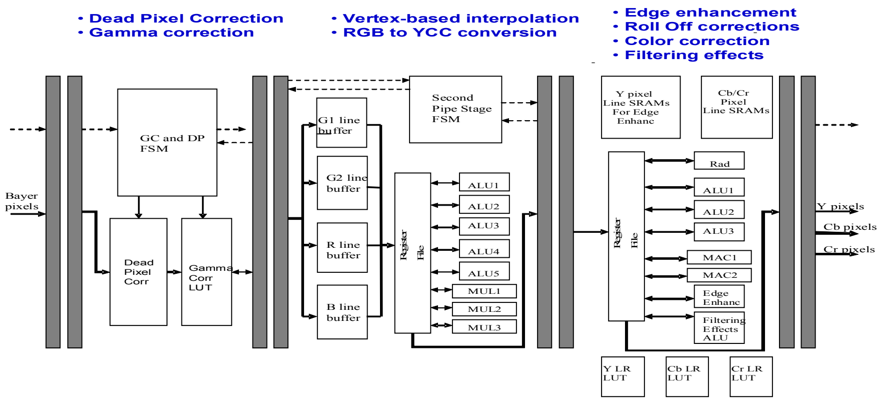

Figure 6 The pipelined architecture of the image processing chain

frequency restriction requires a large number of functional units operating in parallel in each stage to achieve the real time performance by exploiting the high instruction and data level parallelism of the algorithms. The design is modular and can be easily extended to different algorithms, different performance requirements and sensor sizes.

The control signals are used to trigger a data transmit transaction from the sensor to the processor or, more generally from pipeline stage I to pipeline stage I+1. An ACK signal from stage I+1 back to stage I notifies the data sender that stage I+1 has read the data sent by stage I. A double buffer between the stages ensures that successive stages can write and read data simultaneously (although to different buffers).

The data path consists of ALUs, Multiply Accumulate (MAC) units, and dedicated hardware to speed up certain filtering operations. Separate control units are used to control the operations and communicate with neighboring stages. Four line buffers are needed to store Bayer pixels for the 2D filters. The buffers are dual ported to facilitate simultaneous access from different pipeline stages.

The latency in each stage is determined by the ratio of the core clock frequency to the sensor clock frequency. In this design, this ratio is set to three, and this makes the stage latency equal to six cycles in the worst case. The worst case happens when there is an 1:1 or 2:1 interpolation, such as transforming a VGA Bayer frame to a VGA or QVGA YCbCr output. The flicker correction and statistics gathering phases are executed by a small microcontroller before the pixels make it into the pipeline. The microcontroller and accompanying hardware to speed up the flicker correction and histogram analysis are in the same die with the pipelined architecture of Figure 6.

The chip has about 250K gates, and consumes 35 mW power when processing a VGA input frame. Multiple clock domains are used to provide clock gating in fine granularity. For example, when the imaging system operates in single capture mode, the chip and the sensor can be placed into a low power state by clock gating the flip flops.

4. CONCLUSIONS & RELATED WORK

Programmable or ASIC-based image processors have been used successfully to trade-off cost and image quality with processing complexity. In this paper, we described an image acquisition system which captures Bayer RGB data and produces formatted YCbCr data for compression. The proposed algorithm and image processor utilizes correction techniques to minimize adverse visual side-effects such as flickering, dead pixels, and roll-off, owing to the low cost acquisition system.

Previous such systems focused mostly on the color processing and color space conversion without consideration of the optics [1] [2]. Both software and hardware techniques for dead pixel correction have been proposed in [3]. Commercial products that use some form of dead pixel correction and lens shading correction as part of their color processing pipeline have been announced by Freescale and Micron [4].

5. REFERENCES

[1] B.Tang, K.Lee, "An Efficient Color Image Acquisition System for Wireless Handheld Devices," *Proceedings of Acoustics, Speech, and Signal Processing, 2004 (ICASSP '04).* Vol.3, page 105-108, May, 2004.

[2] T. Sakamoto et. al. "Software píxel interpolation for digital cameras suitable for a 32-bit MCU," *IEEE Transactions on Consumer Electronics,* vol. 44, no. 4, pp.1342-1352, 1998

[3] Chapman, G.H., Djaja, S., Cheung, D.Y.H., Audet, Y., Koren, I., Koren, Z. "A self-correcting active pixel sensor using hardware and software correction," *IEEE Design & Test of Computers,* Vol. 21, Issue 6, Nov-Dec 2004, pp: 544 – 551

[4] "MT9D111: 2-Megapixel CMOS Camera System-on-a-Chip," *www.micron.com*