May 2023

Pixel Burn-In Compensation Parameter Compression for Organic Light Emitting Diode Displays

Peter Kim

Sang Young Youn

Recommended Citation
Kim, Peter and Youn, Sang Young, "Pixel Burn-In Compensation Parameter Compression for Organic Light Emitting Diode Displays", Technical Disclosure Commons, (May 08, 2023)
https://www.tdcommons.org/dpubs_series/5871

Pixel Burn-In Compensation Parameter Compression for Organic Light Emitting Diode Displays

Abstract:

This publication describes techniques directed to improved compression algorithms that reduce the size of look-up tables used to store pixel burn-in compensation parameters that are used by Organic Light Emitting Diode (OLED) display burn-in compensation algorithms. The improved compression algorithms include an x-y mapping compression algorithm and a line fitting parameters compression algorithm. The improved compression algorithms simplify logic, save memory space, conserve power, and/or reduce latency relative to many common compression techniques.

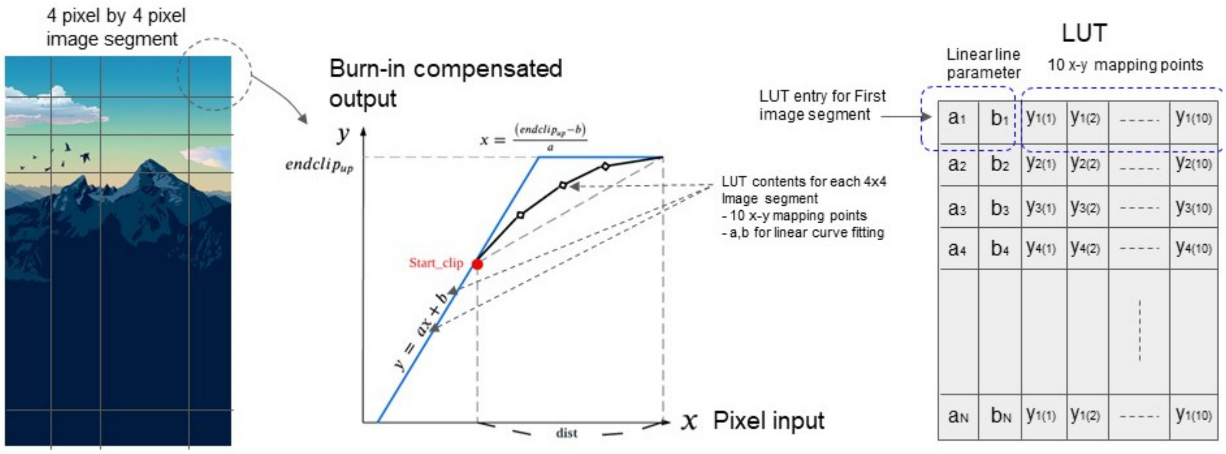
Keywords:

Algorithm, model, compression, pixel degradation, burn-in compensation, look-up table, LUT, compensation gain, compensation parameter compression, compensation algorithm LUT, input-output mapping compression, x-y mapping compression, input pixel count value, linear relation, bounded limit, linear line parameter, delta, line fitting parameters, compensated output y, quantized range encoding, lossless

Background:

An Organic Light-Emitting Diode (OLED) display can incur pixel efficiency degradation, which results in the display of an undesirable residual image (e.g., ghosting or another irrecoverable defect, collectively "burn-in" herein) on the OLED display. The amount of pixel efficiency degradation depends on the driving conditions for each pixel. For example, a static image, such as a default user interface, may be displayed on specific pixels more often than other pixels and be vulnerable to cause burn-in.

To reduce burn-in, a burn-in compensation algorithm may be used. In an example, a burn-in compensation algorithm directs the computing device to count data usage at a specific image segment (e.g., 4x4 or 5x5 pixels become one segment), which indicates how much the image segment pixels have aged. Then, the burn-in compensation algorithm directs the computing device to look up compensation gain for each RGB channel from an 8-bit burn-in LUT. The intensity of a pixel may be a value of 0 to 255 for each RBG value, requiring eight bits (256 bytes) of storage per entry. The 8-bit burn-in LUT provides values to avoid burn-in by increasing the brightness of an image segment at risk for burn-in or dimming other neighboring image segments to avoid large contrasts. Figure 1 illustrates an example of a burn-in compensation algorithm.



Linear line parameter		10 x-y mapping points			
a_1	b_1	$y_{1(1)}$	$y_{1(2)}$	-----	$y_{1(10)}$
a_2	b_2	$y_{2(1)}$	$y_{2(2)}$	-----	$y_{2(10)}$
a_3	b_3	$y_{3(1)}$	$y_{3(2)}$	-----	$y_{3(10)}$
a_4	b_4	$y_{4(1)}$	$y_{4(2)}$	-----	$y_{4(10)}$
				⋮	
a_N	b_N	$y_{1(1)}$	$y_{1(2)}$	-----	$y_{1(10)}$

Figure 1

As illustrated in Figure 1, the 8-bit burn-in LUT includes twelve values for each image segment: two linear line parameters (e.g., “a” and “b”) and ten x-y mapping points for each 4x4 image segment. The 8-bit burn-in LUT may relate the pixel input display input values for first linear line parameters “a”, “b” to corresponding measured display output values. Assuming 4x4 pixel image segments, for a 1344 x 2992 pixel display, 336 x 748 image segments are defined, which correspond to 251,328 LUT entries. Given that each 8-bit burn-in LUT entry is eight bits for the twelve values with three RGB channels and ten Display Brightness Values (DBV) bands, the total 8-bit burn-in LUT size for a 1344 x 2992 display may be as high as 90.1 Megabytes. Accordingly, the 8-bit burn-in LUT consumes a large memory space on the device and is a large burden for the system-on-chip (SOC). This large size further necessitates the utilization of compression and decompression algorithms, making the traditional burn-in compensation algorithm process complicated, time-consuming, and power-consuming.

Description:

This publication describes techniques, implemented on computing devices (e.g., smartphones, tablets, cameras, tablet computers, computers, smart watches, intelligent glasses, and so forth), directed to improved compression algorithms, which utilize compensation parameter compression, used with burn-in compensation algorithms. The improved compression algorithms simplify logic, save memory space, conserve power, and/or reduce latency relative to many common compression techniques. The improved compression algorithms include an x-y mapping compression algorithm and a line fitting parameters compression algorithm.

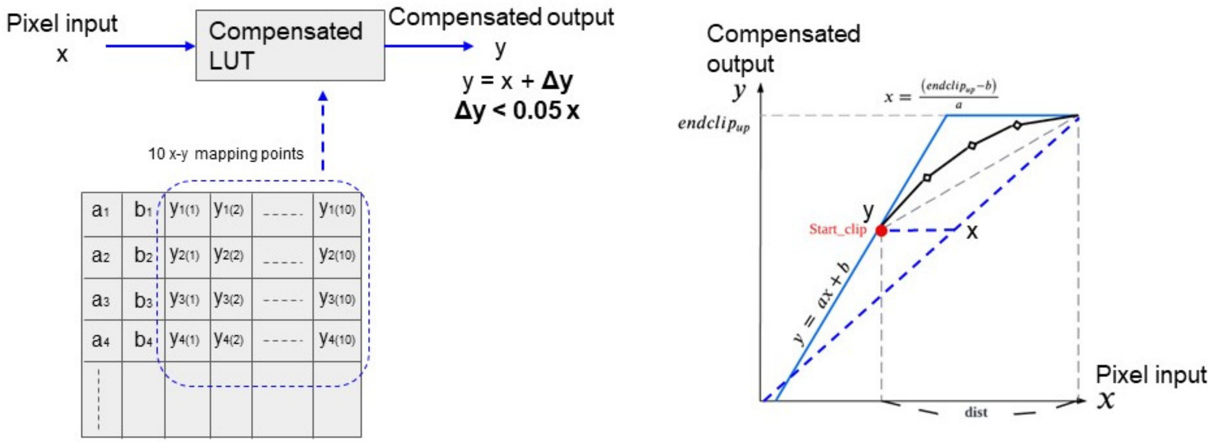
X-Y Mapping Compression Algorithm

Based on OLED display burn-in performance specifications, a typical pixel burn-in compensation is, at most, 5% of color and luminance (e.g., brightness) changes. Accordingly, a difference between an input pixel count value (e.g., x) and a compensated output (e.g., y) may be bounded by a 5% difference. By multiplying an input pixel count value by .05, the worst-case scenario for the 5% difference (e.g., delta compensation (Δy)) may be computed. Equation 1 illustrates an equation to compute a Δy based on a 5% compensation difference, and Equation 2 illustrates an equation to compute a compensated y for each input pixel count value x.

Equation 1: $\Delta y < .05x$

Equation 2: $y = x + \Delta y$

Equations 1 and 2 significantly simplify burn-in compensation algorithms by utilizing an x-y mapping compression. In an aspect, an x-y mapping compression algorithm computes a Δy for ten x-y entries, stores the Δy in a compensated LUT, retrieves Δy for a mapped x-y point, and computes a compensated output y, as illustrated in Figure 2.



a_1	b_1	$y_{1(1)}$	$y_{1(2)}$	-----	$y_{1(10)}$
a_2	b_2	$y_{2(1)}$	$y_{2(2)}$	-----	$y_{2(10)}$
a_3	b_3	$y_{3(1)}$	$y_{3(2)}$	-----	$y_{3(10)}$
a_4	b_4	$y_{4(1)}$	$y_{4(2)}$	-----	$y_{4(10)}$
⋮					

Figure 2

The compensated LUT can store Δy for color and luminescence between x and y instead of storing an entire compensated output y for ten x-y mapping entities to provide compensation gain for each input pixel value x. In Figure 2, the dashed box indicates ten x-y mapping points that can be simplified by an x-y mapping compression to Δy. The burn-in compensation assumes a 5% difference of color and luminance change between x and y. The x-y mapping compression architecture is illustrated below in Figure 3.

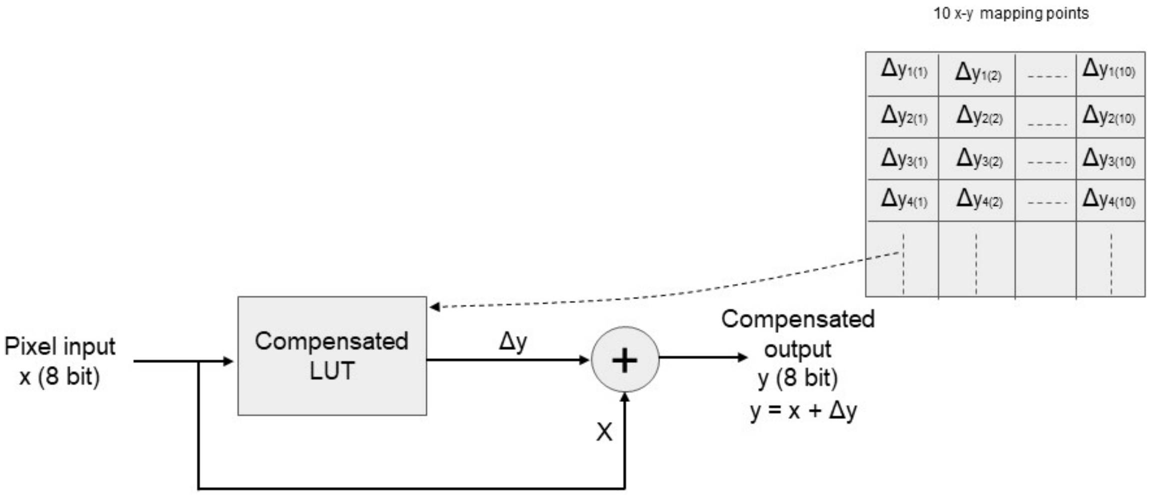


Figure 3

As illustrated in Figure 3, the compensated LUT allows a computer system to use significantly less memory by storing Δy for ten x-y mapping points. The compensated LUT provides a value Δy which can be added to x to find y using Equation 2. As discussed previously, if x pixel input is 255 for R, G, or B and requires eight bits of storage, it can be calculated by Equation 1 that Δy = 13. This value requires only four bits of storage in the compensated LUT, considerably less than the eight bits required in the 8-bit burn-in LUT. With a compensation gain of 5%, an estimated 50% compression rate can be achieved with the compensated LUT. The compression algorithm is able to estimate the compression rate for the mapping points in accordance with the values for Δy. Alternatively, with a compensation gain of 10%, by Equation 1 (Δy = 26), a 37.5% compression rate can be achieved with the compensated LUT. The x-y compression mapping architecture allows encoding to be done off-line, allows only the final encoded Δy values to be saved, simplifies the decoding by using an adder, and allows for much smaller memory usage requirements that do not burden the SOC.

Line Fitting Parameters Compression Algorithm

Based on OLED display burn-in performance specifications, the typical burn-in compensation may be bounded by a value that is a max of 10% higher than x. Accordingly, the line fitting parameter "a" may not deviate from a reference line x = y for more than 10%. The line fitting parameter "a" may be used to simplify the burn-in compensation algorithm by utilizing linear line parameter encoding and line fitting parameters compression in a line fitting parameters LUT. The disclosed techniques may compute Δa based on encoding parameters, store Δa and "b" in a line fitting parameters LUT, retrieve Δa and "b" for an input pixel value x, and compute a compensated output y.

For example, to compute the worst-case scenario of 10% deviation for a linear line parameter encoding, Equation 3 is an equation to compute a compensated y based on its relationship with x and Equation 4 is an equation to describe the slope of a pixel performance's linear behavior to compute compensated y, as illustrated below in Figure 4.

Equation 3: $y < 1.1x$

Equation 4: $y = ax + b$

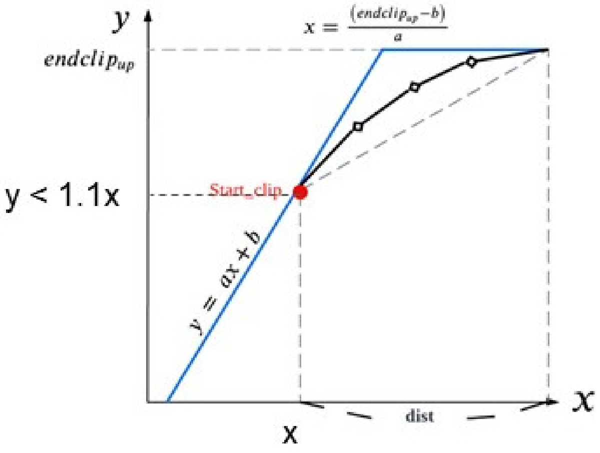
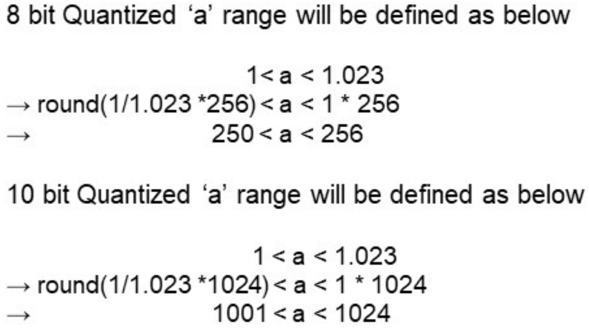


Figure 4

By simple algebra, Equations 3 and 4 are combined to produce Equation 5:

Equation 5: $a < (1.1\,x - b)/x$

Equation 5 may be used to calculate a bounded limit for parameter "a" (e.g., "slope") for the linear line fitting (e.g., shown in Figure 4) based on a starting point of deviation from linear behavior. As illustrated in Figure 4, the line $y = ax + b$ has a deviation from linear behavior which begins at the point indicated as start_clip. Using typical worst-case values $b = 10$ and $x = 127$ for the start_clip in Equation 5 yields the result for bounding values of "a" as $1 < a < 1.023$. In example embodiments, the display pixels may be 8-bit or 10-bit and the line fitting parameters compression algorithm may calculate a bit range for parameter "a" using a quantized range encoding operation as illustrated in Figure 5.



8 bit Quantized 'a' range will be defined as below

$$1 < a < 1.023$$
$$\rightarrow \mathrm{round}(1/1.023 * 256) < a < 1 * 256$$
$$\rightarrow 250 < a < 256$$

10 bit Quantized 'a' range will be defined as below

$$1 < a < 1.023$$
$$\rightarrow \mathrm{round}(1/1.023 * 1024) < a < 1 * 1024$$
$$\rightarrow 1001 < a < 1024$$

Figure 5

The range of parameter "a" may be only six (three bits) in the 8-bit display pixel analysis and only twenty-three (five bits) in the 10-bit display pixel analysis. Now that the linear line parameter encoding has produced a range for parameter "a", the line fitting parameters compression algorithm may perform compression for line fitting parameters by allocating bits for parameters "a" and "b" using a subtraction then 4-bit quantization operation as illustrated in Figure 6.

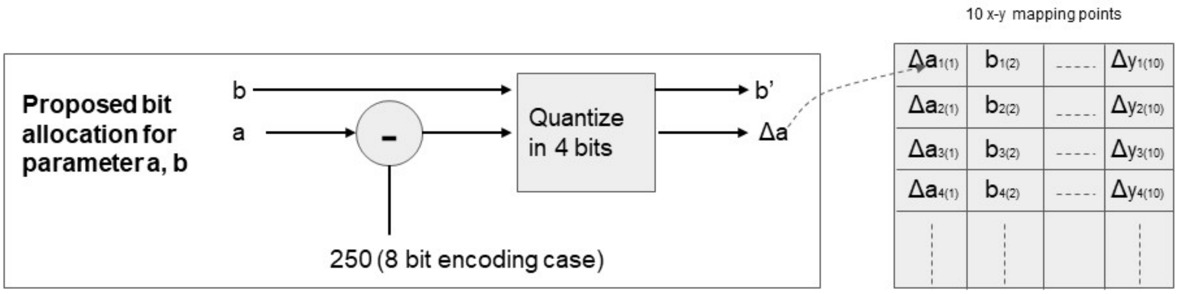


Figure 6

As illustrated in Figure 6, the line fitting parameters compression can perform 4-bit quantization of parameter “a” after subtracting allocated bits within a quantized range from original allocated bits of parameter “a”. For an 8-bit pixel, the allocated bits within the quantized range are 250, as defined in Figure 5. Parameter “b” is typically a small number, and four bits are enough for bias. Thus, instead of saving parameter “a” in full 8-bit resolution in a LUT, saving only $\Delta a = a - 1$ in a Linear Curve Parameter LUT saves 50% of bits for parameter “a”: $0 < \Delta a = a - 1 < 1.023$. By storing significantly fewer bits for parameters “a” and “b,” the line fitting parameters LUT can be compressed by 50% from an 8-bit burn-in LUT.

In an embodiment for decoding the Linear Curve Parameter LUT, the algorithm may implement an adder as illustrated in Figure 7. The adder adds the bits allocated in the quantized range to the parameter Δa to reconstruct the original bits allocated for parameter “a” and obtain an x-y mapping relation $y = ax + b'$.

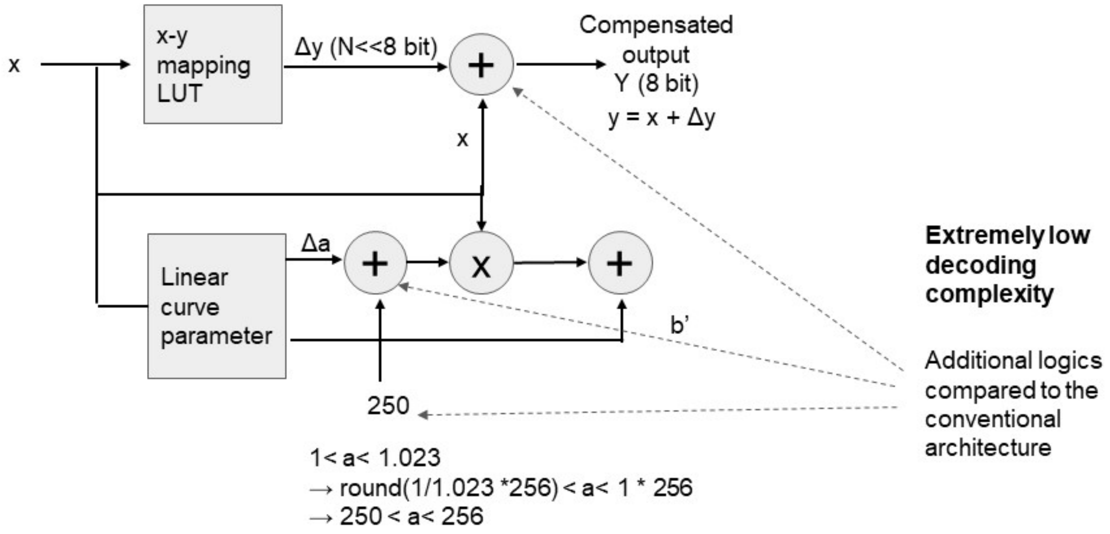


Figure 7

For an 8-bit pixel input, the adder may add 250 to Δa as defined in Figure 5 to decode and decompress the Linear Curve Parameter LUT. The use of simple decoding requires minimal cost and size increases for the decoder logic.

Computing Devices

The computing device performing compression may include a processor (e.g., SOC), a display (e.g., OLED display), and a computer-readable medium that stores device data (e.g., user data, multimedia data, applications, an operating system). The CRM can include any suitable memory or storage device, such as random-access memory (RAM), dynamic RAM (DRAM), non-volatile memory (e.g., flash memory), read-only memory (ROM), a dual in-line memory module (DIMM), a solid-state drive (SSD), and so forth. The device data may include instructions that, when executed by the processor, cause the processor to implement a burn-in compensation algorithm directed at reducing OLED burn-in.

Conclusion:

The above-described compression algorithms provide a lossless compression unless burn-in compensation gain is more than 10%. In this way, the disclosed techniques provide an improved version reducing OLED pixel degradation by providing more efficient pixel burn-in compensation parameter compressions.

References:

[1] Patent Publication: US20130256672A1. Liquid Crystal Display Device. Priority Date: March 11, 2013.

[2] Patent Publication: US20190132001A1. A Compressing Method of a Grayscale Compensation Table of an OLED Display Panel. Priority Date: November 2, 2017.

[3] Patent Publication: US20210183333A1. Burn-in Statistics with Luminance Based Aging. Priority Date: December 11, 2019.

[4] Patent Publication: US20190052904A1. Systems and Methods for Model Parameter Optimization in Three Dimensional Based Color Mapping. Priority Date: October 17, 2018.

[5] Patent Publication: US20080055210A1. Method and Apparatus for Uniformity and Brightness Correction in an Electroluminescent Display. Priority Date: June 20, 2007.

[6] Patent Publication: US20020067326A1. Liquid Crystal Display, Image Data Compensation Circuit, Image Data Compensation Method, and Electronic Apparatus. Priority Date: November 28, 2001.

[7] Antonio-Torres, David, Lister, Paul F., and Newbury, Paul. "LUT-based Compensation Model for OLED Degradation." Journal of the Society for Information Display 13, no. 5 (2005): 435. https://www.researchgate.net/publication/245413898_LUT-based_compensation_model_for_OLED_degradation, DOI: 10.1889/1.1927736.

[8] Patent Publication: WO2000045605A1. Bit Rate Control for Video Data Compression. Priority Date: January 28, 2000.