LAPSE: Low-Overhead Adaptive Power Saving and Contrast Enhancement for OLEDs

LAPSE: Low-Overhead Adaptive Power Saving and Contrast Enhancement for OLEDs

Daniele Jahier Pagliari, *Student Member, IEEE,* Enrico Macii, *Fellow, IEEE,* and Massimo Poncino, *Fellow, IEEE*

***Abstract*—Organic Light Emitting Diode (OLED) display panels are becoming increasingly popular especially in mobile devices; one of the key characteristics of these panels is that their power consumption strongly depends on the displayed image. In this paper we propose LAPSE, a new methodology to concurrently reduce the energy consumed by an OLED display and enhance the contrast of the displayed image, that relies on image-specific pixel-by-pixel transformations. Unlike previous approaches, LAPSE focuses specifically on reducing the overheads required to implement the transformation at runtime. To this end, we propose a transformation that can be executed in real time, either in software, with low time overhead, or in a hardware accelerator with a small area and low energy budget.**

Despite the significant reduction in complexity, we obtain comparable results to those achieved with more complex approaches in terms of power saving and image quality. Moreover, our method allows to easily explore the full quality-versus-power tradeoff by acting on a few basic parameters; thus, it enables the runtime selection among multiple display quality settings, according to the status of the system.

I. Introduction

Organic Light Emitting Diode (OLED) displays are an increasingly used alternative to classic TFT LCDs in mobile devices [1]. OLED technology owes its popularity to a number of advantages compared to LCD, such as higher brightness and better viewing angles, as well as the possibility of building thinner and flexible screens [2].

The most peculiar feature of OLED displays is that they are composed of *emissive* devices, and therefore do not require an external light source. This has the important consequence of making OLEDs power consumption strongly *image-dependent*. While generally more efficient than traditional LCDs, OLEDs consume significantly more power for bright images [1], [2]. Thus, since the display subsystem is a significant contributor to the total consumption of a mobile device [3], [4], a proper management of OLED energy consumption becomes fundamental.

This need for efficiency has spurred a wide body of methods for reducing consumption in OLEDs, most of which exploit the image dependency of power, trading off the "quality" of the displayed image and the achievable saving [4]–[13], [17], [18]. Some of these methods, being specifically targeted at Graphical User Interfaces (GUIs), are not applicable to other use-cases, such as displaying pictures or videos [4]–[7]. Others require modifications of the analog hardware of the panel, and therefore are not suitable for off-the-shelf displays [8], [9]. Lastly, some techniques are appropriate for general images, and rely only on altering the display content to reduce power [10]–[13]. Specifically, methods in the last group transform an input image I_i into an output image $I_o = T(I_i)$, so that the power consumption of the latter is reduced, while perceived visual quality is preserved as much as possible. The parameters of the transformation function T are generally image-dependent, so that the amount of power reduction and quality loss can be tailored to the displayed image. However, this implies that the parameters must be recomputed in *real time*, every time a new image is stored in the Frame Buffer (FB) of the display. Typical display refresh rates are in the order of 50-60 Hz, thus the computation of parameters and the application of T must be repeated every 15-20 ms. Moreover, OLEDs also allow faster response times, possibly making timing constraints even tighter in future scenarios [1].

Most approaches in literature pay very little attention to the analysis of time and energy overheads. Indeed, they propose image transformations that involve computationally intensive operations, such as the solution of nonlinear optimization problems [10], [11], or complex histogram processing [12], [13]. Implementing these operations in software could consume a significant percentage of CPU time, eating up processing power for other tasks. Alternatively, dedicated hardware could be used, but in both cases the energy overhead for the computation of parameters and evaluation of T could drastically reduce the achievable savings. The only solutions that tackle overheads reduction do it relying on a custom camera application [17], [18], and cannot be applied to images obtained from different sources.

This paper extends our previous work of [19], in which we proposed an alternative low-overhead technique for OLED displays energy optimization. Our method is specifically designed to favor hardware acceleration, with very small energy costs. This is achieved thanks to two key factors: (i) the use of a "hardware-friendly" transformation function T and (ii) the offloading of part of the computational burden to a *training* phase, performed offline. A high-level scheme of the proposed framework is shown in Figure 1. The computationally intensive *Offline Phase* produces a fitting model that puts in relation simple features of an image (namely the mean μ and variance σ^2 of its luminance) with the corresponding optimal parameters ($\mathbf{a}$) of the transformation function T. This information is then used to reduce the complexity of the *Online Phase*, which becomes *linear* in the size of the panel. In this phase, μ and σ^2 are computed for the target image, and the parameters of T are derived according to the aforementioned fitting model.

D. Jahier Pagliari, E. Macii and M. Poncino are with the Department of Computer and Control Engineering, Politecnico di Torino, Turin, IT.
E-mail: {daniele.jahier,enrico.macii,massimo.poncino}@polito.it

Manuscript received January XX, XXXX; revised January XX, XXXX.

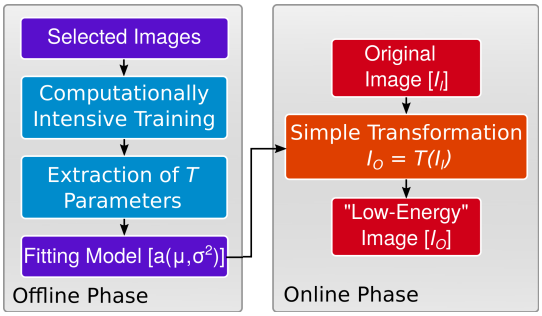


Fig. 1. Flow of the proposed framework.

Finally, the actual low-complexity transformation is applied to produce a low-energy output image.

The following are the main contributions of this paper with respect to the work of [19]:

- We describe the proposed low-overhead framework for OLED energy optimization in greater detail, and we discuss the details of its hardware implementation, at the level of individual components.
- We analyze alternative low-cost image transformations and compare them to our proposed solution. We also consider a new "universal" LAPSE variant.
- We present the results of a subjective test, showing that our method is distinctly superior to simple *brightness scaling* in terms of perceived image quality at the same power level.
- We demonstrate that our method can obtain results comparable to those of more complex techniques (e.g. [10]), in terms of power saving and image quality.
- We show the applicability of our transformation to video sequences, and its ability to closely match the desired quality level throughout sequences of frames.
- Finally, we thoroughly analyze the time and power overheads to implement the image transformation at runtime in either SW or HW.

The rest of the paper is organized as follows. Section II, recaps previous works on display energy consumption optimization. Section III is devoted to the theoretical foundation for our proposed method. Section IV and Section V detail the offline and online phases of the flow introduced in Figure 1, while Section VI presents an even lower-overhead alternative to the proposed transformation. In Section VII we present experimental results, and Section VIII concludes the paper.

II. Background

In classic LCDs, the main contributor to power consumption is the *backlight*, which is either a Cold Cathode Fluorescent Lamp (CCFL) or an array of Light Emitting Diodes (LEDs). Previous studies have shown that it can account for up to 80% of the total display subsystem power [20]. Intensities of pixels in the displayed image have limited impact on LCD power consumption. Consequently, most energy reduction techniques for LCDs rely on some form of *backlight scaling* [20]–[28]. Image transformations are commonly used, but not directly to reduce power. Instead, their purpose is to compensate the brightness reduction due to backlight scaling, so that the perceived image quality is preserved.

In OLEDs, conversely, there is no backlight, and light is directly produced by pixels. Each pixel is formed by three *emissive* devices, corresponding to the components of the RGB color space [4]. The intensity of a component depends on the current flowing through the corresponding device. Therefore, the total power consumption of the panel is therefore strongly dependent on the *brightness* of the displayed image, and on the balance between colors. Measurements on real panels in [4] allowed to build an empirical model for the power consumption of an OLED:

$$P_{tot} = \sum_{i=0}^{W}\sum_{j=0}^{H}(w_0 + w_r \cdot R_{i,j}^{\gamma} + w_g \cdot G_{i,j}^{\gamma} + w_b \cdot B_{i,j}^{\gamma}) \quad (1)$$

where W and H are the width and height of the panel, $(R_{i,j}, G_{i,j}, B_{i,j})$ are the sRGB components of the pixel at position (i, j), and w_x and γ are panel-dependent coefficients, obtained via characterization. For most displays, $\gamma \in [2:3]$.

Power optimization techniques for OLEDs are tailored to this image-dependent power model, and can be broadly split in two categories: those targeting GUIs, and those applicable to general images.

Techniques in the first group are based on the observation that, for GUIs, *usability* is more important than visual *fidelity*. For example, he authors of [4] and [5] propose algorithms that drastically change GUI colors to exploit the color dependency of power consumption. In [6] and [7], power reduction is obtained selectively dimming pixels that are outside of the *area of user interest*, i.e. the section of the screen on which a user is expected to focus his/her attention. All these approaches are not applicable to general images or videos, where fidelity is a fundamental aspect of the perceived quality.

An interesting solution for general images, proposed in [8], consists in applying Dynamic Voltage Scaling (DVS) to the OLED panel to reduce power consumption. Voltage scaling is obtained via a custom driver circuit, which limits the maximum brightness emitted by pixels. Similarly to LCD backlight scaling, this is then compensated acting on pixel values. The same principle is applied in [9] at *fine granularity*, i.e. using different supply voltages for different areas of the panel. Although these techniques are effective, they require custom analog drivers and control circuits, and are not applicable to off-the-shelf OLED displays that do not support DVS.

To overcome this limitation, several techniques have been proposed that act *only* on pixel values via an image transformation T, (see Section I). A seminal approach in this category is Power Constrained Contrast Enhancement (PCCE), first proposed in [10]. This algorithm obtains an image-dependent pixel intensity mapping that concurrently reduces the power consumption of the target image while enhancing its contrast to preserve visual quality. A variant of PCCE based on multiscale retinex is described in [11], while [12] shows that similar results can be obtained without an iterative optimization procedure. Finally, [13] combines Histogram Shrinking with contrast enhancement to achieve the same goal.

While effective, these methods suffer from two major drawbacks. Firstly, they do not allow to set a *hard constraint* on the maximum image "alteration" allowed. Thus, they may yield unpredictable results, and produce too dramatic modifications of the target image. Secondly, and most importantly, the computation of transformation parameters and its subsequent application involve very complex operations. For instance, in [10] a nonlinear optimization problem has to be solved in order to identify the optimal transformation for a given image. Although this is avoided in [12], the proposed alternative still requires several loops involving divisions and fractional powers. The execution times of these approaches, when reported in the papers, are normally not acceptable for a real-time application on high-definition panels. Moreover, these times always refer to a software implementation on a high-end CPU, which is likely to consume a large amount of energy, offsetting or even nullifying the savings on the display.

More recent works in [14], [15] focus on image-adaptive forms of *brightness scaling* for OLED displays. Although effective, these solutions do not propose ways to compensate for brightness reduction (e.g. by contrast enhancement), and therefore are not comparable with our method. Moreover, for both techniques, the reported execution times for transforming still images are in the order of $1 - 2\mu s$ per pixel, which corresponds to $> 1s$ per HD image. Thus, these methods are not suitable for real-time application in a power-constrained computational environment (e.g. a mobile device). A very effective approach for speeding up a "PCCE-like" transformation is proposed in [16]. However, this solution only works assuming that $\gamma = 2$ in the power model of equation (1). In contrast, our technique is applicable regardless of the specific value of this parameter, and in particular when γ is not integer, i.e. the most general and realistic case [4].

Two solutions that directly tackle the overhead problem are presented in [17] and [18]. In both cases, the majority of the computational effort is moved to the *image acquisition* phase, exploiting a custom "camera" application in a mobile device. In [17] the image is transformed directly during acquisition, while in [18] the acquisition phase is used to analyze the image and store metadata, which are then used at display-time to optimize power and quality. The advantage of these solutions is that image acquisition does not have the same tight performance constraints of the display. The obvious disadvantage, however, is that images need to be acquired through the specific camera application proposed by the authors. Hence, these methodologies do not apply to synthetic images (e.g., GUIs, computer graphics, etc.) as well as photos obtained from different sources (e.g., downloaded from the Web).

III. Theoretical Foundation

In this paper we present LAPSE (*Low-overhead Adaptive Power Saving and contrast Enhancement*), a new image transformation for power reduction in OLEDs. LAPSE yields power and visual quality results comparable to those of previous solutions [10]–[13], yet significantly reducing the computational complexity of the transformation.

This section presents the basic theoretical foundation of LAPSE, and motivates the main choices done in the following, most importantly the choice of the transformation function T. In Sections IV and Sections V we will detail the offline and online phases of the flow of Figure 1, respectively.

A. Transforming Luminance

Regardless of the involved optimizations, most of the previous image transformations for OLED power reduction [10]–[13] eventually produce a *pixel-by-pixel* intensity mapping. The latter normally involves only the *luminance* component of each pixel, while *chrominance* components are left untouched. This choice is motivated by the fact that, especially for photos and videos, color alteration dramatically affect the perceived visual quality [10]–[12]. In summary, the image transformation function reduces to:

$$Y_{out,i,j} = T(Y_{i,j}) \quad \forall\ 0 \leq i < W, 0 \leq j < H \tag{2}$$

where $Y_{i,j}$ and $Y_{out,i,j}$ are respectively the input and output luminance components of the pixel in position (i, j). LAPSE also follows this approach. The main difference with previous literature is the selection of the shape of T.

Since most systems internally store images in RGB, pixels must be converted to YCbCr color space (where the Y component represents luminance) before applying the transformation [29]. Therefore, the complete image transformation is composed of three steps: (i) color space conversion from RGB to YCbCr, (ii) application of an intensity mapping to the first component of each YCbCr pixel (iii) conversion of the output image back to RGB. The two color space conversions are obtained through linear operations:

$$[Y_{i,j}, Cb_{i,j}, Cr_{i,j}]^T = \mathbf{C} \cdot [R_{i,j}, G_{i,j}, B_{i,j}]^T + [0, 128, 128]^T \tag{3a}$$

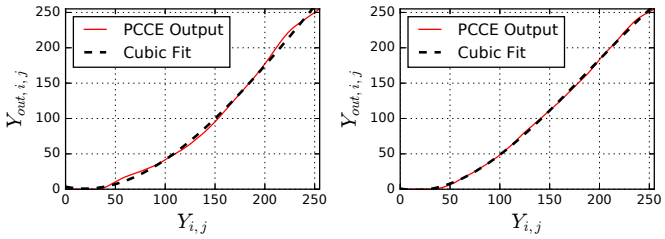
$$[R_{i,j}, G_{i,j}, B_{i,j}]^T = \mathbf{K} \cdot [Y_{i,j}, Cb_{i,j}, Cr_{i,j}]^T - [0, 128, 128]^T \tag{3b}$$

where superscript T indicates the vector transpose, and $\mathbf{C}, \mathbf{K} \in \mathbb{R}^{3x3}$ are matrices of constant coefficients. Using the JPEG standard for YCbCr conversion [31], both RGB and YCbCr pixels are represented on 24-bit, with each component spanning the range $[0, 255]$.

B. Choice of the Generic Transformation Function

Power reduction in OLED displays can be trivially achieved by decreasing pixels luminance: in fact, from (1) and (3b), it follows that $P(Y_{i,j}) \propto Y_{i,j}^{\gamma}$. However, pure "scaling" of luminance degrades significantly the perceived quality. Therefore, as mentioned in Section II, most literature solutions combine a reduction of the total luminance with *contrast enhancement* [10]–[13], [32]. The shapes of the pixel intensity mappings produced by these solutions share some common characteristics. In order to enhance contrast, they are generally *non linear*. Additionally, in most cases, they have *non-monotonic concavity*, in order to alter the contrast of dark and bright areas of the image differently [10], [13].

Since the primary goal of our approach is overhead containment, the chosen form of T must be computable with simple operations. One obvious choice is to use *polynomials*, which only require additions and multiplications, i.e. some of the



(a) Lena (b) Monarch

Fig. 2. PCCE output with $\beta = 1$, for two example images, and corresponding cubic fittings. $RMSE = 1.6\%$ and 0.6% respectively.

TABLE I
CUBIC FITTING OF PCCE PIXEL TRANSFORMATIONS, AGGREGATE RESULTS FOR THE DATASETS DESCRIBED IN SECTION VII.

Data	R^2	RMSE [%]
Live	0.999 ± 0.001	0.858 ± 0.376
BSDS	0.997 ± 0.003	1.464 ± 0.907

least expensive operations to implement in HW. In general, the lower the order of a polynomial, the less operations are required to evaluate it. Therefore, in an attempt to minimize overheads, the ideal choice would be to use a third order polynomial, which is the *lowest order transformation* that can take shapes similar to those observed in previous solutions, since it can have varying concavity.

Figure 2 shows two motivating examples for the choice of this pixel transformation. Red solid curves represent luminance mappings obtained by the PCCE algorithm of [10] for the famous *Lena* and *Monarch* images. Black dashed lines are the best third order polynomial fittings of the PCCE output. As shown, cubic curves can approach PCCE output transformations very closely for both images. For a more quantitative estimate of this similarity, we have determined the best cubic fitting of the PCCE output when applied to each of the images in the two datasets used in our experiments of Section VII. Table I reports the average R^2 score and Root Mean Squared Error (RMSE) obtained for each dataset, and the corresponding standard deviations. The small error values and high scores confirm that, despite its simplicity, a cubic polynomial is effective in producing a very similar mapping to previous power reduction and contrast enhancement algorithms.

Given this analysis, in our method, we transform the luminance of each pixel according to the following equation:

$$Y_{out,i,j} = T(Y_{i,j}) = a_3 Y_{i,j}^3 + a_2 Y_{i,j}^2 + a_1 Y_{i,j} + a_0 \qquad (4)$$

The coefficients a_x in (4) are set to a numeric value that minimizes power while enhancing contrast, and are adapted to the target image, as described in Section IV.

Even in its most general form, (4) only requires 6 multiplications and 4 additions per pixel. However, the degrees of freedom for the general expression of T can be further reduced imposing some additional constraints, similar to those of [10]. Specificallly, we constrain the transformation T to:

1) **Have an output that spans the entire range of luminance intensities, i.e. $[0 : 255]$.** Since one of the goals of the transformation is to enhance contrast, it is desirable

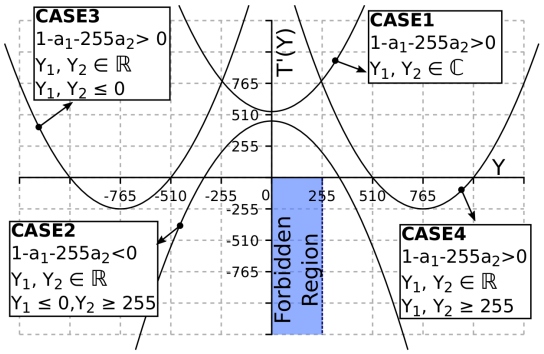


Fig. 3. Examples of first-order derivatives that ensure a monotonically increasing transformation.

to have it span the full range of luminance values, so that the image *dynamic range* is preserved. This is achieved imposing:

$$T(Y_{min}) = Y_{min} \qquad T(Y_{max}) = Y_{max} \qquad (5)$$

where Y_{min} and Y_{max} are the minimum and maximum luminance values, i.e. $Y_{min} = 0$ and $Y_{max} = 255$ in YCbCr. Hence substituting (5) into (4) yields:

$$a_0 = 0 \qquad a_3 = \frac{1 - a_1 - 255a_2}{255^2} \qquad (6)$$

$$Y_{out,i,j} = \frac{1 - a_1 - 255a_2}{255^2} Y_{i,j}^3 + a_2 Y_{i,j}^2 + a_1 Y_{i,j} \qquad (7)$$

2) **Be monotonically increasing in $[0 : 255]$.** This contraint is to avoid the creation of artifacts due to inversions of luminance relations between the input and output images [10]. It allows to set a relation between the two remaining coefficients a_1 and a_2, and defines the search space for the optimal (image dependent) transformation parameters. Monotonicity is imposed forcing the derivative of T to be non negative for the entire luminance range:

$$T'(Y_{i,j}) = 3\frac{1 - a_1 - 255a_2}{255^2} Y_{i,j}^2 + 2a_2 Y_{i,j} + a_1 \geq 0, \quad \forall Y_{i,j} \in [0, 255] \qquad (8)$$

Geometrically, $T'(Y_{i,j})$ is represented by a parabola. The *Forbidden Region*, i.e. the region of plane that the parabola must not intersect in order to meet the constraint expressed by (8), is shown in Figure 3. The figure also reports one example for each of the four (infinite) families of parabolas that meet the constraint, together with the corresponding mathematical relations on a_1 and a_2. The four families differ in the orientation of the curves, determined by the sign of the coefficient that multiplies $Y_{i,j}^2$, and in the value of the two solutions of $T'(Y_{i,j}) = 0$.

Parabolas with upward orientation (i.e. tending to $+\infty$ for $Y \to \pm\infty$) assume negative values when Y is between Y_1 and Y_2. In particular, parabolas with upward orientation and no real solutions are *always* positive, hence they meet the constraint. This is represented by the example labeled CASE1 in Figure 3. Similarly, if there are real solutions, but they are both smaller than 0 or greater than 255, the negative part of the curve will not intersect the Forbidden Region. Examples of these two categories are shown as CASE3 and CASE4. These groups

also include the limit cases of two coincident solutions. In the case of downward orientation, conversely, the curve is *positive* for $Y \in [Y_1, Y_2]$. Therefore, to meet (8), we must have $Y_1 \leq 0$ and $Y_2 \geq 255$, as in the example labeled CASE2.

These four subsets can be further reduced to two based on the following observation: the inflection point of $T(Y_{i,j})$ i.e. the point in which the cubic pixel transformation changes concavity, corresponds to the stationary point (minimum or maximum) of $T'(Y_{i,j})$. For CASE3 and CASE4, this point *always* occurs outside of the $[0, 255]$ interval. Hence, using coefficients from these two cases would significantly reduce the flexibility of the transformation. Additionally, the relations between a_1 and a_2 that can be derived in CASE3 and CASE4 define unbounded regions (semi-planes). Thus, they do not permit the identification of a finite domain for the search of the optimal coefficient values.

For these reasons, we discard CASE3 and CASE4, and limit our analysis to CASE1 and CASE2. Explicit relations between a_1 and a_2 in the latter two cases are obtained simply solving the two systems of inequalities reported in the labels of Figure 3. The results are two bounded plane regions:

$$R_{CASE1} : \begin{cases} 0 \leq a_1 \leq 2 \\ \frac{1-a_1}{255} < a_2 < \frac{3-2a_1}{255} \end{cases} \tag{9a}$$

$$R_{CASE2} : \begin{cases} 0 \leq a_1 \leq 4 \\ \frac{-3a_1-\sqrt{3(4a_1-a_1^2)}}{2 \cdot 255} \leq a_2 \leq \frac{-3a_1+\sqrt{3(4a_1-a_1^2)}}{2 \cdot 255} \end{cases} \tag{9b}$$

The union of R_{CASE1} and R_{CASE2} is the largest bounded area of $\mathbb{R}^2$ for which T is monotonically increasing in $[0 : 255]$, and defines the search domain used in the following to determine the optimal transformation coefficients.

Notice that the theoretical formulation above, although presented for the case of 24-bit RGB/YCbCR is mostly *independent on the chosen pixel representation format*. In principle, all operations in (4)-(9b) are executed on real numbers, and therefore do not rely on a specific data format. For example, the methodology could be extended also to process high dynamic range images (e.g. in LogLuv or RGBE format) [30]. The only modification needed in this case would be in the rightmost constraint of (6), where the value 255 should be replaced with the maximum luminance value of the corresponding representation. Clearly, changing the format *will* have an impact on the complexity of a practical implementation of our method, and especially on the hardware acceleration described in Section V-A. However, since this is the most common format in energy-constrained portable devices, we focus only on 24-bit RGB in the following.

IV. Offline Phase

A. Training Algorithm

In Section III-B we have determined the mathematical form of the generic image transformation T, and the search domain for its free parameters a_1 and a_2. The offline phase of LAPSE consists in the optimization of the value of these two parameters for a set of *training* images. The information gathered during training is then used to identify a relation among the optimal values of a_1 and a_2 and simple quantitative features of the target image.

Parameters are optimized according to the cost function:

$$F(I) = w_p \cdot P_{tot}(I) - \sigma(Luminance(I)) \tag{10}$$

In this equation, I represents the image pixel matrix and P_{tot} is its total power consumption on an OLED panel, obtained according to the model of (1). $Luminance(I)$ corresponds to the operation of extracting the luminance component of pixels, e.g. from RGB using (3a), σ is the standard deviation of the luminance matrix, and w_p is a weighting factor. This function tries to concurrently *minimize power consumption* (P_{tot}) and *maximize contrast*, of which σ is a simple measure, thanks to the minus sign. The two goals are balanced by means of w_p.

Moreover, we also constrain our optimization, limiting the amount of *image alteration* that can be introduced by the transformation. This constraint, not present in previous solutions [10]–[13], controls the quality impact of our method. To measure alteration we leverage the popular Mean Structural Similarity Index (MSSIM) [33]: during training, we only consider solutions for which $\text{MSSIM}(\text{I}, \text{I}_\text{t}) \geq \text{MSSIM}_\text{min}$, where I and I_t are the input and transformed images, and MSSIM_min is a user-imposed threshold.

```
1:  procedure TRAINING
2:      for I ∈ Training Images do
3:          (Y, Cb, Cr) = YCbCrComponents(I)
4:          a_opt = [1, 0]
5:          F_opt = F(I)
6:          for a ∈ R_CASE1 ∪ R_CASE2 do
7:              Y_t = T(Y, a)
8:              I_t = Image(Y_t, Cb, Cr)
9:              if F(I_t) < F_opt and P_tot(I_t) ≤ P_tot(I) and
10:             MSSIM(I,I_t) ≥ MSSIM_min then
11:                 F_opt = F(I_t)
12:                 a_opt = a
13:             end if
14:         end for
15:         Store [a_opt, μ(Y), σ(Y)] in a database.
16:     end for
17: end procedure
```

Fig. 4. Training phase algorithm.

The pseudocode for the training phase is reported in Figure 4. For clarity, the two transformation parameters have been grouped to form a vector $\mathbf{a} = [a_1,\ a_2]$. Matrices $\mathbf{Y}$, $\mathbf{Cb}$ and $\mathbf{Cr}$ indicate the three YCbCr components of image I, while the subroutines YCbCrComponents() and Image() encompass color space conversions and components separation/grouping. F and F_opt are the current and optimal values of the objective function, and w_p is a weighting coefficient used to balance the impact of power and contrast in F. The remaining symbols are defined as in previous sections. The additional constraint $P_{tot}(I_t) < P_{tot}(I)$ in line 9 ensures that the transformed image consumes less than the original. It is inserted because power reduction is the primary goal of LAPSE, and is always favored over pure contrast enhancement.

The initial optimal parameter values $\mathbf{a_{opt}} = [1,\ 0]$ correspond to the identity transformation, i.e. $T(\mathbf{Y}) = \mathbf{Y}$, as evident from (7). The search space for $\mathbf{a_{opt}}$ is explored exhaustively, with a granularity that depends on the desired accuracy. In our experiments, we considered a grid of 40000

points. Exhaustive search is necessary for global optimality, since both the objective function and the constraints are not convex [34]. However, the resulting computational effort is acceptable, given the relatively limited search space and that training is performed only once, offline.

B. Linear fitting of Transformation Coefficients

To obtain an adaptive transformation that is flexible (i.e. image-adaptive) but simple enough for online usage, coefficients of T have to be put in relation with elementary features of the image being processed. Since the transformation affects the Y component of pixels, the features that we select are also luminance-related. The two simplest choices are the average *brightness* and *contrast*, measured by the luminance mean (μ) and standard deviation (σ) respectively [33]. However, in order to simplify the hardware implementation of the transformation, as explained in Section V, we substitute σ with σ^2.

We use a *linear regression* model to put in relation $\mathbf{a}$ with μ and σ^2. This model is trained with the outputs of the previous optimization phase (Section IV-A), using a Least Squares cost function, to produce coefficients $\mathbf{P} \in \mathbb{R}^{2x2}$ and $\mathbf{q} \in \mathbb{R}^{2x1}$, that identify the regression plane:

$$\mathbf{a^T} = \mathbf{P} \cdot \begin{bmatrix} \mu \\ \sigma^2 \end{bmatrix} + \mathbf{q} \tag{11}$$

$\mathbf{P}$ and $\mathbf{q}$ can then be used to determine the transformation parameters for new images.

The values of a_1 and a_2 used as training data, and consequently the values of $\mathbf{P}$ and $\mathbf{q}$, depend on the $\text{MSSIM}_{\text{min}}$ constraint set in the algorithm of Figure 4. Therefore, when estimating a_1 and a_2 for a new image, the regression model will output parameters that, apart from the estimation error, generate a transformation that meets the same quality constraint. Repeating training with different MSSIM constraints yields different fitting coefficients. For example, if training is performed setting first $\text{MSSIM}_{\text{min}} = 0.95$ and then $\text{MSSIM}_{\text{min}} = 0.90$, it will produce coefficients $(\mathbf{P_{0.95}}, \mathbf{q_{0.95}})$ and $(\mathbf{P_{0.90}}, \mathbf{q_{0.90}})$ respectively. When these are used for new images, the latter will produce larger power reductions in the display, but will also cause more visible image alterations.

Notice that, while model *training* is performed offline, its *evaluation* according to (11) must be executed online, in real time. Hence, the choice of a linear model was dictated by its low evaluation complexity. As shown in Section VII this simple model is sufficient to obtain transformations that are very similar to the optimal ones, from both visual inspection and quantitative analysis.

V. Online Phase

Having determined the fitting coefficients that link the basic features of an image (μ, σ^2) to the optimal values of $\mathbf{a}$, we can adapt the pixel-by-pixel transformation expressed by (7) to the content of the OLED display at runtime. A flow diagram summarizing the online part of LAPSE is shown in Figure 5a. Globally, these operations take an input image I_i in RGB format and transform it into an energy-efficient output image I_o, in the same format. Internally, the flow can be subdivided into three main phases. The complexity of each phase in terms of number of required operations, with respect to the size of the OLED panel, is reported in the figure.

TABLE II
Number of additions/subtractions and multiplications required in each part of the online transformation.

Phase	Add/Sub	Mul
RGB to YCbCr	$6WH$	$9WH$
Compute μ and σ^2	$2WH + 1$	$WH + 3$
Compute $\mathbf{a_1}$ and $\mathbf{a_2}$	4	2
Apply T()	$4WH$	$5WH$
YCbCr to RGB	$6WH$	$4WH$

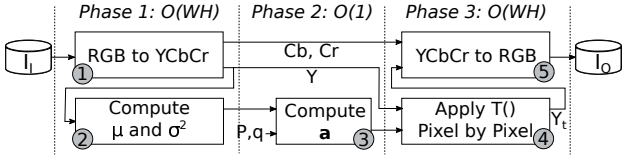
The first part, labeled *Phase 1*, consists in the conversion from RGB to YCbCr space and in the computation of the average luminance and contrast of the input image. Both these tasks need to execute operations on every pixel, hence the complexity is linear, i.e. $O(WH)$. Notice, however, that the two tasks can be completely *overlapped*: as soon as one pixel has been converted to YCbCr space, its luminance can immediately be used for the calculation of μ and σ^2.

The section labeled *Phase 2* consists in the evaluation of the fitting model described in Section IV-B. It takes as input the fitting coefficients $\mathbf{P}$ and $\mathbf{q}$, as well as the values of μ and σ^2, computed in the previous phase. Since the number of transformation parameters is 2 (a_1 and a_2) regardless of the size of the panel, the complexity of this phase is constant.

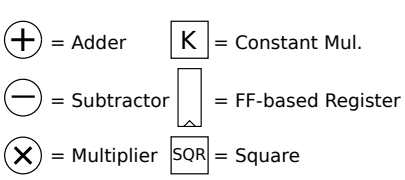
Phase 3 includes the actual application of the transformation T to the luminance component of each pixel, and the conversion of the transformed image to RGB. As for *Phase 1* complexity is linear, since both tasks must be executed once per pixel. Again, the two tasks can be overlapped, applying the color space conversion to a pixel as soon as it has been transformed.

Overall, the scheme of Figure 5a is significantly simpler than those proposed in previous works [10]–[13]. First, as detailed in Section V-A, the only operations involved in the online transformation, apart from some basic control flow, are additions, subtractions and multiplications. Second, very few of such operations are required to process each pixel of the image, and the overall complexity grows linearly with the OLED panel size. Table II shows a breakdown of the number of basic operations required in each block of Figure 5a.

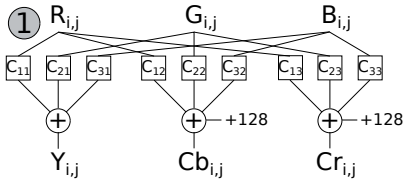
One important aspect of this flow is the possibility of changing coefficients $\mathbf{P}$ and $\mathbf{q}$ at runtime. According to the analysis of Section IV-B, this permits runtime switching of the transformation "quality mode", i.e. the amount of allowed image alteration. For example, this change can be triggered by external conditions (e.g. battery state of charge, ambient illumination), or by a user-driven quality setting. Conversely, previous methods for concurrent power reduction and contrast enhancement do not allow to *directly* set a maximum alteration constraint on their transformations. In [10], [11], [13], different degrees of alteration can be obtained acting on algorithm parameters, but this requires non-trivial tuning. Other works, such as [12], propose transformations that reach a target power saving, without taking alteration into account.



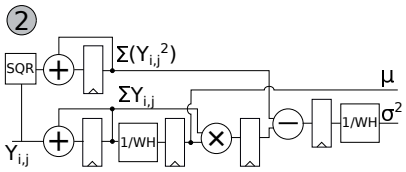
(a) Online phase flow diagram



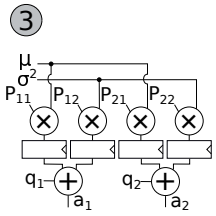
(b) Diagrams Legend



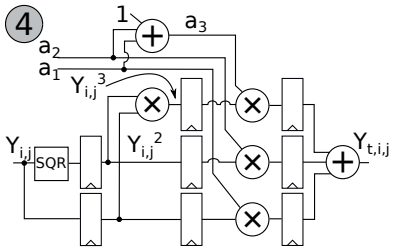
(c) RGB to YCbCr conversion



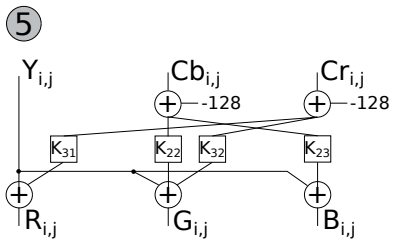
(d) Computation of μ and σ^2



(e) Fitting of a_1 and a_2



(f) Application of transformation T



(g) YCbCr to RGB conversion

Fig. 5. Online phase of LAPSE and hardware implementation.

A. Hardware Implementation Details

While a software implementation of Figure 5a is relatively trivial, the online part of LAPSE also lends itself to hardware acceleration, for better performance and energy efficiency. One possible hardware implementation of the blocks that compose Figure 5a is shown in Figures 5c-5g, whereas Figure 5b contains a legend of the main elementary components used in the different blocks. All circuits work on fixed-point data, and only use addition, multiplication and shift operations. The choice of σ^2 in place of σ as a feature eliminates the need of computing a square root. Moreover, the divisions required to compute μ and σ^2 can be implemented as multiplications via inversion. In fact, the denominator $W \cdot H$ is the size of the panel, which is constant.

Figure 5c implements equation (3a) to convert each image pixel from RGB to YCbCr. Square blocks represent constant multiplications, which do not require full hardware multipliers. Figure 5d shows the circuit used to compute μ and σ^2. The block labeled SQR represent the squaring operation (i.e. $Y_{i,j}^2$). Mean luminance is obtained from the basic sample average formula, whereas for variance we use the expression:

$$\sigma^2 = \frac{(\sum_{i=0,j=0}^{W,H} Y_{i,j}^2) - \mu \cdot \sum_{i=0,j=0}^{W,H} Y_{i,j}}{WH} \qquad (12)$$

The advantage of this factorization with respect to other alternatives is that μ is not included in the summation. Hence, $\sum Y_{i,j}^2$ can be computed in parallel with $\sum Y_{i,j}$, and both μ and σ^2 can be calculated with a *single scan* of all pixels. Cancellation errors typical of floating point implementations of this equation do not occur in fixed point.

Figure 5e shows the hardware for evaluating the fitting model described in Section IV-B. This is the circuit with the largest power consumption, since it includes four multipliers, although with small bit-widths (12x12-bit according to the precision analysis described in the following). However, it is only active for two clock cycles per image. Thus, low-power optimization techniques such as clock and/or power gating can be used to dramatically reduce its energy impact.

The circuit in Figure 5f implements the actual pixel luminance transformation of (7). To this end, parameter a_3 is first extracted from a_1 and a_2 using (6). In this operation, the constant 255 can be replaced with $256 = 2^8$ without a significant impact on the output error. Using a power of two allows to replace multiplication with proper wiring, with zero hardware cost. Thus, the computation of a_3 reduces to a simple addition. This circuit is pipelined, so that one transformed pixel is produced at each clock cycle. A non pipelined implementation can be easily designed to obtain even smaller hardware, loosing in throughput.

Finally, Figure 5g shows the hardware for converting the transformed pixels back to RGB color-space, according to equation (3b). The circuit is similar to that of Figure 5c, with the difference that some constant multiplications can be replaced by proper wirings, thanks to the fact that the corresponding elements of matrix $\mathbf{K}$ are either 1 or 0 [29].

Given that both inputs and final outputs are 8-bit values, most intermediate operations do not need large bit-widths to obtain sufficient accuracy. For example, the constant factors in the multiplications of Figure 5c and 5g must be represented on 9-bit and 10-bit two's complement respectively, with proper ranges, to generate color-space conversions that are accurate up to one intensity unit, with respect to a double precision floating point reference. With similar considerations, it is possible to devise the bit-widths of all other operations. For

the fitting model evaluation, we select bit-widths that make the hardware arithmetic error negligible with respect to the fitting RMSE on a set of test images [36], [37].

Since both RGB and JPEG YCbCr formats represent pixels on 24-bits, no additional memory is required to implement the transformation, except for the intermediate registers shown in the figures. In fact, pixels can be overwritten *in place* after color-space conversions and luminance transformations. The memory used for this purpose can be either the main system memory or the display Frame Buffer (FB) depending on the global system architecture.

It should be remarked that the ones presented are just some of the possible implementations for the tasks of Figure 5a. For instance, most of the operations involved in the transformation lend themselves to parallelization. Hence, vectorized versions of the circuits can be designed, in order to improve throughput, at the cost of an equivalent increase in area and power. On the other hand, architectures that favor hardware reuse (e.g., non-pipelined) are also possible, in order to reduce the number of required adders and multipliers. In general, the optimal architecture depends on system-specific requirements.

VI. Universal LAPSE

An interesting variant of the method described in Sections III-V consists in maintaining the cubic form of T, but making its parameters *independent* of the considered image. We call this variant *universal* LAPSE. In practice, rather than computing the optimal image-dependent parameters of the transformation through fitting, two constant values are assigned to a_1 and a_2 throughout the online phase. These constants are chosen as the *averages* of the a_1 and a_2 values obtained in the training algorithm of Section IV-A.

The advantage of the universal solution is a reduction in the complexity of the online phase. Specifically, the computation of μ and σ, as well as the fitting of a_1 and a_2 (i.e. tasks labeled 2 and 3 in Figure 5a) are not necessary. The obvious drawback is that the transformation is not adapted to the considered image. As a consequence, its effect on visual quality, and especially on the MSSIM between input and output images, will vary significantly from one image to the other. The choice between universal and adaptive LAPSE identifies a tradeoff between quality of results and overhead impact, which will be analyzed further in Section VII.

VII. Experimental Results

A. Setup

In the following experiments, we used the model of equation (1) to estimate the power consumption of an OLED panel. Model coefficients have been set to the same values used in [10], i.e. $(\gamma, w_0, w_r, w_g, w_b) = (2.2, 0, 70, 115, 154)$. To evaluate image similarity, we used the MSSIM [33].

We tested the proposed solution on two publicly available image datasets: the *LIVE* dataset, for visual quality assessment (29 images) [36] and the *Berkeley Segmentation Dataset* (BSDS), originally designed to test image segmentation algorithms (500 images) [37]. Both contain images with a wide variety of subjects, luminance and contrast. For videos, we considered sequences from the *Open Video Project* [38] and from the *Derf's Test Media Collection* [39].

We implemented the offline part of LAPSE in Python 3.5. In the algorithm of Figure 4, we set w_p to:

$$w_p = \frac{255}{W \cdot H[w_0 + (w_r + w_g + w_b)255^{\gamma}]} \tag{13}$$

The denominator of this equation is the maximum power consumed by a panel of size $W \cdot H$, corresponding to a totally white image. Thus, using this value for w_p normalizes power to the $[0:255]$ range, and makes it comparable to the values assumed by the luminance standard deviation $\sigma \in [0 : \frac{255}{2}]$. This guarantees a good balance between power reduction and contrast enhancement during training. The search space for $\mathbf{a_{opt}}$, defined by equations (9a) and (9b) has been divided into a regular grid of 40.000 points for exhaustive search. Smaller discretizations of the search space do not produce a significant difference in the power saving and visual quality achieved during training.

The online part of the transformation has been implemented in both software and hardware. For the former we wrote a single-thread program in C, and compiled it with GCC version 6.3.0 for a high-end x86_64 platform (Intel Xeon E3-12, 8MB L3, Linux Kernel v. 2.6). We measured execution times with the Linux real time clock library, averaging the results of 1000 runs. The hardware version was designed at Register Transfer Level (RTL) using VHDL. It was then synthesized with Synopsys Design Compiler v2016.03, targeting a 45nm CMOS standard-cell library from ST Microelectronics. The clock frequency for synthesis was set to 1GHz for all blocks. The execution of the synthesized hardware was evaluated by means of simulations in Mentor Graphics Questa Sim v10.6. Power consumption was estimated in Synopsys PrimeTime Suite v2016.6, using annotated switching activity from simulations. Since the number of operations required by the online transformation is independent on data, the hardware time is constant for a given panel size.

B. Fitting and universal transformation

In a first experiment, we evaluated the fitting model of Section IV-B, and assessed its accuracy in relating image features with optimal transformation parameters. We also compared it with polynomial models of different order. For this, we used the BSDS dataset [37], which is already conveniently split into training and test sets. We ran the algorithm of Figure 4 on all training images, and used the corresponding values of μ, σ^2, and $\mathbf{a}$ to train the different models. We then evaluated the goodness of each model on test images. As accuracy metric, we used the (normalized) RMSE of a_1 and a_2, and that of the output MSSIM. The latter was computed transforming all test images with fitted a_1 and a_2 and evaluating the MSSIM difference with respect to the image obtained with ideal parameters, directly generated by the algorithm of Figure 4.

Results of this evaluation for the constraint $MSSIM_{\text{min}} = 0.80$ are shown in Figure 6, where the x axis shows the order of the polynomial. The 0 point corresponds to a *constant* polynomial, i.e., a model that estimates a_1 and a_2 as constant

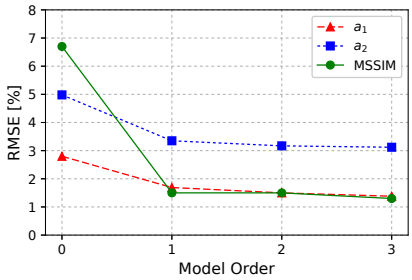


Fig. 6. Fitting RMSE versus fitting model order for BSDS images.

values, using the average of the training set. It is evident that this corresponds to *universal LAPSE* described in Section VI.

These results confirm the validity of the choice of a linear model (order 1) for fitting [1]. In fact, with respect to the constant approximation, all errors drop significantly. In particular, the RMSE on MSSIM decreases from 6.7% to 1.5%. On the other hand, increasing the model order beyond 1, the error on MSSIM remains approximately constant (1.3% for a 3rd order model). Thus, the linear solution represents a knee-point, and is a good tradeoff between evaluation complexity and accuracy. Moreover, the low errors obtained for the different models also demonstrate that the mean and variance of the image luminance are sufficient features to determine close-to-optimal parameters for the transformation T of LAPSE. Although more complex features could provide even better fitting results, they would also require more complex calculations to be extracted, hence increasing the transformation overheads. Finally, notice that the universal solution could be still acceptable, depending on the system requirements. Obviously, it will produce images whose similarity with the input is sometimes significantly lower than the imposed constraint. However, this loss in accuracy must be traded off with the advantages in terms of complexity, detailed in Section VII-H.

C. Subjective evaluation

Although simpler than previous solutions, LAPSE still introduces some overheads for implementing the online part of the transformation. To justify these overheads, the subjective quality of the transformed images must be superior to simple *luminance scaling*, i.e. the standard approach used in mobile devices today. To determine whether this was the case, we performed a subjective evaluation test, using the entire *Live* dataset, together with the first 50 images of the *BSDS* training set. The number of images has been limited in order to gather a sufficient number of votes per image. We compared LAPSE with luminance scaling under the *same power consumption* conditions. To do so, we first transformed all benchmark images with LAPSE, setting the minimum MSSIM constraint to 0.80. We then computed the power saving obtained for each image. Finally, we applied luminance scaling to the input images, setting the scaling factor k to a value that produces the *same* power saving as LAPSE. This value can be obtained numerically using a bisection method. The luminance scaling transformation that we implemented is the same adopted in the work of [15]. The generated pairs of images were shown to a pool of 169 subjects, not expert in image or video processing. Each subject was asked to evaluate 10 pairs of images, and for each pair, select the version that he or she preferred. Results are shown in Table III.

[1]The order of the fitting model (linear) should not be confused with the order of the polynomial used for transforming pixels (cubic).

TABLE III
SUBJECTIVE EVALUATION RESULTS. NUMBER AND PERCENTAGE OF PREFERENCES FOR EACH DATA SET.

Data	Images	LAPSE	Scaling	Draw
Live	29	24 (82.8%)	5 (17.2%)	0 (0.0%)
BSDS	50	38 (76.0%)	11 (22.0%)	1 (2.0%)
Total	79	62 (78.5%)	16 (20.3%)	1 (1.3%)

Columns "LAPSE" and "Scaling" report the number of images for which the proposed approach was preferred over luminance scaling by the voters, and vice versa. The last column report draws. This result confirms that, for the large majority of images, LAPSE produces subjectively better outputs compared to luminance scaling, at the same power level.

D. Comparison with PCCE

PCCE, presented in [10], is one of the most popular image transformations for OLED power reduction and contrast enhancement. Although PCCE is significantly more complex than LAPSE, it achieves a similar objective, hence we chose this technique as state-of-the-art reference for comparison. Notice that the goal of LAPSE is to produce images that are visually and quantitatively *comparable to* (rather than *better than*) those generated by more complex transformations; our technique does not aim at improving the state-of-the-art in terms of sheer output quality, but at dramatically reducing the transformation complexity. Therefore, although there are countless transformations for power saving and contrast enhancement in literature [10]–[13], some more recent than PCCE, we limited our comparison in terms of image quality to one representative and well-recognized approach. In Section VII-G, instead, we considered one of these most recent methods, together with PCCE, when comparing the performance of LAPSE in terms of software execution time.

Unlike the case of brightness scaling (Section VII-C), it is not possible to compare the two methods for equal power consumption. In fact, both PCCE and LAPSE do not allow to fix the desired power saving. Rather, PCCE performs an unconstrained optimization of power and contrast, only controlled through the parameter β, whereas in our solution, power is minimized under a MSSIM constraint. Thus, we compared the two methods in *same similarity* (MSSIM) conditions, and evaluated whether they obtain similar power savings and image enhancement scores. To do so, we applied PCCE to all the images in the *BSDS* and *Live* datasets. We then computed the MSSIM between each original and transformed image, and set that value as a constraint in our framework. As metric of image enhancement we used the popular EME [40]. Finally, we also computed the Peak Signal-to-Noise Ratio (PSNR) between input images and LAPSE or PCCE outputs, to analyze

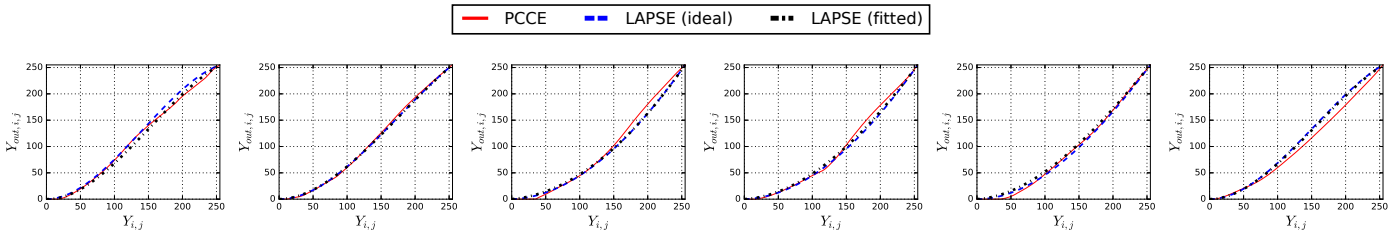
(a) Original

(b) PCCE ($\beta = 1$)

(c) LAPSE (ideal parameters $\mathbf{a}$)

(d) LAPSE (fitted parameters $\mathbf{a}$)

Fig. 7. Comparison between PCCE [10] and the proposed LAPSE algorithm (images).



(b) Cat (c) Boat (d) Fish (e) Lighthouse (f) Parrots (g) Bikes

Fig. 8. Comparison between PCCE [10] and the proposed LAPSE algorithm (luminance transformation curves).

TABLE IV
COMPARISON WITH PCCE, DETAILED RESULTS.

Data	Power Saving [%]			EME			MSSIM			PSNR		
	PCCE	LAPSE Ideal	LAPSE Fitted	PCCE	LAPSE Ideal	LAPSE Fitted	PCCE	LAPSE Ideal	LAPSE Fitted	PCCE	LAPSE Ideal	LAPSE Fitted
Cat	37.1	32.1	36.4	28.5	23.9	24.3	0.812	0.842	0.825	24.14	24.69	24.09
Boat	47.1	51.1	44.0	29.3	25.3	23.0	0.794	0.795	0.805	22.13	22.27	22.41
Fish	52.7	59.8	58.1	12.6	11.3	10.3	0.755	0.756	0.791	19.34	18.80	19.18
Lighthouse	61.0	62.2	60.3	17.2	15.6	14.7	0.764	0.768	0.782	18.64	18.49	18.99
Parrots	51.8	55.1	51.2	14.4	5.9	5.6	0.724	0.725	0.772	19.16	19.08	19.54
Bikes	49.3	42.0	36.4	31.6	33.7	36.1	0.778	0.778	0.788	22.33	22.78	23.51

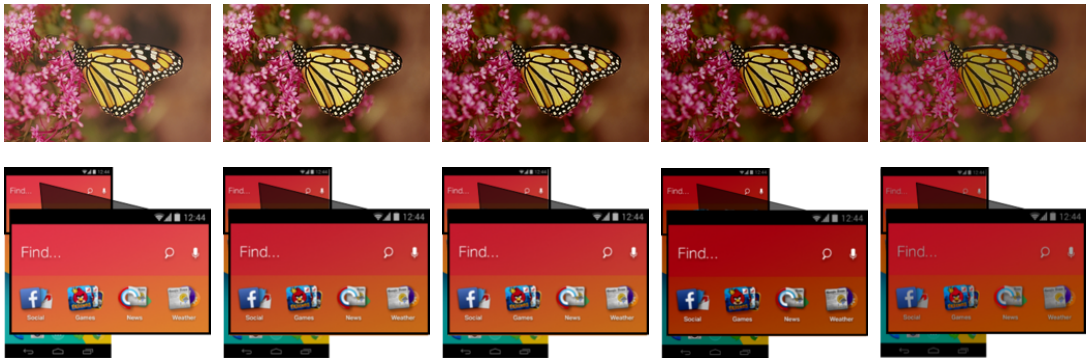
whether the two algorithms produce images with comparable "similarity", even when considering a different metric with respect to the one used as constraint for LAPSE training (i.e. the MSSIM).

Table V shows aggregate results of this analysis. For power savings, EME and PSNR, the table reports average values and standard deviation intervals over the entire datasets. Under the same MSSIM conditions, the two methods achieve almost identical power savings and similarity, although PCCE has less variation over different images. In terms of quantitative enhancement, PCCE is slightly better than our approach, due to its greater flexibility in the shape of the pixel transformation T. It must be remarked, however, that the primary goal of our method not to improve PCCE performance, but rather to achieve comparable results at a much lower overhead cost. Thus, results should be read taking into account the difference in complexity among the two methods.

A more detailed comparison for six example images is shown in Figure 7 and Table IV. In addition to the original images and PCCE outputs, two sets of LAPSE outputs are reported. The first was obtained with *ideal* parameters, i.e. running the algorithm of Figure 4 directly on the target

TABLE V
COMPARISON WITH PCCE, AGGREGATE RESULTS.

Data	Power Saving [%]		EME		PSNR	
	LAPSE	PCCE	LAPSE	PCCE	LAPSE	PCCE
Live	47.5 ± 15.9	49.5 ± 10.9	24 ± 16.1	28.7 ± 15.1	20.45 ± 2.85	20.18 ± 2.21
BSDS	44.2 ± 20.1	44.6 ± 10.6	19.7 ± 14.4	25.9 ± 12.8	21.34 ± 3.93	21.19 ± 3.00



(a) Original (b) $MSSIM = 0.95$ (c) $MSSIM = 0.85$ (d) $MSSIM = 0.75$ (e) Scaling

Fig. 9. Examples of power versus quality tradeoff.

TABLE VI
EXAMPLES OF POWER VERSUS QUALITY TRADEOFF, DETAILED RESULTS.

Target MSSIM	0.95		0.85		0.75	
Image	**Saving [%]**	**Out MSSIM**	**Saving [%]**	**Out MSSIM**	**Saving [%]**	**Out MSSIM**
Butterfly	24.0	0.961	41.3	0.881	57.9	0.765
Android	27.7	0.977	43.7	0.849	54.4	0.777

TABLE VII
POWER VERSUS QUALITY TRADEOFF, AGGREGATE RESULTS.

Target	MSSIM = 0.95	MSSIM = 0.85	MSSIM = 0.75
Data	**Sav. [%]**	**Sav. [%]**	**Sav. [%]**
Live	24.87 ± 5.91	39.38 ± 8.73	49.37 ± 10.41
BSDS (Test)	23.35 ± 7.77	37.51 ± 11.88	47.21 ± 13.84
Data	**Out MSSIM**	**Out MSSIM**	**Out MSSIM**
Live	0.959 ± 0.015	0.855 ± 0.026	0.773 ± 0.038
BSDS (Test)	0.950 ± 0.019	0.855 ± 0.042	0.776 ± 0.051

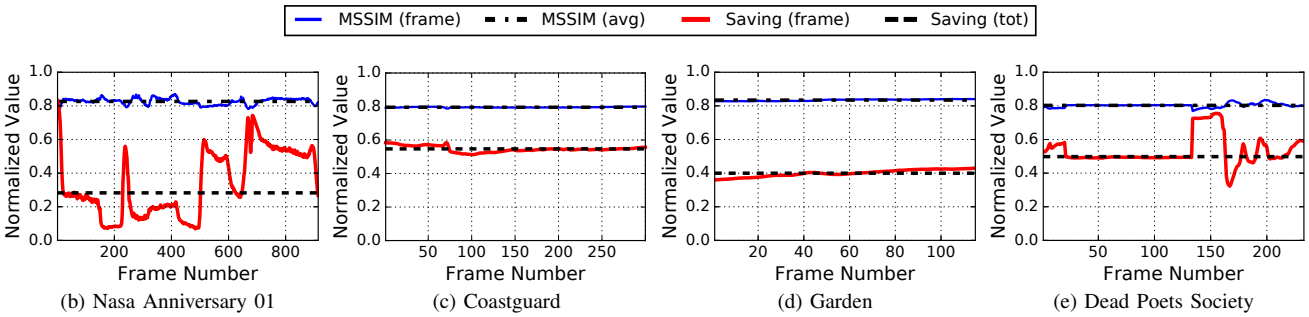
image, setting the MSSIM of the PCCE output as constraint. The second shows the transformation with *fitted* parameters, according to the model of equation (11). These images are affected by fitting error, and are the actual outputs of LAPSE at runtime. Fitting coefficients for each image have been obtained running training with the MSSIM of PCCE as constraint, after removing the image from the training set. To better evaluate the similarity of the two methods, Figure 8 shows the luminance transformation curves obtained by PCCE and the two LAPSE variants for the six images of Figure 7.

These examples highlight the similarity between the outputs of our solution and those of PCCE. Looking at the images, it is very hard to notice any difference in terms of quality and level of detail. This is confirmed also by the plots of Figure 8, which show that the luminance transformations obtained by LAPSE are very close to the ones of PCCE. The occasional differences are due to the higher number of degrees of freedom available to PCCE for shaping the luminance intensity mapping, but, as clear from the figure, they are typically very small.

Moreover, the examples also show that the fitting error does not worsen the performance of our transformation significantly, neither in terms of visual quality nor quantitative metrics. The latter is shown both by the similar values of saving and EME in Table IV, and by the fact that the fitted luminance transformations in Figure 8 are almost exactly superimposed to the ideal ones.

E. Quality versus power tradeoff

LAPSE directly takes a maximum alteration (minimum MSSIM) constraint for its training phase. This allows to easily change "quality-modes" at runtime, as explained in Section V. Notice that none of the previous solutions for concurrent power reduction and contrast enhancement allows to set a similar constraint; some receive as input a target power consumption (e.g. [12]), while some others have parameters specifying the balance among power reduction and contrast increase (e.g. [10]). However, none of them allows to directly limit the alteration on the output image. To show this novel aspect of our framework, we trained LAPSE on BSDS images, setting three different MSSIM constraints: 0.95, 0.85 and 0.75. Then, we calculated regression coefficients for each of these conditions and used them to transform two example images, not included in the training set. Results are shown in Figure 9 and Table VI. The last column of the figure shows the output of a simple



(b) Nasa Anniversary 01 (c) Coastguard (d) Garden (e) Dead Poets Society

Fig. 10. Power saving and MSSIM for video sequences.

brightness scaling, with a scaling factor that guarantees the same power saving as LAPSE in the case $MSSIM = 0.75$. Notice how the contrast and details (e.g. the butterfly wings) are preserved better by LAPSE. For completeness, Table VII shows the aggregate power saving and output MSSIM results when the experiment described above is performed on the entire Live dataset and on the BSDS test set (notice that fitting coefficients were obtained considering the BSDS *training* set only). This table shows that our method obtains relevant savings even for large MSSIM values, and that the fitting model is able to match the target MSSIM very accurately, on average.

The Android screenshot example in Figure 9 highlights the generality of our approach, which yields good results also when applied to GUIs, despite not being expressively designed for them. In particular, the icons and text are much more contrasted and visible in Figure 9d than in Figure 9e, for identical power savings. In this experiment, the transformation has been trained with *natural pictures*, typically characterized by a bell-shaped luminance intensity histogram. The screenshot of Figure 9 has a relatively similar histogram, hence we obtain good results. Clearly, quality will worsen if the input GUI has a strongly *multimodal* histogram, with few distinct intensity values [23].

Even training LAPSE with GUI images, we do not expect to obtain better results compared to approaches specifically targeted at them, such as [4]–[7]. In fact, due to the continuous nature of cubic polynomials, LAPSE affects *ranges* of pixel intensities rather than single values, preserving fidelity and avoiding artifacts. While this might reduce the power saving opportunities on some GUIs, in which most consumption is due to few histogram bars, it makes our strategy generally applicable also to pictures and videos.

F. Performance on video sequences

The low overheads of LAPSE compared to previous solutions allow real-time application to video sequences. Some examples of transformed videos, put side by side with the original input have been uploaded to [41] for visual inspection. The sequences have been transformed using a fitted parameter vector $\mathbf{a}$, trained under a constraint of $MSSIM_{min} = 0.80$. These examples show that, although $\mathbf{a}$ is adapted on a frame by frame basis, this does not generate visible flickering artifacts in the videos. Indeed, $\mathbf{a}$ is a function of the luminance mean and variance of each frame, and frames belonging to the same scene tend to have similar luminance. Thus, abrupt changes of the transformation coefficients only occur in correspondence of scene changes, and do not impact user experience.

Figure 10 shows the trend over time of power saving (normalized to $[0:1]$, where 0 means no saving) and MSSIM for some of the uploaded sequences. Each plot also reports the average MSSIM over the entire sequence, as well as the total power saving i.e., $1 - \frac{P_{in,tot}}{P_{out,tot}}$ where $P_{in,tot}$ and $P_{out,tot}$ are the total power consumptions over all frames, for the original and transformed videos. Notice how the MSSIM stays very close to the target value (0.80) throughout the videos. Instantaneous power saving, instead, can vary significantly over time, as in the case of the first and last videos, which are both characterized by varying luminance. These examples clearly show the effectiveness of LAPSE in limiting the amount of image alteration allowed, and automatically tuning the achievable saving accordingly.

G. Software implementation results

The software execution time of LAPSE, under the conditions described in Section VII-A, is reported in Table VIII for two different image sizes. Each cell reports the average time and the corresponding standard deviation. For comparison, we also implemented the PCCE algorithm [10] and the Noniterative PCCE (NIPCCE) of [12] in C language, and computed their execution times on the same platform. For PCCE, we set the $\beta = 1$, whereas for NIPCCE, we used 8x8 local windows and we set $TPCR = 0.7$, $w_{cl} = 0.05$, and $\gamma = 2.2$. The last two table rows report execution times of color-space conversions alone.

In all three algorithms, we minimized the number of image scans grouping operations that can be performed in the same loop (e.g. RGB to YCbCr conversion, and μ and σ^2 computation in LAPSE). Since these are single-thread programs, operations are still executed sequentially. However, overheads due to control operations and memory accesses are reduced. Moreover, we did not make use of any advanced image processing or mathematical library, nor of GPU acceleration. This choice was made in order to build implementations that (although tested on a high-end processor) could be easily ported to less powerful embedded devices, i.e. the final targets for LAPSE, which might not have these features available.

As shown in Table VIII the execution of LAPSE is dominated by colors-space conversions, regardless of the size of the

TABLE VIII
SOFTWARE EXECUTION TIME RESULTS. PCCE ACRONYMS: HIST = LUMINANCE HISTOGRAM COMPUTATION, LHM = LOG-BASED HISTOGRAM MODIFICATION, PCCE = MAIN OPTIMIZATION LOOP. NIPCCE ACRONYMS: GA = GLOBAL ATTRIBUTE COMPUTATION, LA = LOCAL ATTRIBUTE COMPUTATION, LT = LOCAL TRANSFORMATIONS COMPUTATION, BI = BILINEAR INTERPOLATION, ADJ = FINAL IMAGE ADJUSTMENT.

LAPSE		
Task	**512x512 [ms]**	**1280x1280 [ms]**
RGB-YCbCr and μ, σ^2	2.96 ± 0.35	15.19 ± 0.88
Compute **a**	$(3.23 \pm 5.52)10^{-4}$	$(5.98 \pm 1.56)10^{-4}$
$T()$ and YCbCr-RGB	3.53 ± 0.32	19.49 ± 1.05
Total	6.49 ± 0.49	34.68 ± 1.62
PCCE		
Task	**512x512 [ms]**	**1280x1280 [ms]**
RGB-YCbCr and HIST	2.61 ± 0.35	13.68 ± 0.94
LHM and PCCE	13.3 ± 0.72	13.6 ± 0.76
$T()$ and YCbCr-RGB	3.36 ± 0.31	13.62 ± 0.76
Total	19.29 ± 1.07	45.01 ± 2.48
NIPCCE		
Task	**512x512 [ms]**	**1280x1280 [ms]**
RGB-YCbCr and GA	3.01 ± 0.33	13.96 ± 0.75
LT and LA	5.16 ± 0.29	31.67 ± 1.32
BI	4.58 ± 0.25	30.78 ± 1.11
ADJ and YCbCr-RGB	2.59 ± 0.13	15.48 ± 0.72
Total	15.35 ± 0.89	91.89 ± 3.15
Color Conversions Only		
Task	**512x512 [ms]**	**1280x1280 [ms]**
RGB-YCbCr	2.31 ± 0.37	12.75 ± 0.95
YCbCr-RGB	2.34 ± 0.21	13.94 ± 1.17
Total	4.67 ± 0.48	26.69 ± 1.95

considered image. The largest overhead is the application of T, which increases the duration of *Phase 3*, compared to the pure YCbCr to RGB conversion. Overall, the transformation requires $\approx 2ms$ more with respect to color conversions for a 512x512 image ($6.49ms$ vs $4.67ms$), and $\approx 8ms$ more for a 1280x1280 image ($34.68ms$ vs $26.69ms$). Notice that the execution time of LAPSE is independent of the considered image, and only changes in response of variations in external system conditions (e.g. CPU load).

On the contrary, the second phase of PCCE requires more than $10ms$, independently on the size of the image (it operates on histogram bins). Moreover, while independent on size, the duration of this phase is highly dependent on image content. Results in the table refer to two scaled versions of *Lena*, but the execution time could be even higher for other images [2].

Considering total execution times, LAPSE is approximately $3x$ faster than PCCE for 512x512 images, and $1.3x$ for 1280x1280 images. Taking into account only the additional time with respect to color space conversions, performance ratios become $8x$ and $2.3x$ respectively.

Noniterative PCCE [12] does not use the iterative optimization loop present in standard PCCE. However, this algorithm still includes complex operations (e.g. bilinear interpolation, local image filtering for strucural sensitivity evaluation, and internal power consumption estimation). Moreover, since NIPCCE applies different transformations to different local windows of the image, its complexity increases significantly for large images. This is confirmed by the results in Table VIII, which show that NIPCCE is faster than PCCE for 512x512 images, but significantly slower for 1280x1280 images. In both cases, NIPCCE remains significantly slower than LAPSE (2.3x times and 2.65x respectively). When excluding color space conversions, these speed-ups become 5.87x and 8.16x. Therefore, although this method improves the output image quality with respect to PCCE, as shown in [12], it is still not usable in a real-time setting.

Importantly, for the 1280x1280 image, even just the execution time of the two color-space conversions is too long for real-time application, with the constraints mentioned in Section I. An implementation that achieves the required performance could be obtained resorting to parallelism (multi-thread or GPU). However, this would require a high-end computing device exclusively dedicated to the transformation of images, with very large power and cost overheads. This confirms the need for hardware acceleration, in order to obtain the desired benefits from an energy-reducing image transformation. Software results are still useful to compare LAPSE with PCCE and NIPCCE, as no hardware acceleration for these methods has been proposed.

Finally, notice that the execution time reduction achieved by our method is mainly due to the fact that most of the complexity is moved to the offline training phase. In general, the time for running the algorithm of Figure 4 depends on the chosen grid granularity, as well as on the size and number of training images. As an example, we ran the algorithm with the platform and settings described above and in Section VII-A, on the biggest dataset considered in our experiments, i.e. the BSDS training set, which contains 400 images. The execution took about 102 minutes, or 1.7 hours, a perfectly acceptable time for a one-time offline procedure.

H. Hardware implementation results

A breakdown of the main figures of merit of each hardware component of the online transformation (Figures 5c-5g) is reported in Table IX. Notice that, differently from software, hardware blocks that belong to the same phase execute in a fully parallel fashion. The energy consumption is assumed to be zero when a block is not being used (e.g., the *RGB to YCbCr* block does not consume during phases 2 and 3). This simplification is quite accurate assuming power and/or clock gating are used. The power consumption reported in the last row is the average during the transformation of an image (i.e., the total energy divided by the execution time).

Hardware execution times are almost one order of magnitude faster than software. Even for the higher resolution image, execution completes in $\approx 3.3ms$. This result is more than sufficient for a real time implementation of the transformation on a HD OLED panel. Nonetheless, the total energy consumption is still extremely low. For sake of comparison, the datasheet of a 240x320 pixels AMOLED panel reports a typical power consumption of 260 mW [42]. With a refresh rate of 15 ms, this corresponds to an energy consumption per frame of $0.26 \cdot 0.015 = 3.9 \cdot 10^{-3}$ J. Although the panel is

[2]The timing constraints imposed by the display refresh rate must be respected in the worst case, i.e. for *any* image.

even *smaller* than the minimum image considered in Table IX, the energy consumption of the display is three orders of magnitude higher than that of the additional hardware required to implement LAPSE. This guarantees that the estimated savings reported in Sections VII-D and VII-E are maintained in a real implementation.

Table IX also helps in assessing the advantages of universal LAPSE (Section VI). In this variant, the computations of μ, σ^2 and $\mathbf{a}$ are not needed, and the corresponding hardware can be removed. While the latter has practically no impact on energy (being active in very few clock cycles), the former contributes to the 22.3% of the total consumption. This is reported in the last column of Table IX which shows a breakdown of the energy consumption of the different blocks. Hence, universal LAPSE is expected to consume $\approx 78\%$ of the energy of the adaptive version. Due to operation overlapping, the advantage in latency is practically null. As explained in Section VII-B, energy reduction comes at the cost of a loss in accuracy in terms of MSSIM of the output image. Whether this is acceptable depends on the system; adaptive LAPSE might be preferable for high-end devices, e.g. smartphones, whereas the universal solution could be interesting for low cost products like smartwatches.

VIII. Conclusion

In this paper, we have presented a new method for OLED displays power optimization, called LAPSE, that poses particular attention to the overheads of an online application. We have detailed the theoretical foundations of LAPSE, as well as the practical aspects of its implementation. In particular, we have described the components of an ASIC hardware accelerator for applying the proposed image transformation at runtime, with high performance and low overheads. This accelerator could be integrated in embedded System-on-Chips for devices that leverage OLED technology.

We have demonstrated that the effectiveness of LAPSE is comparable to that of previous state-of-the-art algorithms, both in terms of power reduction and image quality, despite the significant reduction in complexity. Moreover, we have shown how different levels of power reduction can be obtained accepting a proportional degree of image alteration. This aspect, not present in previous solutions, allows runtime reconfiguration of power consumption and quality in the OLED display, in response to environmental conditions.

References

[1] J. Bardsley, "International oled technology roadmap," *IEEE J. of Sel. Topics Quantum Electron.*, vol. 10, no. 1, pp. 3–9, Jan 2004.
[2] V. C. Bender, T. B. Marchesan, and J. M. Alonso, "Solid-state lighting: A concise review of the state of the art on led and oled modeling," *IEEE Ind. Electron. Mag.*, vol. 9, no. 2, pp. 6–16, June 2015.
[3] T. Simunic et al, "Event-driven power management," *IEEE Trans. Comput.-Aided Design of Integr. Circuits Syst.*, vol. 20, no. 7, pp. 840–857, Jul 2001.
[4] M. Dong and L. Zhong, "Power modeling and optimization for oled displays," *IEEE Trans. Mobile Comput.*, vol. 11, no. 9, pp. 1587–1599, Sept 2012.
[5] M. Dong and L. Zhong, "Chameleon: A color-adaptive web browser for mobile oled displays," *IEEE Trans. Mobile Comput.*, vol. 11, no. 5, pp. 724–738, May 2012.
[6] P. Ranganathan et al, "Energy-aware user interfaces and energy-adaptive displays," *Computer*, vol. 39, no. 3, pp. 31–38, March 2006.
[7] C.-H. Lin, C.-K. Kang, and P.-C. Hsiu, "Catch your attention: Quality-retaining power saving on mobile oled displays," in *Proc. of the ACM/EDAC/IEEE DAC*, 2014, pp. 1–6.
[8] D. Shin et al, "Dynamic driver supply voltage scaling for organic light emitting diode displays," *IEEE Trans. Comput.-Aided Design of Integr. Circuits and Syst.*, vol. 32, no. 7, pp. 1017–1030, July 2013.
[9] X. Chen et al, "Fine-grained dynamic voltage scaling on oled display," in *Proc. of ASP-DAC*, 2012, pp. 807–812.
[10] C. Lee et al, "Power-constrained contrast enhancement for emissive displays based on histogram equalization," *IEEE Trans. Image Process.*, vol. 21, no. 1, pp. 80–93, Jan 2012.
[11] Y.-O. Nam, D.-Y. Choi, and B. C. Song, "Power-constrained contrast enhancement algorithm using multiscale retinex for oled display," *IEEE Trans. Image Process.*, vol. 23, no. 8, pp. 3308–3320, Aug 2014.
[12] C. Y. Jang, S. J. Kang, and Y. H. Kim, "Noniterative power-constrained contrast enhancement algorithm for oled display," *IEEE J. Display Technol.*, vol. 12, no. 11, pp. 1257–1267, Nov 2016.
[13] Y.-T. Peng et al, "Histogram shrinking for power-saving contrast enhancement," in *Proc. of the IEEE ICIP*, 2013, pp. 891–894.
[14] S. Kang, "Image-Quality-Based Power Control Technique for Organic Light Emitting Diode Displays," in *IEEE J. Display Technol.*, vol. 11, no. 1, pp. 104–109, Jan 2016.
[15] S. Kang, "Perceptual Quality-Aware Power Reduction Technique for Organic Light Emitting Diodes," in *IEEE J. Display Technol.*, vol. 12, no. 6, pp. 519–525, June 2016.
[16] J. Shin, R. Park, "Power-constrained contrast enhancement for organic light-emitting diode display using locality-preserving histogram equalisation", in *IET Image Processing*, vol. 10, no. 7, pp. 542–551, July 2016
[17] C.-K. Kang, C.-H. Lin, and P.-C. Hsiu, "A win-win camera: Quality-enhanced power-saving images on mobile oled displays," in *Proc. of the IEEE/ACM ISLPED*, 2015, pp. 267–272.
[18] X. Chen et al, "Morph: Mobile oled-friendly recording and playback system for low power video streaming," in *Proc. of the ACM/EDAC/IEEE DAC*, 2016, pp. 1–6.
[19] D. J. Pagliari, M. Poncino, and E. Macii, "Low-overhead adaptive constrast enhancement and power reduction for oleds," in *Proc. of DATE*, 2016, pp. 343–348.
[20] A. K. Bhowmik and R. Brennan, "System-level display power reduction technologies for portable computing and communications devices," in *Proc. of IEEE PORTABLE07*, 2007, pp. 1–5.
[21] W.-C. Cheng and M. Pedram, "Power minimization in a backlit tft-lcd display by concurrent brightness and contrast scaling," *IEEE Trans. Consum. Electron.*, vol. 50, no. 1, pp. 25–32, Feb 2004.
[22] A. Iranli, W. Lee, and M. Pedram, "Backlight dimming in power-aware mobile displays," in *Proc. of ACM/IEEE DAC*, 2006, pp. 604–607.
[23] H. Shim, N. Chang, and M. Pedram, "A backlight power management framework for battery-operated multimedia systems," *IEEE Des. Test*, vol. 21, no. 5, pp. 388–396, Sept 2004.
[24] C.-C. Lai and C.-C. Tsai, "Backlight power reduction and image contrast enhancement using adaptive dimming for global backlight applications," *IEEE Trans. on Consum. Electron.*, vol. 54, no. 2, pp. 669–674, May 2008.
[25] S.-J. Kang, "Ssim preservation-based backlight dimming," *IEEE J. Display Technol.*, vol. 10, no. 4, pp. 247–250, April 2014.
[26] C. Y. Jang, S. J. Kang, and Y. H. Kim, "Perceived distortion-based progressive lcd backlight dimming method," *IEEE J. Display Technol.*, vol. 12, no. 10, pp. 1130–1138, Oct 2016.
[27] D. Y. Choi and B. C. Song, "Power-constrained Image Enhancement Using Multiband Processing for TFT LCD Devices with Edge LED Backlight Unit," in *IEEE Trans. Circuits Syst. for Video Technol.*, vol. PP, no. 99, pp. 1-1, 2017.
[28] S. C. Pei and C. T. Shen, "Color Enhancement With Adaptive Illumination Estimation for Low-Backlighted Displays," in *IEEE Trans. Multimedia*, vol. 19, no. 8, pp. 1956-1961, Aug. 2017.
[29] A. Koschan and M. Abidi, *Digital Color Image Processing*. Wiley, 2008.
[30] RK. Mantiuk, K. Myszkowski, and HP. Seidel. "High dynamic range imaging". John Wiley & Sons, Inc., 2015.
[31] ITU-T, *Information technology - Digital compression and coding of continuous-tone still images; JPEG File Interchange Format (JFIF)*, Recommendation T.871, May 2011.
[32] D. Shin et al, "Dynamic voltage scaling of oled displays," in *Proc. of ACM/EDAC/IEEE DAC*, 2011, pp. 53–58.
[33] Z. Wang et al, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. on Image Process.*, vol. 13, no. 4, pp. 600–612, April 2004.

TABLE IX
HARDWARE IMPLEMENTATION RESULTS.

Block	Area [mm^2]	Power [mW]	Latency/Image [ms]		Energy/Image [J]		
			512x512	1280x1280	512x512	1280x1280	Breakdown [%]
RGB to YCbCr	0.013	2.12	0.26	1.64	$5.55 \cdot 10^{-7}$	$3.47 \cdot 10^{-6}$	26.1
Compute μ and σ^2	0.029	1.81			$4.74 \cdot 10^{-7}$	$2.96 \cdot 10^{-6}$	22.3
Compute $\mathbf{a}$	0.042	5.28	$2 \cdot 10^{-6}$	$2 \cdot 10^{-6}$	$1.06 \cdot 10^{-11}$	$1.06 \cdot 10^{-11}$	≈ 0
Apply $T()$	0.044	3.10	0.26	1.64	$8.13 \cdot 10^{-7}$	$5.08 \cdot 10^{-6}$	38.2
YCbCr to RGB	0.008	1.09			$2.85 \cdot 10^{-7}$	$1.78 \cdot 10^{-6}$	13.4
Total	0.138	4.06	0.52	3.28	$2.13 \cdot 10^{-6}$	$13.29 \cdot 10^{-6}$	100

[34] D. Brunet, E. R. Vrscay, and Z. Wang, "On the mathematical properties of the structural similarity index," *IEEE Trans. Image Process.*, vol. 21, no. 4, pp. 1488–1499, April 2012.
[35] P. Nilsson et al, "Hardware implementation of the exponential function using Taylor series" NORCHIP, 2014, pp. 1-4.
[36] H. Sheikh et al, "Live image quality assessment database release 2." Online: http://live.ece.utexas.edu/research/quality
[37] P. Arbelaez et al, "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, May 2011. Online: http://dx.doi.org/10.1109/TPAMI.2010.161
[38] "The open video project." Online: open-video.org
[39] "Derf's test media collection." Online: https://media.xiph.org/video/derf/
[40] S. S. Agaian, K. P. Lentz, and A. M. Grigoryan, "A new measure of image enhancement," in *Proc. of IASTED*, 2000.
[41] https://vimeo.com/album/4604862
[42] "USMP-A24240TP AMOLED Product Specification.", Ver 1.2, US Micro Products, 2008.

Daniele Jahier Pagliari (M'15) received the M.Sc. and Ph.D. degrees in Computer Engineering from the Politecnico di Torino, Italy, in 2014 and 2018 respectively. He is currently a postdoctoral research assistant at the same institution. His main research interests focus on computer-aided design of digital systems, with particular emphasis on low-power optimization and approximate computing.

Enrico Macii (SM02, F07) is a Full Professor of Computer Engineering at Politecnico di Torino, Italy. Prior to that, he was an Associate Professor (1998-2001) and an Assistant Professor (1993-1998) at the same institution. From 1991 to 1997 he was also an Adjunct Faculty at the University of Colorado at Boulder. He holds a Laurea Degree in Electrical Engineering from Politecnico di Torino (1990), a Laurea Degree in Computer Science from Universit di Torino (1991) and a PhD degree in Computer Engineering from Politecnico di Torino (1995). From 2009 to 2016, he was the Vice Rector for Research at Politecnico di Torino; he was also the Rector's Delegate for Technology Transfer (2009-2015) and for International Affairs (2012-2015). His research interests are in the design of electronic circuits and systems, with particular emphasis on low-power consumption, optimization, testing, and formal verification. In the last few years, he has been growingly involved in projects focusing on the development of new technologies and methodologies for smart cities and bioinformatics. In the fields above he has authored over 450 scientific publications. Enrico Macii is a Fellow of the IEEE.

Massimo Poncino (SM12, F18) received the Ph.D. degree in Computer Engineering and the Dr. Eng. degree in Electrical Engineering from the Politecnico di Torino, Italy. He is currently a Full Professor of Computer Engineering at Politecnico di Torino. His research interests include several aspects of design automation of digital systems, with particular emphasis on the modeling and optimization of low-power systems. He is the author or coauthor of more than 350 journal and conference papers. He is an Associate Editor of the ACM Transactions on Design Automation of Electronic Systems and of IEEE Design & Test. Prior to that, he was an Associate Editor of IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (2006-2012). He was the Technical Program Co-Chair (in 2011) and the General Chair (in 2012) of the ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED). He serves on the Technical Program Committee of several IEEE and ACM technical conferences, including DAC, ICCAD, DATE, ISLPED, ASP-DAC, CODES-ISSS, and GLSVLSI. Massimo Poncino is a Fellow of the IEEE.