Fast ACE (FACE): An Error-Bounded Approximation of Automatic Color Equalization

Alice Plutino and Marco Tarini

***Abstract*— We present an efficient algorithm to approximate the Automatic Color Equalization (ACE) of an input color image, with an upper-bound on the introduced approximation error. The computation is based on Summed Area Tables and a carefully optimized partitioning of the plane into rectangular regions, resulting in a pseudo-linear asymptotic complexity with the number of pixels (against a quadratic straightforward computation of ACE). Our experimental evaluation confirms both the speedups and high accuracy, reaching lower approximation errors than existing approaches. We provide a publicly available reference implementation of our algorithm.**

***Index Terms*— Image processing, automatic color equalization (ACE), contrast enhancement.**

I. Introduction

AUTOMATIC Color Equalization (ACE) [1], [2] is a well-known unsupervised image processing procedure, which enhances globally and locally lightness and contrast while resulting in visual color constancy, and optimizing the quantization of the dynamic range. ACE is fairly widely used. Since its introduction in 2002 [1], [2], [3], ACE has had a great impact on several application domains. The recent review [4] tracks over 80 published scientific articles reporting utilizing ACE in numerous application fields, including: general-purpose image processing pipeline [5], [6], astrophotography [7], [8], cultural heritage [9], [10], [11], underwater imaging [12], [13], tone mapping applications [14], machine vision [15], medical imaging [16], [17]; it has also been used to reproduce and explain visual illusions such as simultaneous contrast [2]. The overall impact of ACE is also testified by measures such as the citation count of the articles introducing it [1], [2].[1]

Compared to other image filters, the appeal of ACE is that its definition stems directly from the self-adjusting behavior of the Human Visual System (HVS), as described in [18]. Among other practical benefits, ACE uses a non-Bayesian approach and is completely unsupervised and automatic, bypassing the need for any reference image or information.

Manuscript received 22 June 2022; revised 30 January 2023; accepted 21 April 2023. Date of publication 15 May 2023; date of current version 19 May 2023. This work was supported in part by CORISAMIL under Project VSP 170 7122-04C 120PB 27 182-016 and in part by NVIDIA. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Giacomo Boracchi. *(Corresponding author: Alice Plutino.)*

The authors are with the Computer Science Department, Universita degli Studi di Milano, 20133 Milan, Italy (e-mail: alice.plutino@unimi.it).

This article has supplementary downloadable material available at https://doi.org/10.1109/TIP.2023.3270770, provided by the authors.

Digital Object Identifier 10.1109/TIP.2023.3270770

[1]To date, Scopus, https://www.scopus.com/ reports more than 400 combined citations, and Google Scholar reports more than 600.

Unfortunately, from an algorithmic point of view, ACE original algorithm is inherently work-intensive to compute, making it unfeasible on large images. Its salient characteristic is that the value of every output pixel depends on the entire input image, leading to an $O(N^2)$ complexity, with N the number of pixels. The quadratic asymptotic complexity makes a direct application of ACE unpractical, except for very low resolution images. This probably hindered its further adoption in additional scenarios and determined that many situations where ACE has been adopted actually resorted to one approximation or another, rather than the original definition. At the same time, accurate approximations are difficult because, in ACE, the combined effect of far-away pixels never vanishes with their increasing distance (see Sec. III-B).

In this work, we present a new algorithm to rapidly approximate ACE, based on Summed Area Tables (SAT). Our approximation takes one user-defined parameter to let users control the trade-off between computation time and accuracy.

Compared to existing approximations (see Sec. II-B), ours offers key benefits:

- **high accuracy:** empirical evaluation (Sec. IX) shows that RMSE below 1 (on an intensity pixel intensity scale in 0..255) can be easily reached in short times. To our knowledge, no existing method can reliably offer an RMSE below 5.
- **error bounds:** our approximation comes with an *a priori* strict upper-bound for the discrepancy with the original ACE, as a function of the user parameter (even if the *a posteriori* validation shows this upper-bound to often be overly conservative).
- **arbitrary accuracy:** additionally, the error upper-bound can be set arbitrarily low, progressively forgoing the speedup; in the limit, an upper-bound of 0 removes completely the speedup, resulting in the original quadratic complexity. In contrast, most existing approximations have inherent, non-compressible, sources of error.

II. Related Work

A. Original ACE Algorithm

ACE is part of the Spatial Color Algorithms (SCAs) [18]. It was presented for the first time by Gatta et al. in 2002 [3], and published by Rizzi et al. in 2003 [1]. ACE is derived from the Retinex model [19], as other algorithms in this family [2]. As such, it applies principles that are motivated by the characteristics of the HVS reconstruction [20], [21]. Specifically, ACE focuses on the ability of the HVS to

reproduce color and lightness constancy, using a global (but localized) approach [22]. See Sec. III for a recap of the original algorithm.

B. Algorithms for Fast ACE Approximations

Since its initial proposal [1], the quadratic computational cost of ACE was identified as a crucial practical obstacle. This concern has led to the proposal of several countermeasures. The original algorithm already suggests, as an approximation technique, considering a subset of the image around a given pixel. We will discuss in Sec. III-B why this approach is inherently flawed, and leads to significant (and unbounded) approximation errors.

A similar approximation has been proposed by Artusi et al. in [23]. Here the authors propose a similar approach that leverages Singular Value Decomposition (SVD) to create a filtering mapping function before the application of ACE, anyway, this leads to similar issues (see Sec. III-B).

Chambah et al. [24] propose two linear techniques, Local Linear LUT (LLL) and PC2D (Principal Component analysis 2D mapping), for the enhancement of image sequences considering the spatial relationship between image areas. Similarly, Gatta et al. in [25] propose the use of LLL (Local Linear Look-up Table) to speed up the original ACE algorithm, keeping its local filtering properties, while adding a limited amount of chromatic noise. In those works, the idea is that ACE is fully computed for only a subset of the image, and then color to color piece-wise linear mapping is extracted from this subset and applied to the rest of the image. While this achieves considerable speedups, it sacrifices the strong dependency of the enhancement on immediately neighboring pixels, causing large approximation errors. It is also not clear if the introduced error can be bounded. An empirical comparison offered in Sec. IX, confirms these considerations.

In 2007 Bertalmío et al. [26] reduced the ACE complexity to $O(N \log N)$ starting from the idea that the major cost of the algorithm is in the computation of the contrast modification function. This study was reported in [27], where a new model named Kernel-Based Retinex (KBR) was proposed.

One of the most successful fast approximations of ACE, based on fast-convolutions, was introduced by Getreuer [28] and is the current go-to implementation of ACE. It was adopted for applications ranging from traffic images [29], underwater imaging [30], [31] and medical imaging [32], among many others. This approach introduces several sources of approximation error; the first one pertains to the treatment of the image boundaries; to make the processing more "convolution friendly", the input image is considered as indefinitely extended symmetrically beyond its original boundaries. This modification departs, to some extent, from the HVS-inspired principles that motivate the adoption of ACE. In practical terms, the effect on the output image of this modification is difficult to predict or bound; previous work neglect to measure this effect, but our empirical experiments (Sec. IX-C) reveal it to be significant, being around 5 RMSE (on a scale of 255 intensity levels). Furthermore, this is an inherent incompressible source of error that cannot be controlled or reduced by, for example, the choice of a parameter. It should be noted that the measured errors reported in [28] are computed net of the effect of this initial modification, and only reflect the effect of further approximations stacked on top of it as the used "ground truth" is also affected by this initial approximation; this was probably due to practical reasons, as producing ground truths to compare against can absorb days of computation (for a high-resolution image). This approach is, however, extremely efficient, and [28] imposes processing times that are significantly shorter than our owns, making it still the best choice in many circumstances. Plots in Figure 10 summarize this trade-off.

C. GPU-Based Accelerations

A CUDA implementation of ACE for stereoscopic streams was published by Gadia et al. in [33], and an evaluation of gamut changes in the final image was presented by the authors. Similarly, a further implementation of ACE algorithm through FPGA using VHDL was proposed by Romero et al. [34]. Hardware-based implementations achieve constant speedups, which can be of great practical significance, although they do not affect the asymptotic cost. These efforts are orthogonal to our proposal: our prototypical implementation of the proposed algorithm has not been explicitly Hardware-accelerated (other than what is automatically done by the MATLAB suite), and we conjecture that it could similarly benefit from a comparable engineerization process.

D. Per-Frame Enhancement for Video Processing

Among the motivations for accelerating ACE computation, we aim at unlocking the possibility of using it as a filter for videos and providing SCAs for movie restoration. In 2003, Chambah et al. [35] proposed the first such application and this work opened a new direction for *perceptual color restoration*. Films and the analog materials that need to be conserved and restored have been subject, by now, to considerable decay and aging. Furthermore, during cinema history, different techniques of color reproduction have been developed, using different film bases, materials, filters, and instruments. As a consequence, a trustworthy and faithful reference of the original colors and contrast is not available to perform the restoration and enhancement of films [36]. With their work, Chambah et al. proposed a new approach to film color and contrast restoration, using the SCAs implementations to recover the original appearance of old digitized films.

From this first application, SCAs have been used for frame restoration [35], [37], [38] and as a kick-off technique in the restoration workflow [39]. A recent analysis of image quality in film restoration demonstrated the significant advantages that unsupervised SCAs can provide to film restoration [40], [41]. Despite the demonstrated advantages that ACE algorithm would provide in this context, its execution time still limits considerably its potential applicability.

III. ACE Definition and Computation

Here, we recap the original Automatic Color Equalization (ACE) algorithm [1], using a formulation that is equivalent to

the original one but eases the definition of our modification of the algorithm later. Used notation is summarized in Table I.

ACE computation consists of two steps: the chromatic/spatial adjustment, and the dynamic tone reproduction. Given an input rasterized gray-scale image I, the first stage produces an enhanced image E. In RGB color images, this computation is repeated for each color channel separately (red, green, and blue).

For each pixel $\mathbf{p}$ in image I, its *unnormalized ACE value* $\mathrm{V}[\mathbf{p}]$ is defined as the summation, over all other pixels $\mathbf{q}$, of the *intensity difference* between the two pixels $c(\mathbf{p}, \mathbf{q})$, scaled by the inverse of their Euclidean distance $d(\mathbf{p}, \mathbf{q})$:

$$\mathrm{V}[\mathbf{p}] = \sum_{\mathbf{q} \in \mathrm{I} \setminus \{\mathbf{p}\}} \frac{c(\mathbf{p}, \mathbf{q})}{d(\mathbf{p}, \mathbf{q})}, \tag{1}$$

The intensity difference $c(\mathbf{p}, \mathbf{q})$ is given by

$$c(\mathbf{p}, \mathbf{q}) = f_\sigma \left(\mathrm{I}[\mathbf{p}] - \mathrm{I}[\mathbf{q}] \right), \tag{2}$$

where the non-linear *amplification function* f_σ is defined as

$$f_\sigma(x) = \begin{cases} -1 & \text{if } \sigma x \leq -1 \\ \sigma x & \text{if } -1 < \sigma x < +1 \\ +1 & \text{if } \sigma x \geq +1 \end{cases} \tag{3}$$

(that is, f_σ is a scaling by a factor σ followed by clamping in the -1 to $+1$ interval). The value σ is the only parameter of ACE, and acts as a way to tune the contrast [1]. While more generic functions can be used in place of f_σ, Equation (3) is the most commonly adopted one. We employ this definition, but our method can be easily extended to arbitrary functions.

Normalization: the normalized ACE value $\mathrm{E}[\mathbf{p}]$ for pixel $\mathbf{p}$, in the interval $[-1, +1]$, is then given by

$$\mathrm{E}[\mathbf{p}] = \frac{\mathrm{V}[\mathbf{p}]}{\mathrm{V}_{\max}[\mathbf{p}]} \tag{4}$$

where $\mathrm{V}_{\max}[\mathbf{p}]$ is the maximal value that can be assumed, in absolute value, by $\mathrm{V}[\mathbf{p}]$ for a pixel in position $\mathbf{p}$ (irrespective of the content of the image), which is:

$$\mathrm{V}_{\max}[\mathbf{p}] = \sum_{\mathbf{q} \in \mathrm{I} \setminus \{\mathbf{p}\}} \frac{1}{d(\mathbf{p}, \mathbf{q})} \tag{5}$$

Final Tone Mapping: the resulting enhanced image E is mapped into the final output image. This is done maximizing its dynamic range by re-normalizing the white at a global level. Computationally, this is a much simpler process, and it is linear with the number of pixels. The simplest method, which we adopt in our examples, is to scale linearly the range of values E (in each channel) into the range $[0, 1]$. As an alternative, orthogonal to our work, Gray World and White Patch assumptions can be used, considering that the final output colours may differ significantly depending on the adopted scaling method [42], [43], [44].

A. Straightforward Computation of ACE

A straightforward approach to compute Equation (1) is to iterate over each pixel $\mathbf{p}$, and, for each pixel, iterate over all the other pixels $\mathbf{q}$ (and likewise for Equation 5).

TABLE I
LIST OF SYMBOLS AND NOTATION USED IN THIS PAPER

I	Input image.
E	Enhanced image.
$n \times m$	Resolution of I.
$\mathbf{p}, \mathbf{q}, \ldots$	Pixel coordinates.
$\mathrm{I}[\mathbf{p}]$	Intensity of I at point $\mathbf{p}$.
$\mathrm{V}[\mathbf{p}]$	Unnormalized ACE value in $\mathbf{p}$ (Eq. 1).
$\mathrm{V}_{\max}[\mathbf{p}]$	Maximal value that can be assumed by $\mathrm{V}[\mathbf{p}]$ (Eq. 5).
$\mathrm{E}[\mathbf{p}]$	Normalized ACE value in $\mathbf{p}$ (Eq. 4).
$c(\mathbf{p}, \mathbf{q})$	Intensity difference between pixels in $\mathbf{p}$ and $\mathbf{q}$ (Eq. 2).
f_σ	Amplification function (Eq. 3).
σ	User-defined ACE amplification factor.
$d(\mathbf{p}, \mathbf{q})$	Euclidean distance between $\mathbf{p}$ and $\mathbf{q}$.
$C(\mathbf{p}, R)$	Contributions to pixel in $\mathbf{p}$ by all pixels in rectangle R.
$\tilde{C}(\mathbf{p}, R)$	Approximation of $C(\mathbf{p}, R)$ (Eq. 11).
L	Layout, a set of rectangular regions (Sec. IV).
R_i	A rectangular region in L.
S_k	Summed Area Table to be used for pixels with intensity value k.
d_{min}	Minimal and maximal distance from any pixel inside
d_{max}	a rectangle to the central pixel of L (Fig. 3).
d_{avg}	The average between d_{min} and d_{max}.

This approach would be, as noted, quadratic with the number of pixels, and thus extremely slow. For example, in our experiments, a weakly parallelized implementation of this approach took more than a day for every single three channeled 1000×1000 image (see Table III).

B. Failure of Trivial Approximations of ACE

An immediate observation is that, when computing $\mathrm{E}[\mathbf{p}]$, the further an input pixel $\mathbf{q}$ is from pixel $\mathbf{p}$, the less it contributes to the final value. Therefore, a tempting time-saving approximation is to neglect, in the inner loop over other pixels $\mathbf{q}$, all pixels further away than a given threshold distance from $\mathbf{p}$. This approach has been suggested in the literature, including in the original definition of ACE [1].

At a closer scrutiny, however, the approximation resulting from this class of approaches is excessively inaccurate. While the individual contributions of pixels at any given distance d diminishes linearly, their *number* increases linearly, so the expected combined contribution of all pixels at distance d is constant for every d (within the boundary of the image). Any optimization that only considers a subset of the image around a given pixel will introduce unbounded errors.

We conjecture that this intrinsic characteristic of ACE is directly linked to its desirable characteristics. For example, if the scaling of the distance were squared, substituting $||\mathbf{p}-\mathbf{q}||$ with $||\mathbf{p}-\mathbf{q}||^2$ in Equations (1) and (5), the combined effect of all pixels at distance d would vanish with increasing d, but [1] reports that the resulting filter downgrades, functionally, to a simple local contrast enhancement filter.

IV. Overview of FACE: Fast ACE Computation

Like in the straightforward application of the definition, our approximated algorithm processes each pixel of the input image I, at position $\mathbf{p}$, and produces the corresponding value $\mathrm{E}[\mathbf{p}]$ of the output image (potentially, in parallel).

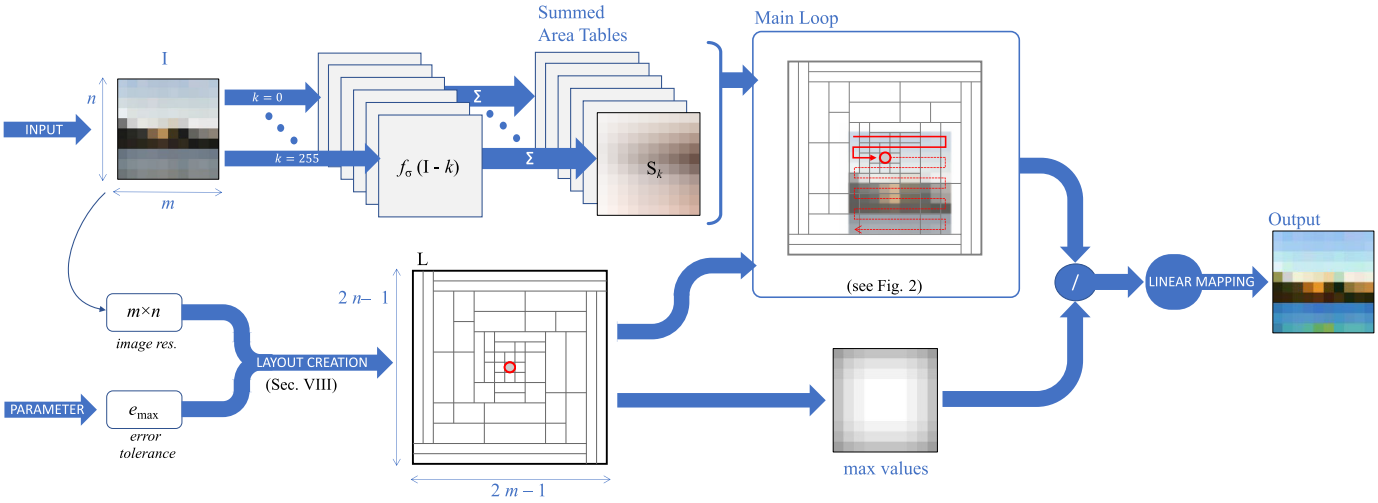


Fig. 1. A graphical overview of the proposed FACE algorithm. Given an input image I sized $m \times n$, we first precompute an optimized layout or rectangles L, which partitions an area around a central pixel (red circle) into several rectangles of appropriate shapes and sizes. The layout L is constructed for a given maximal tolerated error e_{max} and image resolution. For each possible intensity value k (and for each channel), we pre-compute image $f_\sigma(I-k)$ (equation 3) over the entire image channel and integrate it into a corresponding Summed Area Table S_k. At this point (main loop), the Layout L is "slid" over the image, by positioning its central pixel over each pixel in position $\mathbf{p}$ of I; for each pixel in $\mathbf{p}$ having intensity I[$\mathbf{p}$], the combined contribution from all pixels inside a rectangle of L is quickly computed, in constant time, by accessing S_k with $k = \mathrm{I}[\mathbf{p}]$. Summing this over all the rectangles of L, we get an approximation of V[$\mathbf{p}$] (Equation 1) for that pixel. The layout L is also used to compute, for every pixel, the maximal value of that pixel, that is, value $\mathrm{V}_{\max}$ (Equation 5). The per-pixel division of these two images gives us the final image, which is then linearly mapped in the final output image, as normal.

Consider one pixel in position $\mathbf{p}$. To quickly evaluate the value E[$\mathbf{p}$], we partition the rest of the input image, I $\setminus$ {$\mathbf{p}$}, into a set L of a certain number of disjoint integer rectangles of various sizes $L = \{R_0, R_1, \dots\}$, with

$$\bigcup_{R_i \in L} R_i = \mathrm{I} \setminus \{\mathbf{p}\} \tag{6}$$

and

$$\forall_{R_i, R_j \in L} \; R_i \cap R_j = \varnothing. \tag{7}$$

We then rewrite the sum over all other $\mathbf{q}$ in I (Equation 1) as the sum over all rectangles in L:

$$\mathrm{V}[\mathbf{p}] = \sum_{R_i \in L} \left(\sum_{\mathbf{q} \in R_i} \frac{c(\mathbf{p}, \mathbf{q})}{d(\mathbf{p}, \mathbf{q})} \right) \tag{8}$$

and similarly for $\mathrm{V}_{\max}[\mathbf{p}]$ (Equation 5).

The *combined* contribution to V[$\mathbf{p}$] from all pixels inside a rectangular region R_i, which we denote as as $C(\mathbf{p}, R)$ is:

$$C(\mathbf{p}, R) \triangleq \sum_{\mathbf{q} \in R} \frac{c(\mathbf{p}, \mathbf{q})}{d(\mathbf{p}, \mathbf{q})} \tag{9}$$

leading to

$$\mathrm{V}[\mathbf{p}] = \sum_{R_i \in L} C(\mathbf{p}, R_i). \tag{10}$$

The core of our approximation technique is to quickly approximate the values for $C(\mathbf{p}, R_i)$, in constant time (independently from the size of the rectangle R_i), introducing only a small, upper-bounded approximations. This process, based on Summed Area Tables (SAT) [45], is detailed in Sec. V. The consequence is that V[$\mathbf{p}$] is evaluated using a number of operations that is linear with the number of rectangles in L, rather than the drastically larger number of original pixels.

This process requires two pre-processing steps.

a) SAT creation (Sec. VI): we pre-compute a set of Summed Area Tables for the given input image. One SAT is created for each possible gray-scale value of a pixel (for RGB images, this is repeated for each channel).

b) Layout creation (Sec. VIII): we produce an optimized *layout* L of rectangles partitioning the region around the processed pixel $\mathbf{p}$. A single layout is created, and used for all processed pixels, by sliding the layout over the image during the processing (see Figure 2).

To construct L, we analyze the error introduced by any considered rectangle R (Sec. VII), which depends on its size, shape, and position. We derive maximal and expected errors as closed functions of R, and use these estimations to keep the total error under an user-defined upper-bound.

The overall resulting algorithm is summarized in Figure 1.

V. Per-Rectangle Approximated Evaluation

At the core of our system, we approximate $C(\mathbf{p}, R)$ (Equation 9) as $\tilde{C}(\mathbf{p}, R)$, defined as follows:

$$C(\mathbf{p}, R) = \sum_{\mathbf{q} \in R} \frac{c(\mathbf{p}, \mathbf{q})}{d(\mathbf{p}, \mathbf{q})} \approx \sum_{\mathbf{q} \in R} \frac{c(\mathbf{p}, \mathbf{q})}{d_{avg}(R)} = \tilde{C}(\mathbf{p}, R) \tag{11}$$

where $d_{avg}(R)$ is defined as the average between the maximal d_{max} and minimal d_{min} distances from any $\mathbf{q} \in R$ to $\mathbf{p}$ (see Figure 3). The substitution of $d(\mathbf{p}, \mathbf{q})$ with $d_{avg}(R)$ is central in our approximation; we analyze its impact on accuracy is in Sec. VII. The rationale is that, because $d_{avg}(R)$ does not depend on $\mathbf{q}$, the value $\tilde{C}(\mathbf{p}, R)$ can be written as

$$\tilde{C}(\mathbf{p}, R) = \frac{1}{d_{avg}(R)} \sum_{q \in R} c(\mathbf{p}, \mathbf{q}) \tag{12}$$

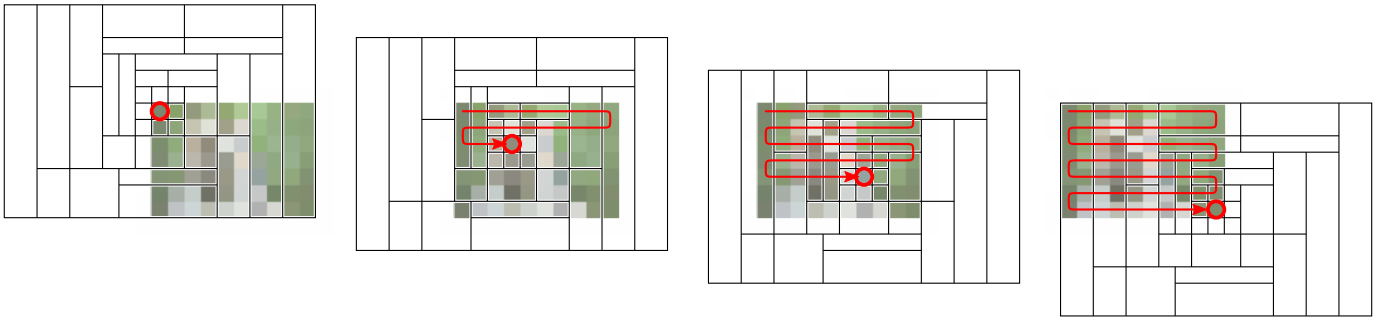


Fig. 2. A toy example of the main loop, for a 10×7 input image I (green-hued pixels). The optimized layout L is slid over I, centering its central pixel (red circle) over each pixel $\mathbf{p}$ of I; this induces a different partition of I into rectangles, some of which can be partially or completely empty. For illustration, four iterations are shown; the order of the processed pixels is arbitrary. The layout L is sized $(2n-1) \times (2m-1) = 19 \times 10$, so that it always covers the entire image, including when centered in a corner (leftmost and rightmost iterations).

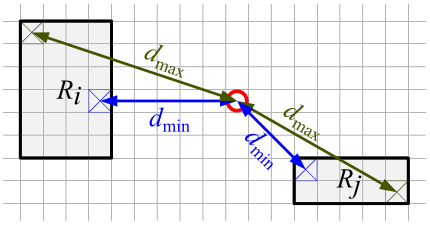


Fig. 3. Examples of minimal (d_{min}) and maximal (d_{max}) distances, for two rectangles R_i and R_j, defined as the distances from the central layout pixel (red circle) to the nearest (blue cross) and the farthest pixel (green cross) in R.

and, the summation can be quickly computed using SAT tables, in constant time (Sec. VI).

VI. Using Summed Area Tables

A Summed Area Table (SAT) [45], sometimes referred to as *Integral Image*, the is a well-known technique, in Computer Graphics and Image Processing, to evaluate the sum of values inside any axis-aligned sub-rectangle of an image, irrespective of its size, in constant time.

Specifically, given a $w \times h$ grid of values, a SAT is a $(w+1) \times (h+1)$ table of partial sums, each element $S[i, j]$ storing the sum of all original grid values at coordinates x, y, with $x < i$ and $y < j$.

The summed value for any rectangle $R = [x_0, x_1) \times [y_0, y_1)$ (that is, R spans from x_0 included to x_1 excluded in the horizontal direction, and from y_0 included to y_1 excluded in the vertical direction) is then given by:

$$S(R) \triangleq S[x_1, y_1] + S[x_0, y_0] - S[x_1, y_0] - S[x_0, y_1] \quad (13)$$

In our case, we need the sum of contributions that are defined, at every pixel in position $\mathbf{q} \in R$, as $c(\mathbf{p}, \mathbf{q})$ (Equation 2), which depends not only on the pixel values inside R but also on the value $\mathrm{I}[\mathbf{p}]$ at the currently input processed pixel $\mathbf{p}$.

In a quantized image, however, any pixel value can only assume one of M possible values, $[v_0 \dots v_{M-1}]$. For example, with common 8-bit images, $M = 256$. For each of the M possible values v_i, we first precompute the image $f_\sigma(\mathrm{I} - v_i)$ (by subtracting v_i to the entire image and computing f_σ to every pixel), and then compute the corresponding SAT S_i. We cache all M separated SAT $\{S_0 \dots S_{M-1}\}$. For color images, this process is repeated for each RGB channel.

Efficiency: SAT computation is only $\mathrm{O}(N)$ with the number of pixels [45], and can be made very efficient, leveraging on GPU parallelization techniques [46].

Consistency: Because we slide the layout L over the image (Fig 2), some rectangles in L will be partially or completely outside of image I. By simply clamping the boundary of the rectangle R on the boundary of I in Equation 13, we obtain the correct result for rectangles partially outside I, and (correctly) zero for any rectangle completely outside I.

VII. A-Priori Estimation of Approximation Error

A. Maximal Error for a Rectangle

The approximation in Equation (11) introduces an error for each pixel $\mathbf{q}$ in R defined as the discrepancy between the real value of its contribution to $C(\mathbf{p}, R)$ and the approximation of this value in $\tilde{C}(\mathbf{p}, R)$:

$$\begin{aligned} error(\mathbf{q}) &= \left| \frac{c(\mathbf{p}, \mathbf{q})}{d(\mathbf{p}, \mathbf{q})} - \frac{c(\mathbf{p}, \mathbf{q})}{d_{avg}(R)} \right| \\ &= |c(\mathbf{p}, \mathbf{q})| \cdot \frac{|d(\mathbf{p}, \mathbf{q}) - d_{avg}|}{d(\mathbf{p}, \mathbf{q}) \cdot d_{avg}}. \end{aligned} \quad (14)$$

By the definition of ACE, we know that $|c(\mathbf{p}, \mathbf{q})|$ is upper-bounded by 1 (Equations 2 and 3).

Because $\mathbf{q}$ is inside R, the value $d(\mathbf{p}, \mathbf{q})$ is in the interval from d_{min} and d_{max}, and $|d(\mathbf{p}, \mathbf{q}) - d_{avg}|$ is upper-bounded by half the length of that interval. This leads to the following upper-bound for the error introduced for pixel $\mathbf{q}$:

$$error(\mathbf{q}) \leq \frac{d_{max} - d_{min}}{2 \cdot d_{min} \cdot d_{avg}}. \quad (15)$$

Therefore, the total error introduced by a given rectangle R cannot be larger than the above quantity repeated for each pixel inside R, and, in conclusion, is bound from above by

$$error_{max}(R) = area(R) \frac{d_{max} - d_{min}}{2 \cdot d_{min} \cdot d_{avg}} \quad (16)$$

where $area(R)$ is the number of pixels inside rectangle R.

B. Predicted Average Error for a Rectangle

The bound in Equation (16) assumes the worst-case scenario, where the errors from each pixel inside the rectangle R never cancel out. In reality, this is extremely unlikely to occur: for the sign of the error to always match, each pixel in R at a distance $d > d_{avg}$ would need to be brighter than the pixel in $\mathbf{p}$, and each pixel at a distance $d < d_{avg}$ would need to be darker (or the vice-versa). If, for example, the entire sub-image inside R is all uniformly brighter or darker than the pixel in $\mathbf{p}$, the error would cancel out almost completely.

It is impossible to estimate the *expected* amount of approximation error generated for a rectangle R unless the probabilistic distribution of pixel values is known. As a simplifying assumption, we consider an ideal case where each pixel has an equal and independent probability to be either overestimated or underestimated by the approximation in Equation (11), which is equivalent to postulating that the probability for a pixel in $\mathbf{q} \in R$ to be brighter or darker than the central pixel $\mathbf{p}$ is independent from $\mathbf{q}$ being closer or further than d_{avg} from $\mathbf{p}$. Under this assumption, the expected averaged error, after Donsker's invariance principle [47], is predicted by

$$error_{avg}(R) = \sqrt{area(R)}\frac{d_{max} - d_{min}}{2 \cdot d_{min} \cdot d_{avg}} \tag{17}$$

This estimation is still conservative because, among other things, the distance of every pixel from $\mathbf{p}$ is assumed to always be maximally discrepant from d_{avg}; still, it reflects the expected cancellation of errors. We use this prediction to inform the creation of optimal layouts (Sec. VIII); we empirically validate this choice in Sec. IX-B.

C. Observations on the Error Estimation

A few considerations of practical importance stem from the formulation of the maximal (Equation 16) and average (Equation 17) error.

The (worst case) approximation error introduced by a rectangle grows *linearly* with its *area*, but it also decreases *quadratically* with its *distance* from the processed pixel $\mathbf{p}$. Therefore, in a layout where rectangle areas scale quadratically with their distance from $\mathbf{p}$, the error contributed by each rectangle can be expected to be similar. This is ideal, because each rectangle imposes the same computational cost, regardless of its dimension.

This consideration also suggests that FACE is scalable: as the input resolution increases, the number of rectangles grows slowly, as each additional rectangle can cover larger and larger regions. Our empirical experiments confirm this expectation.

The error bound for a rectangle is also affected by its *shape*, for the same area and distance. A wide and short rectangle located straight above or below $\mathbf{p}$ will have a more favorable error bound, compared to square-shaped one, because its internal pixels will present a smaller variation of distance from $\mathbf{p}$ (making $d_{max} - d_{min}$ smaller); likewise, a rectangle straight to the left or the right of $\mathbf{p}$ must be tall and thin in order to minimize the error bound.

The error bound and the predicted error are always $< +\inf$, as $d_{avg} \geq d_{min} > 0$, because the central point is never found inside any rectangle. Also, any 1×1 rectangle has an error bound and error prediction of 0, as $d_{max} = d_{min}$, and, in this case, the approximated value is exact.

D. Global Error Estimation

To estimate, or bound, the total error, we just sum up all estimations, or bounds, from every rectangle. Considering that, at most, one-quarter of the area of the layout contributes to the error, and the rest of it falls outside the image, a better estimation is obtained by dividing by 4. Both the bound and the estimation are still conservative, for the reasons discussed; however, they serve well to inform the optimization of the layout, as described in the next Section.

VIII. Automatic Synthesis of Layouts

We construct a layout L consisting of a set of non-overlapping rectangles, in a pre-processing phase. Our objective is to minimize the combined error bound while keeping the number of rectangles in L low. Finding the optimum among all possibilities would be unfeasible; instead, we employ a greedy heuristic that produces a sufficiently good layout.

The $m \times n$ resolution of input image I is an input of this procedure. The sought layout L must cover a rectangular area sized $(2m-1) \times (2n-1)$ pixels so that when its central pixel is positioned on any pixel of I, the entire image is still covered by L (see Figure 2); the central pixel of that area must be outside of all rectangles in L and all other pixels must belong to exactly one rectangle in L.

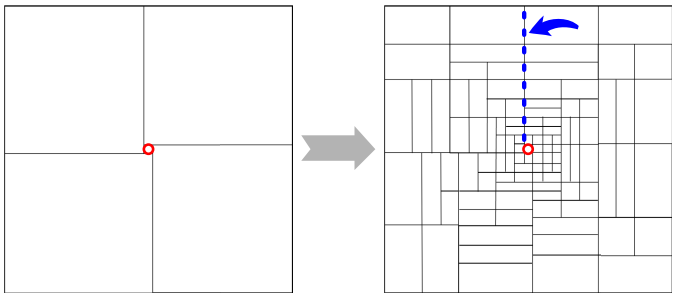
We start with an initial procedurally-generated layout (Sec. VIII-B), which covers the necessary areas using only a few rectangles; then, we iteratively refine it (Sec. VIII-A), by selecting and splitting one rectangle at a time.

A. Iterative Layout Refinement

At each iteration, we identify the rectangle R_i currently contributing the largest total error, given by Equation (16), split it into two sub-rectangles R_{i0} and R_{i1}, and substitute R_i in the layout with R_{i0} and R_{i1}.

Splitting Procedure: given a chosen R_i, we evaluate up to four potential ways to split it in two (see Figure 5): up to two horizontal ones (unless R_i is already only 1 pixel tall), and up to two vertical ones (unless R_i is already only 1 pixel wide). For each direction, we split the rectangle in half; if the extension of the rectangle is an odd number of pixels, we consider two split positions, by rounding either up or down. We pick whichever alternative results in the smallest combined error of the two resulting rectangles R_{i0} and R_{i1}, again using Equation (16). Although many other potential splits could be considered, in our greedy strategy we only evaluate these four, for the sake of simplicity.

Stopping Condition: The rectangles are split until their combined error falls below a user-prescribed maximal error tolerance e_{max}. As an alternative, the user can directly specify a targeted number of rectangles k_{max}, and the resulting final error bound is a function of k_{max}. Setting k_{max} serves as a way to control the computation times, while obtaining the best possible approximation within that budget. Vice-versa,



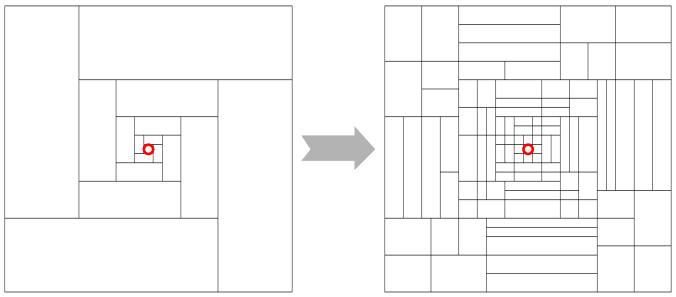


Fig. 4. Example of layouts automatically generated for an image of resolution 16×16. Left: two procedurally generated *initial* solutions; right: the result of our iterative refinement, which targets 100 rectangles, and minimizes the maximal error. If the refinement is initialized with the coarsest possible pattern (top), the final result presents a less favorable error bound and predicted average error; this is due to the preservation of the poorly positioned initial lines (such as the blue dotted line). To avoid this issue, we initialize the layout using a slightly denser procedurally generated solution (bottom).

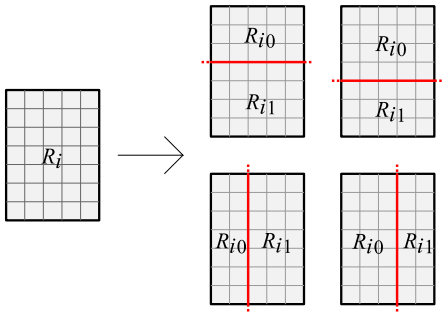


Fig. 5. The four splits considered when R_i is a 5×7 rectangle.

setting e_{max} is a way to control the quality of the results, while striving to minimize the computation times to achieve that quality. See Sec. IX for examples of used values.

Guiding Criterion: as a variant, instead of minimizing the error *upper-bound* of Equation (16), we can choose to target the minimization of the *expected* error in Equation (17), trading off some guarantee on the maximal error for a lower expected error. To do so, we simply employ the latter Equation, in place of the former, to select the rectangle, to choose the split, and to evaluate the stopping criteria. Figure 6 shows the impact of this choice on the produced layouts.

The only difference between the two equations is in the exponent of the area, which is 1 in Equation (16) and 0.5 in Equation (17), so we can generalize and use any other exponent, to get intermediate solutions. Experimental

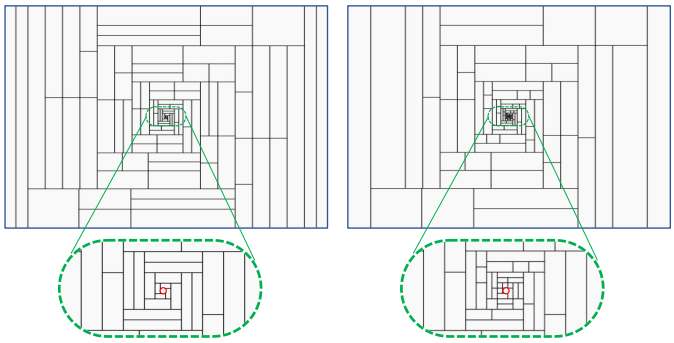


Fig. 6. Two different layouts were obtained for an image of resolution 150×100, targeting a fixed number of 100 rectangles, and optimized to minimize the global error bound (left), or the predicted average error (right). Optimizing for the predicted average error results in a comparatively denser rectangles partition around the central pixel of the layout.

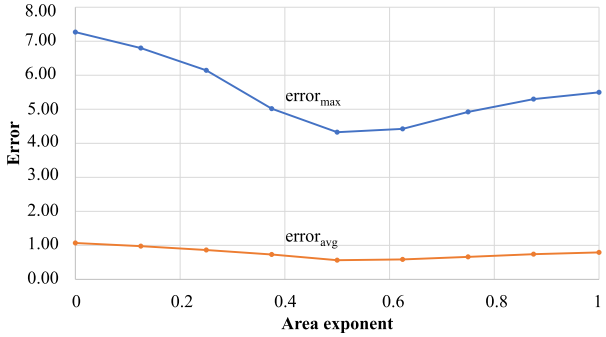


Fig. 7. Measurement of the maximum (blue line) and average (orange line) error over all pixels obtained when using layouts constructed using different Area Exponent values in the per-rectangle error estimation, averaged over all reported in Table II. The lowest values are found for the Area Exponent value 0.5, corresponding to the adoption of $error_{avg}$ (Equation 17) as the criterion guiding the construction of the layout.

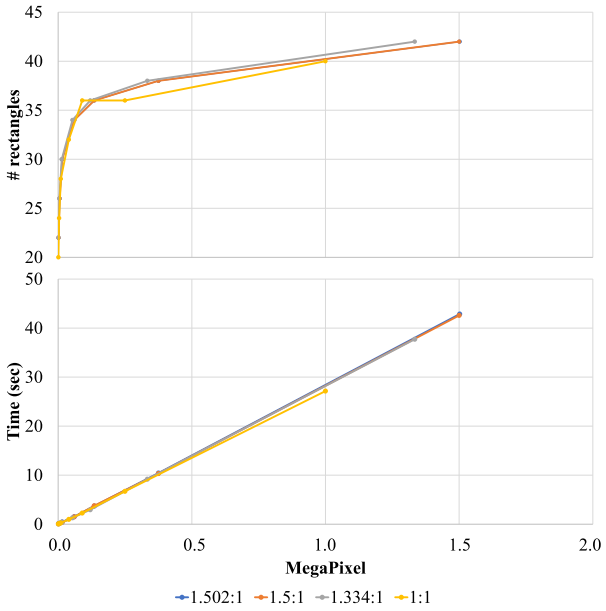


Fig. 8. Empirical evaluation of the scalability of FACE. Progressively increasing the number of pixels in the input image, the number of rectangles required in our layout to target the same expected error increases slowly (top), resulting in an almost linear increase in total computational time (bottom).

results (see Figure 7) confirm the expectation that the lowest *average* error is indeed obtained using the value 0.5, providing

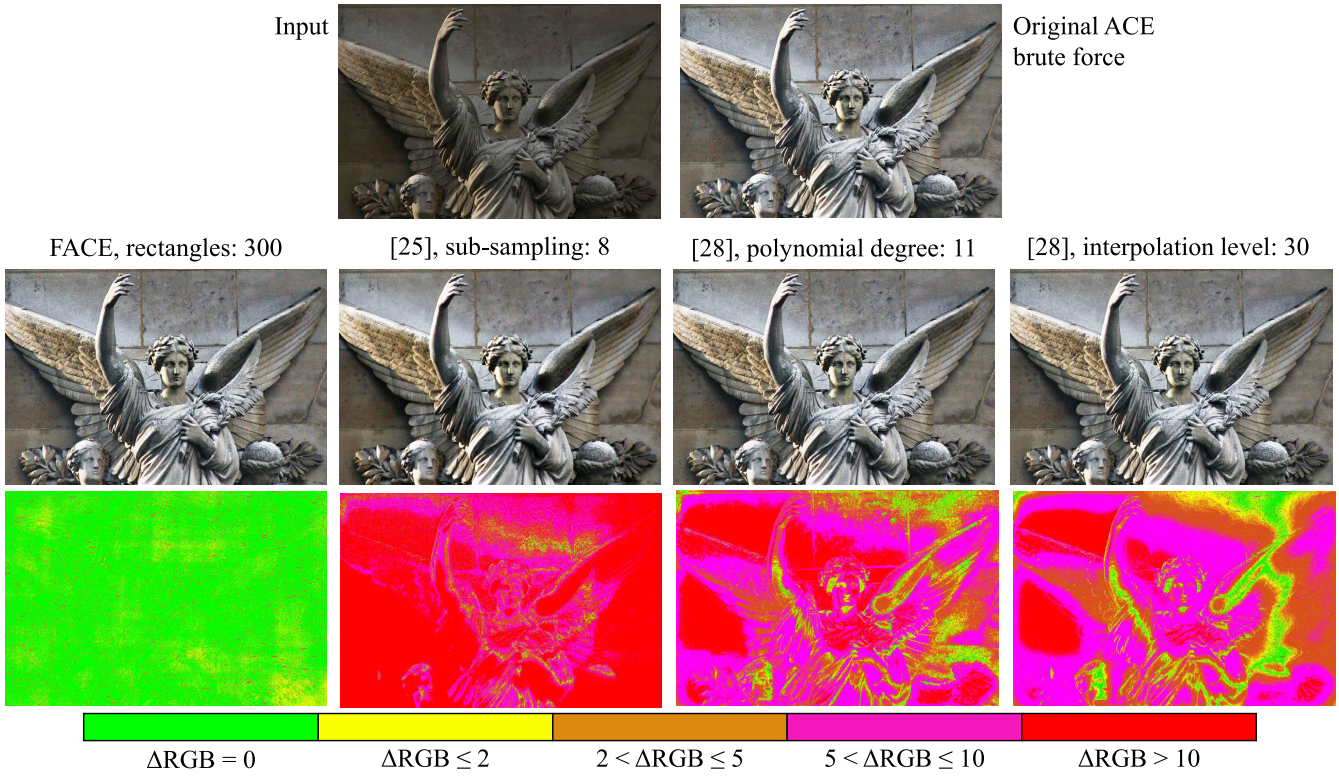


Fig. 9. Example image ("Angel") from the data-set [48] (see Table II). Top: the input image and original ACE enhancement. Second line: four approximations of ACE: FACE, Gatta et al. [25], and polynomial and interpolation-based methods of Getreuer [28]. Bottom: the difference images versus ACE, color coded with the Euclidean distance in the RGB space (ΔRGB), on a scale from 0 to 255.
.

empirical indirect support for our assumption on error distribution. Unless otherwise specified, we use that value in our experiments.

Results: the generated rectangle layouts (e.g., Figures 4, bottom right and 6) automatically adhere to the considerations on ideal sizes, positions, and shapes of the rectangles (Sec. VII-C). For example, the sizes of the rectangles tend to increase with their distance from the central **p**, and long and thin rectangles tend to be constructed on the sides of **p**, while tall and flat rectangles are generated above and below it.

Soundness: 1×1 rectangles are associated with 0 error, meaning that they will never be elected for a split, and also that the procedure always converges, as the total error will eventually go below any user-requested positive maximal error. If, hypothetically, a maximal error of 0 is requested, all rectangles will be split until only 1×1 rectangles remain, as any larger rectangle is associated with an error > 0. This observation means that FACE is a real generalization of the original ACE, and it naturally produces the exact same result when a global error of zero is requested (but, in this case, no benefit is offered in terms of performances).

B. Initial Layout Definition

We need an initial layout as a starting point for the iterative refinement. This layout must be very coarse, to leave the optimization procedure in control of the produced layout. We use a simple procedurally generated layout, as follows.

The trivial solution with a single rectangle covering the entire region is not valid, because we need the central pixel to be outside any rectangle. The coarsest possible solutions use 4 rectangles, such as the "gammadion" pattern depicted in Figure 4, top-left. This pattern, however, it is not ideal for our purposes because it features four axis-aligned lines emanating from the central point and traversing the entire image. These lines will be preserved by the refinement phase, hindering the creation of ideally shaped rectangles (Figure 4, top).

As a better solution, we opt to construct the initial layout by a sequence of "gammadion" concentric frames of doubling size, as depicted in Figure 4, bottom left. Starting with 1, we double the width of every subsequent frame, until the prescribed layout size is covered. In the end, we trim the rectangles in the outer layer to get the prescribed layout dimension. While this initial pattern features a logarithmic (rather than constant) number of rectangles, it avoids the aforementioned problem, meaning that the final refined pattern will present a smaller error bound for the same number of rectangles (compare Figure 4, right sides).

IX. Implementation and Experimental Results

To test our algorithm, we implemented a prototype using MATLAB (provided as Additional Materials). Our implementation is optimized only in the sense that it exploits the built-in parallelization mechanisms of that suite.

We test our method over 20 natural images taken from a standard benchmark [48] (RGB images, 8 bits per channel).

TABLE II

ANALYSIS OF FOUR IMAGES FROM THE DATA-SET [48] (SEE FIGURE 9) AT DIFFERENT RESOLUTIONS; SEE TABLE V IN THE ADDITIONAL MATERIAL FOR THE REST OF THE DATASET. WE REPORT THE COMPUTATIONAL TIME OF THE ORIGINAL ACE AND FACE (PRE-PROCESSING TIME INCLUDE COMPUTATION OF LAYOUT AND THE SATs). WE ALSO REPORT THE (MAXIMAL AND AVERAGE) ERRORS AS PREDICTED BY OUR METHOD (SEE SEC VII), AND THE ACTUAL (MAXIMAL AND AVERAGE) ERRORS AS MEASURED IN THE OUTPUTS, BOTH AS RAW RGB DISCREPANCIES (IN A SCALE FROM 0 TO 255), AND USING RMSE AND $\Delta E00$ MEASURES [50]

Image	Resolution	Original ACE brute force processing time (s)	FACE processing time (sec) Pre-processing	Main Loop	Total	Expected error max	Expected error avg	Measured error max	Measured error avg	Measured RMSE	Measured $\Delta E00$ max	Measured $\Delta E00$ avg
Angel	30×46	0.192	0.015	0.107	0.122	6.751	0.965	1.732	0.465	0.536	1.138	0.346
	60×91	3.090	0.0004	0.192	0.193	8.427	0.708	2.449	0.331	0.547	1,258	0.354
	100×151	21.660	0.0005	0.462	0.462	9.709	0.533	3.464	0.497	0.586	1.593	0.358
	200×301	360 ($\approx$ 6 min)	0.0005	1.528	1.529	11.179	0.336	3.464	0.447	0.616	1.515	0.386
	300×451	1811 ($\approx$ 30 min)	0.0005	3.707	3.708	11.940	0.255	3.464	0.452	0.573	1.502	0.348
	500×751	17674 ($\approx$ 5 hours)	0.0005	10.425	10.425	13.373	0.174	3.464	0.485	0.591	2.026	0.361
	1000×1502	104338 ($\approx$ 29 hours)	0.035	42.836	42.870	13.542	0.101	4.690	0.846	0.630	2.151	0.382
Athletes	30×46	0.172	0.0005	0.062	0.063	6.752	0.965	2.449	0.407	0.672	1.453	0.289
	60×91	3.061	0.0005	0.165	0.166	8.427	0.708	4.123	0.484	0.697	1.755	0.305
	100×151	21.828	0.0006	0.390	0.390	9.709	0.533	4.123	0.452	0.727	1.741	0.339
	200×301	359 ($\approx$ 6 min)	0.0005	1.510	1.510	11.179	0.336	4.359	0.586	0.807	2.389	0.348
	300×451	1839 ($\approx$ 30.6 min)	0.0005	3.790	3.791	11.940	0.255	4.123	0.649	0.749	2.314	0.350
	500×751	18371 ($\approx$ 5 hours)	0.0006	10.335	10.335	13.373	0.174	4.123	0.529	0.753	2.268	0.341
	1000×1502	104338 ($\approx$ 29 hours)	0.001	41.461	41.462	13.542	0.101	4.690	1.143	0.780	2.715	0.358
Berries	30×40	0.138	0.0004	0.058	0.058	6.754	1.005	3.162	0.410	0.751	1.082	0.337
	60×80	2.29	0.0005	0.140	0.141	8.571	0.743	3.742	0.516	0.703	1.832	0.318
	100×134	16.81	0.0006	0.344	0.345	9.865	0.558	5.385	0.691	0.774	1.716	0.339
	200×267	280.6 ($\approx$ 4.7 min)	0.0005	1.299	1.300	11.418	0.353	5.477	0.820	0.885	1.773	0.383
	300×400	1418 ($\approx$ 23.64 min)	0.0005	2.942	2.942	11.735	0.267	4.583	0.723	0.786	1.906	0.356
	500×667	13615 ($\approx$ 3.7 hours)	0.0005	9.167	9.167	13.216	0.183	6.000	0.782	0.833	2.238	0.378
	1000×1334	92668 ($\approx$ 26 hours)	0.003	37.713	37.717	13.393	0.106	6.403	1.216	0.834	2.565	0.372
Toque	30×46	0.173	0.0004	0.062	0.062	6.752	0.965	5.196	0.693	1.013	1.477	0.464
	60×91	3.11	0.0005	0.159	0.159	8.427	0.708	4.690	0.615	0.903	1.858	0.447
	100×152	22.78	0.0005	0.386	0.387	9.707	0.531	5.196	0.891	0.919	1.993	0.567
	200×303	363.37 ($\approx$ 6 min)	0.0007	1.524	1.525	11.173	0.335	5.196	0.664	1.358	2.120	0.628
	300×454	1849 ($\approx$ 30.8 min)	0.0005	3.848	3.848	11.955	0.255	5.196	0.618	1.145	2.228	0.614
	500×756	18621 ($\approx$ 5 hours)	0.0005	10.384	10.384	13.389	0.174	5.196	0.714	1.135	2.476	0.612
	1000×1511	104963 ($\approx$ 29 hours)	0.001	42.422	42.423	13.556	0.101	6.928	1.573	1.111	2.145	0.516

TABLE III

COMPARISON BETWEEN THE ORIGINAL ACE ALGORITHM, AND FOUR DIFFERENT ALGORITHMS TO APPROXIMATE IT: FACE (OURS), [25], [28] POLYNOMIAL AND [28] INTERPOLATION. THE AVERAGE ERROR AND THE STANDARD DEVIATION IN THE IMAGES HAVE BEEN COMPUTED AS EUCLIDEAN DISTANCE IN RGB SPACE, USING THE RMSE (ROOT-MEAN-SQUARE ERROR) AND $\Delta E00$, AGAINST A GROUND TRUTH ACE IMPLEMENTATION. FOR A FAIR COMPARISON, THE PARAMETERS OF COMPETING METHODS HAVE BEEN SELECTED TO RESULT IN THE LOWEST ERROR. SEE TABLE VI IN THE ADDITIONAL MATERIAL FOR THE INDIVIDUAL MEASURES FOR EACH IMAGE

Method	TotPixel$_{avg}$	Parameters	Time$_{avg}$ (s)	$error$	RMSE	$\Delta E00$
Original ACE brute force	1'417'950	-		*Ground Truth*		
FACE (ours)		rectangles, 100	39.468	1.171 ± 0.223	0.826 ± 0.134	0.414 ± 0.056
Gatta C. et al. [25]		sub-sampling, 8	21	21.760 ± 5.216	14.355 ± 3.346	$4.618 \pm 1,084$
Getreuer P. [28]		polynomial degree, 11	0.710	10.752 ± 2.930	7.019 ± 1.901	2.224 ± 0.565
		interpolation level, 30	1.751	10.290 ± 3.217	6.774 ± 2.094	2.254 ± 0.587
"Ground truth" used in Getreuer P. [28]		interpolation level, 256	14.035	10.349 ± 3.197	6.780 ± 2.082	2.262 ± 0.584

Although intended for a different context (Non Photo Realistic images), it is a good fit for our experiment because it is designed to feature a wide range of visual characteristics, in terms of detail size, texture variation, visual clutter, contrasts, lighting, gradients, etc.

We run our all experiments on consumer hardware (Intel-R Core i7-7700 CPU 3.60 GHz, 16GB RAM). We always used 5 as the value of the slope parameter σ (Equation 3).

To measure the produced error, we compare it against a ground truth, which we computed using the brute-force quadratic approach. This computation is, as expected, extremely slow and takes more than one full day for one high resolution image, on the same hardware setup. The used implementation is also available in the provided code and the resulting images are available in the multimedia materials.

A. Time and Error Evaluation

In one experiment, we run our approximated FACE implementation on images at various resolutions, and measure timings and distances with the ground truth. In this setup, we use layouts targeting a fixed number of 100 rectangles and the minimization of the *average* error. Processing times and measured errors for different images and resolutions are reported in Table II; examples of resulting images are shown in Figures 9, 11, 13 and in Figure 13, in the Additional Materials; all images, including Ground Truths, are attached as additional multimedia materials; spatial error distributions are shown in Figure 9, bottom.

A *qualitative*, visual evaluation indicates that FACE reproduces images that are indistinguishable from ACE, on a wide variety of natural images. Additional visual experiments, specifically targeting boundary cases and challenging inputs,

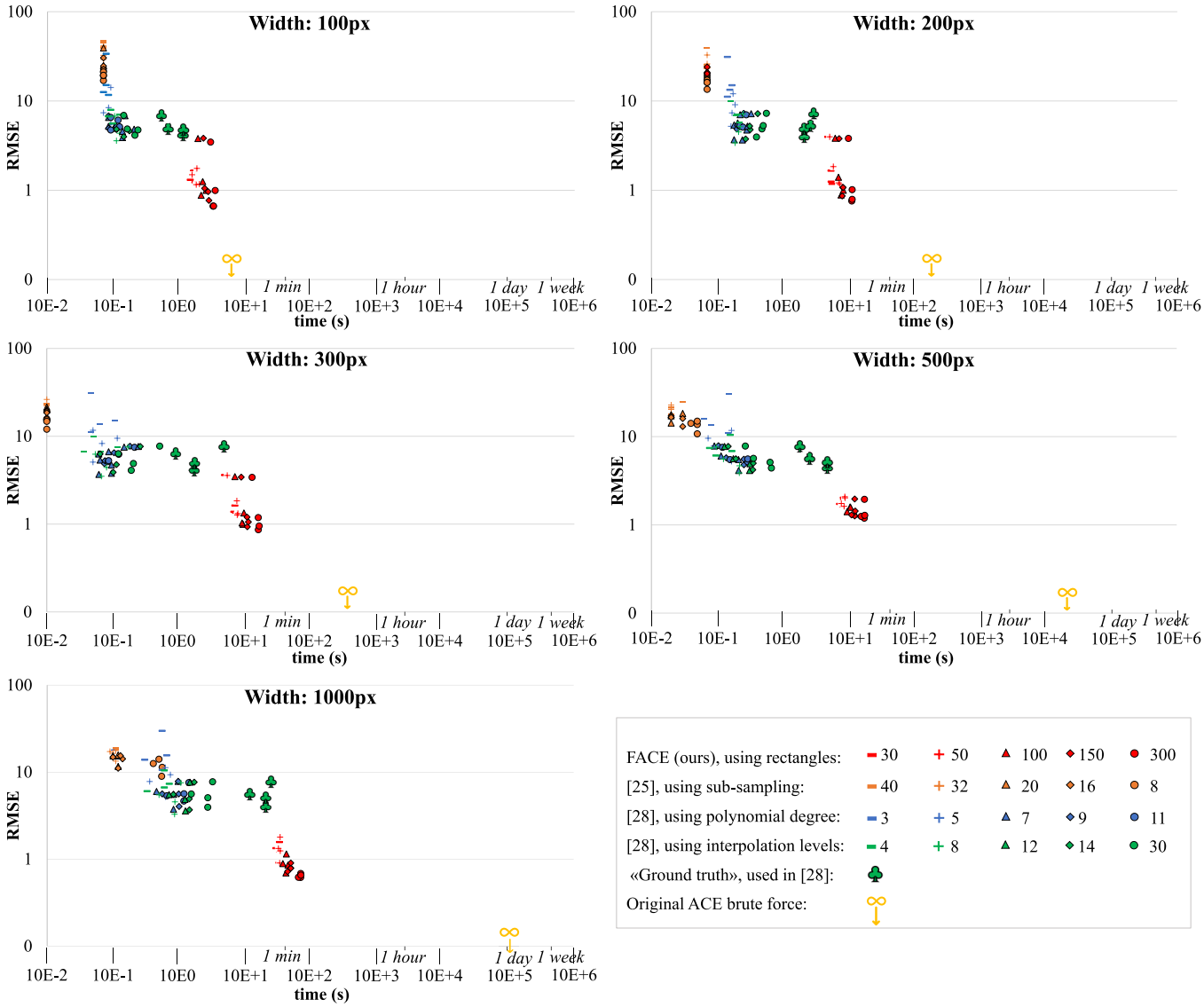


Fig. 10. Scatter plots of time (seconds, horizontal axis) versus error (RMSE, vertical axis), obtained with various ACE approximations methods, including our own (red dots). Both axes are logarithmic. Each symbol corresponds to a test performed over four images from the database [48], which have been re-scaled to different widths (100, 200, 300, 500 and 1000 pixels). For completeness, we also mark, on the horizontal axis, the timings obtained with our straightforward implementation of the original ACE method (having an error 0 by definition, their vertical position is at minus infinite).

reveals that FACE perfectly reproduces desirable behaviours of the original ACE algorithm with images featuring bordering light and dark regions, avoiding halo effects (Figure 12), and color casts (Figure 13).

Quantitative error measurements confirm this, reporting very low $error_{max}$ and $error_{avg}$ (the maximum and average difference) and Root-Mean-Square Error (RMSE) in RGB space.

We also attempt a quantitative estimation of the *perceptual* difference, although produced images are not necessarily intended for direct human consumption (e.g., ACE is used in Machine Visions applications such as [6], [49]). Table III reports the average $DeltaE_{00}$ [50] between the original ACE algorithm and four different algorithms which approximate it. The average $DeltaE_{00}$ [50] between the original ACE and FACE is 0.414 ± 0.056 (1 is often considered a plausible discernibly threshold), the average SSIM value [51] is $0.999\pm$ 0.0004 (on 1), the PSNR is 49.897 ± 1.403, and the average value of HDR-VDP 3.0.6 Quality Correlation measure [52] is 9.999 ± 0.0008 on 10 (see Table VII in the Additional Materials for the individual measures for each image). These results strongly suggest that FACE consistently produces images that are, in most contexts, indistinguishable from ACE.

In Table II, timings go from sub-second for low-resolution images, to sub-minute for high-resolution images. This is, as expected, orders of magnitudes faster than a direct implementation of the original ACE, which is also reported. The table also report the time spent on each sub-phase. Despite being only linear with the number of pixels, SAT construction dominates the computation times. As noted, SAT construction is known to be amenable to GPU specialized speedups, indicating that the performances of our algorithm can potentially be further improved in a more engineered implementation.

TABLE IV
COMPUTATIONAL TIMES OF FACE ON VIDEO SEQUENCES. THE DIGITIZED SUPER8 FILMS ARE THE SAME ANALYZED IN [41], AND THE DIGITAL VIDEOS ARE TAKEN FROM PEXELS STOCK [53]

Digitized Super8 Films			
Title	Frames	Resolution	Processing
Fiat 508 (1931)	888	680 × 576	2.77 hours
La lunga calza verde (1961)	363	720 × 576	1.11 hours
La Ciudad en la Playa (1961)	125	1321 × 1079	1.79 hours
I funerali delle vittime di Piazza Loggia (1974)	261	1228 × 902	1.09 hours
Digital Videos			
Kittens (V01)	305	540 × 960	1.01 hours
Cheetah (V02)	365	640 × 360	0.53 hours
Balloons (V03)	428	960 × 540	1.41 hours
Parrot (V04)	402	540 × 960	1.33 hours

B. Comparing Actual Vs. Predicted Error

Table II also reports data empirically validating our predicted *a priori* error upper-bounds and estimations: the measured error is always lower than its bounds and, as expected, predictions are strongly conservative. An additional sequence of experiments, reported in Figure 7, serves as an empirical validation of the assumptions we use to estimate the expected *average* error (Sec. VII-B), and as an analysis of the actual effect on an error of the parameter which is supposed to control it. As we report, the average error is indeed minimized when the layout is constructed by estimating the per-rectangle average error using an area exponent of 0.5, that is, using Equation (17).

C. Comparison With Competing Methods

In a third set of experiments, we compare FACE against the existing State-of-the-Art approximations of ACE Gatta et al. [25] and Getreuer [28] in terms of efficiency and accuracy. To this end, we use publicly available implementations of competing methods offered by the respective authors. All methods, and our own, are controlled by one parameter to balance between error and speed: for our method, we use the number of rectangles; the method presented in Gatta et al. [25] uses "sub-sampling ratio;" Getreuer [28] comes in two variants, which are assessed individually, each controlled by one parameter: "polynomial degrees" and "interpolation level." We perform a large number of runs, assigning the respective parameter to several values inside suggested ranges.

Results are plotted in Figure 9; timing and quantitative error measures against the ground truth are reported, for the most accurate choice of settings, in Table III. It is important to remark that the error measures originally self-reported in [28] (i.e., *"Ground Truth" used in Getreuer P. [28]*) are measured not against the original ACE algorithm, but against the best possible approximation obtainable with that particular method (an interpolation level of 256 in this case), for any parameter choice (as discussed in II-B); the residual incompressible error, which is also reported in Table III, is substantial.

The overall picture for this set of experiments emerges from the time *versus* RMSE error scatter-plots shown, for images at increasing resolutions, in Figure 10. The experiment evidences that our method results in a discrepancy with the ground truth

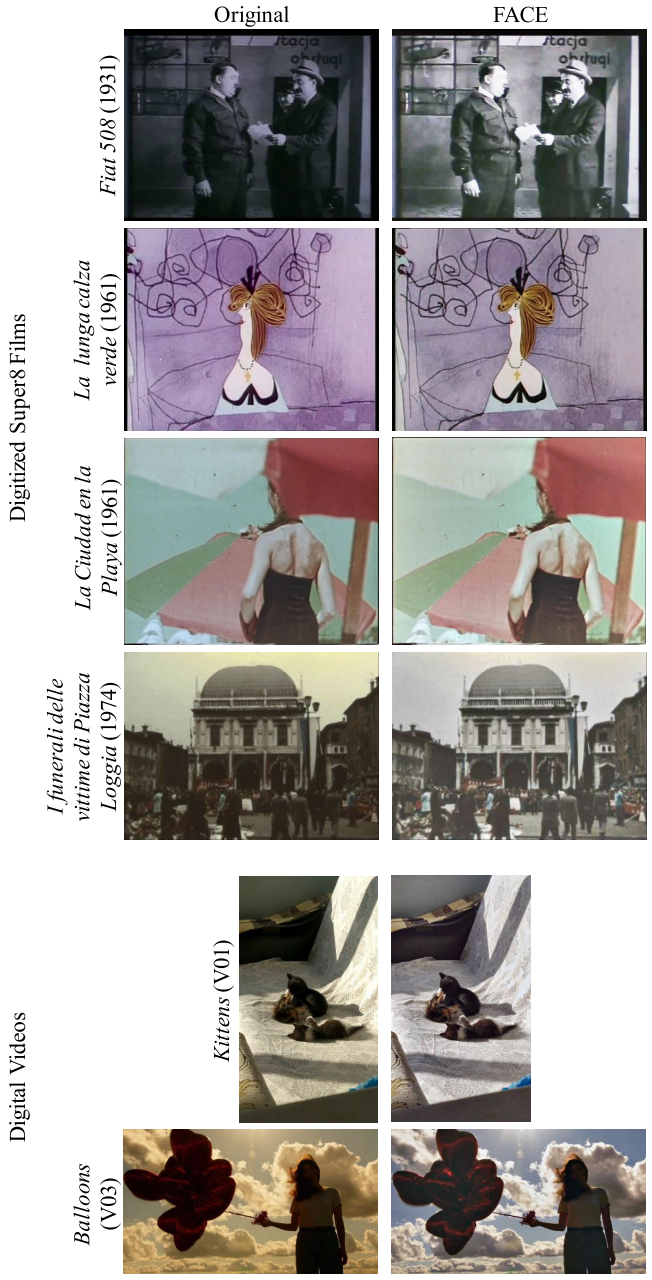


Fig. 11. Frames of the films enhanced through FACE (see Table IV and attached multimedia materials).

which is almost one full order of magnitude smaller than the ones obtainable with any other method, even when they use the settings favoring the best possible accuracy (and worst speed). In other words, our method reaches an accuracy that currently cannot be obtained with any existing method.

The most accurate competing method [28], using its most expensive settings, results in an average value of $DeltaE_{00}$ of 2.224 ± 0.565, on average over the tested images. This suggests that our method is the first one to approximate ACE with an error that is below the threshold of visual discernibly.

The situation is reversed for computation times, which, while still more than three orders of magnitude shorter than exact, brute-force ACE computations, are almost one order of

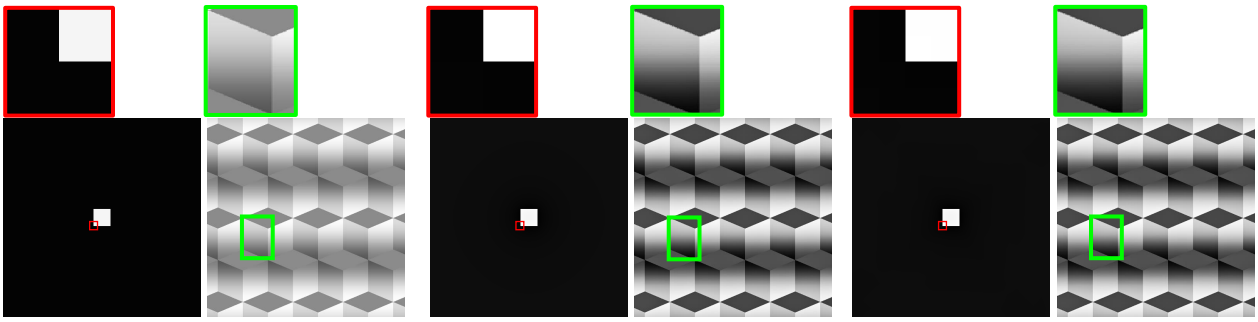


Fig. 12. Results obtained by the original ACE and our FACE approximation over the same test images originally used in [54] to showcase the benefit of ACE over images featuring bordering light and dark regions, while avoiding halo effects. Above: closeups, revealing the ability of our approximation to exactly reproduce this characteristic.

Fig. 13. The results obtained by the original ACE and our FACE approximation over the same test image originally used in [54] to showcase the benefit of ACE over images with color casts (Fig. 13) and noise. Above: closeups, revealing the ability of our approximation to exactly reproduce this characteristic.

magnitude slower than competing methods, or more when they are used with settings favoring speed over accuracy.

In conclusion, being more accurate but also slower, our method does not dominate, nor it is dominated, by any competing method.

D. Application to Video Sequences

Lastly, we successfully test FACE over movies, which, as mentioned, is one of the original motivations for our work (in the context of video restoration). Table IV reports FACE execution times on some digitized videos, and Figure 11 features a few individual frames. Both input and processed movies are available in the attached multimedia materials. Although FACE (and ACE) does not explicitly enforce any form of frame-to-frame consistency, the results do not seem to exhibit, at a preliminary analysis, any obvious artifact in this sense; a more detailed investigation is required to confirm this conclusion.

X. Discussion and Conclusion

The presented FACE algorithm is fully automatic and error bounded. It is controlled with a single parameter, which balances computation time and accuracy.

Experimental evaluations, qualitative and quantitative, empirically confirm that FACE images are in practice indistinguishable from ACE images, and are fit to replace them in basically any context. At the same time, the speedup is drastic, unlocking the practical possibility to adopt this beneficial image enhancement technique in a wider range of scenarios.

Compared to existing ACE approximation strategies, the proposed method is considerably more accurate, but also slower, at least with the current implementation.

In conclusion, we think that our new approximation algorithm can potentially unlock the use of Automatic Color Equalization technique in several application contexts.

To help the reproduction of this work, and to foster further research, we provide our prototype as a reference implementation of FACE (in the attached multimedia materials).

FACE for Video Processing: the achieved speedup makes it possible to use ACE over every frame of a movie or video sequence (see Fig. 11 and additional material). In our algorithm, the optimized layout and the per-pixel normalization factors can be computed, once and for all, for a given choice of parameter and images resolution, and then used for all images at that resolution. This makes our schema an even better fit, for example, to process all frames of a movie (also, to a large collection of pictures shot with the same camera). This is a steppingstone to unlock the use of ACE to video processing,

potentially improving over the current methodologies of color movie restoration and color grading, by providing an automatic enhancement to by colorists and restorers.

Future Works: several aspects of FACE can be improved, for example, designing stricter error bounds, more conservative average error estimations, or better layout-construction heuristics. More aggressive GPU-based optimizations can better exploit the intrinsic parallelism. Finally, while our work unlocks the possibility of a direct use of ACE over video sequences, we still did not experiment with adapting it to this scenario, for example, to take into account inter-frame equalization.

Limitation: poor fit for HDR. The spatial and temporal complexity of FACE, specifically the preparation and the storage of the SAT, is linear with the number of distinct possible intensity values. With high-dynamic-range (HDR) images, the pre-processing time and storage become unfeasible, making FACE unusable, for example, on 16-bit per channel images. In these cases, FACE can be, naturally, applied after tone-mapping. A potential modification to make FACE directly applicable to HDR images is to compute and store SATs only for a subset of the possible intensity values, and interpolate linearly between them during the *main loop*; a problem with this approach is that, due to the non-linearly of function f_σ, it is unclear how to bound or predict the resulting error.

REFERENCES

[1] A. Rizzi, C. Gatta, and D. Marini, "A new algorithm for unsupervised global and local color correction," *Pattern Recognit. Lett.*, vol. 24, no. 11, pp. 1663–1677, Jul. 2003.

[2] A. Rizzi, C. Gatta, and D. Marini, "From Retinex to automatic color equalization: Issues in developing a new algorithm for unsupervised color equalization," *J. Electron. Imag.*, vol. 13, no. 1, pp. 75–84, 2004.

[3] C. Gatta, A. Rizzi, and D. Marini, "ACE: An automatic color equalization algorithm," in *Proc. Conf. Colour Graph., Imag., Vis.*, 2002, pp. 316–320.

[4] A. Plutino, B. R. Barricelli, E. Casiraghi, and A. Rizzi, "Scoping review on automatic color equalization algorithm," *J. Electron. Imag.*, vol. 30, no. 2, pp. 1–32, Apr. 2021.

[5] M. Fierro, H.-G. Ha, and Y.-H. Ha, "Noise reduction based on partial-reference, dual-tree complex wavelet transform shrinkage," *IEEE Trans. Image Process.*, vol. 22, no. 5, pp. 1859–1872, May 2013.

[6] X. Fu, R. Yu, W. Zhang, L. Feng, and S. Shao, "Pedestrian detection by feature selected self-similarity features," *IEEE Access*, vol. 6, pp. 14223–14237, 2018.

[7] D. L. R. Marini, C. Bonanomi, and A. Rizzi, "Perceptual contrast enhancement in visual rendering of astrophotographs," *J. Electron. Imag.*, vol. 26, no. 3, Mar. 2017, Art. no. 031205.

[8] D. L. R. Marini, C. Bonanomi, and A. Rizzi, "Processing astrophotographs using Retinex based methods," *Electron. Imag.*, vol. 28, no. 6, pp. 1–10, Feb. 2016.

[9] D. Gadia, C. Bonanomi, M. Marzullo, and A. Rizzi, "Perceptual enhancement of degraded etruscan wall paintings," *J. Cultural Heritage*, vol. 21, pp. 904–909, Sep. 2016.

[10] E. Roe and C. A. B. de Mello, "Restoring images of ancient color postcards," *Vis. Comput.*, vol. 31, no. 5, pp. 627–641, May 2015.

[11] A. Rizzi and C. Parraman, "Developments in the recovery of colour in fine art prints using spatial image processing," *J. Phys., Conf. Ser.*, vol. 231, Jun. 2010, Art. no. 012003.

[12] P. Zhang and C. Li, "Region-based color image segmentation of fishes with complex background in water," in *Proc. IEEE Int. Conf. Comput. Sci. Autom. Eng.*, Jun. 2011, pp. 596–600.

[13] C. J. Prabhakar and P. U. Praveen Kumar, "Underwater image denoising using adaptive wavelet subband thresholding," in *Proc. Int. Conf. Signal Image Process.*, Dec. 2010, pp. 322–327.

[14] C. Gatta, A. Rizzi, and D. Marini, "Perceptually inspired HDR images tone mapping with color correction," *Int. J. Imag. Syst. Technol.*, vol. 17, no. 5, pp. 285–294, 2007.

[15] S. Paisitkriangkrai, C. Shen, and A. Van Den Hengel, "Strengthening the effectiveness of pedestrian detection with spatially pooled features," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2014, pp. 546–561.

[16] G. Schaefer, M. I. Rajab, M. E. Celebi, and H. Iyatomi, "Skin lesion extraction in dermoscopic images based on colour enhancement and iterative segmentation," in *Proc. 16th IEEE Int. Conf. Image Process. (ICIP)*, Nov. 2009, pp. 3361–3364.

[17] B. R. Barricelli et al., "Ki67 nuclei detection and ki67-index estimation: A novel automatic approach based on human vision modeling," *BMC Bioinf.*, vol. 20, no. 1, p. 733, Dec. 2019.

[18] A. Rizzi and J. McCann, "On the behavior of spatial models of color," *Proc. SPIE*, vol. 6493, pp. 11–24, Jan. 2007.

[19] E. Land and J. McCann, "Lightness and Retinex theory," *J. Opt. Soc. Amer.*, vol. 61, no. 1, p. 1–11, 1971.

[20] A. Rizzi and C. Bonanomi, "Milano retinex family," *J. Electron. Imag.*, vol. 26, no. 3, Mar. 2017, Art. no. 031207.

[21] J. J. McCann, "Retinex at 50: Color theory and spatial algorithms, a review," *J. Electron. Imag.*, vol. 26, no. 3, Feb. 2017, Art. no. 031204.

[22] A. Rizzi, "Colour after colorimetry," *Coloration Technol.*, vol. 137, no. 1, pp. 22–28, Feb. 2021.

[23] A. Artusi, C. Gatta, D. Marini, W. Purgathofer, and A. Rizzi, "Speed-up technique for a local automatic colour equalization model," *Comput. Graph. Forum*, vol. 25, no. 1, pp. 5–14, Mar. 2006.

[24] M. Chambah, C. Gatta, and A. Rizzi, "Linear techniques for image sequence processing acceleration," *Proc. SPIE*, vol. 5667, Jan. 2005, pp. 263–274.

[25] C. Gatta, A. Rizzi, and D. Marini, "Local linear LUT method for spatial colour-correction algorithm speed-up," *IEE Proc. Vis., Image Signal Process.*, vol. 153, no. 3, pp. 357–363, Jun. 2006.

[26] M. Bertalmio, V. Caselles, E. Provenzi, and A. Rizzi, "Perceptual color correction through variational techniques," *IEEE Trans. Image Process.*, vol. 16, no. 4, pp. 1058–1072, Apr. 2007.

[27] M. Bertalmío, V. Caselles, and E. Provenzi, "Issues about Retinex theory and contrast enhancement," *Int. J. Comput. Vis.*, vol. 83, no. 1, pp. 101–119, Jun. 2009.

[28] P. Getreuer, "Automatic color enhancement (ACE) and its fast implementation," *Image Process. Line*, vol. 2, pp. 266–277, Nov. 2012.

[29] Q. Hu, S. Paisitkriangkrai, C. Shen, A. van den Hengel, and F. Porikli, "Fast detection of multiple objects in traffic scenes with a common detection framework," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 4, pp. 1002–1014, Apr. 2016.

[30] M. Mangeruga, M. Cozza, and F. Bruno, "Evaluation of underwater image enhancement algorithms under different environmental conditions," *J. Mar. Sci. Eng.*, vol. 6, no. 1, p. 10, Jan. 2018.

[31] C. Tang, U. F. von Lukas, M. Vahl, S. Wang, Y. Wang, and M. Tan, "Efficient underwater image and video enhancement based on retinex," *Signal, Image Video Process.*, vol. 13, no. 5, pp. 1011–1018, Jul. 2019.

[32] R. Marée, S. Dallongeville, J.-C. Olivo-Marin, and V. Meas-Yedid, "An approach for detection of glomeruli in multisite digital pathology," in *Proc. IEEE 13th Int. Symp. Biomed. Imag. (ISBI)*, Apr. 2016, pp. 1033–1036.

[33] D. Gadia, D. Villa, C. Bonanomi, A. Rizzi, and D. Marini, "Local color correction of stereo pairs," *Proc. SPIE*, vol. 7524, Feb. 2010, Art. no. 75240W.

[34] J. S. Romero, L. M. Procel, L. Trojman, and D. Verdier, "Implementation and optimization of the algorithm of automatic color enhancement in digital images," in *Proc. IEEE Int. Autumn Meeting Power, Electron. Comput. (ROPEC)*, Nov. 2017, pp. 1–6.

[35] M. Chambah, A. Rizzi, C. Gatta, B. Besserer, and D. Marini, "Perceptual approach for unsupervised digital color restoration of cinematographic archives," *Proc. SPIE*, vol. 5008, Jan. 2003, pp. 138–149.

[36] A. Plutino, M. Lanaro, G. Alfredo, R. Cammarata, and A. Rizzi, "ACE for super 8 movie restoration," in *Mathematics for Computer Vision*, vol. 2. Feb. 2018, pp. 15–16.

[37] A. Rizzi, M. Chambah, D. Lenza, B. Besserer, and D. Marini, "Tuning of perceptual technique for digital movie color restoration," *Proc. SPIE*, vol. 5308, Jan. 2004, pp. 1286–1294.

[38] O.-M. Machidon and M. Ivanovici, "Digital color restoration for the preservation of reversal film heritage," *J. Cultural Heritage*, vol. 33, pp. 181–190, Sep. 2018.

[39] A. Rizzi and M. Chambah, "Perceptual color film restoration," *SMPTE Motion Imag. J.*, vol. 119, no. 8, pp. 33–41, Nov. 2010.

[40] A. Plutino, M. P. Lanaro, S. Liberini, and A. Rizzi, "Work memories in super 8: Searching a frame quality metric for movie restoration assessment," *J. Cultural Heritage*, vol. 41, pp. 238–248, Jan. 2020.
[41] B. R. Barricelli, E. Casiraghi, M. Lecca, A. Plutino, and A. Rizzi, "A cockpit of multiple measures for assessing film restoration quality," *Pattern Recognit. Lett.*, vol. 131, pp. 178–184, 2020.
[42] A. Rizzi, C. Gatta, and D. Marini, "Color correction between gray world and white patch," *Proc. SPIE*, vol. 4662, pp. 367–375, May 2002.
[43] E. Provenzi, C. Gatta, M. Fierro, and A. Rizzi, "A spatially variant white-patch and gray-world method for color image enhancement driven by local contrast," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 10, pp. 1757–1770, Oct. 2008.
[44] A. Rizzi, B. R. Barricelli, C. Bonanomi, A. Plutino, and M. P. Lanaro, "Spatial models of color for digital color restoration," in *Conservation, Restoration, and Analysis of Architectural and Archaeological Heritage*. Hershey, PA, USA: IGI Global, 2019, pp. 386–404.
[45] F. C. Crow, "Summed-area tables for texture mapping," in *Proc. 11th Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*, 1984, pp. 207–212.
[46] J. Hensley, T. Scheuermann, G. Coombe, M. Singh, and A. Lastra, "Fast summed-area table generation and its applications," in *Computer Graphics Forum*, vol. 24. Princeton, NJ, USA: Citeseer, Mar. 2005, pp. 547–556.
[47] M. D. Donsker, *An Invariance Principle for Certain Probability Limit Theorems*, no. 6. RI, USA: Memoirs of the American Mathematical Society, 1951.
[48] D. Mould and P. L. Rosin, "A benchmark image set for evaluating stylization," in *Proc. NPAR*, pp. 11–20, 2016.
[49] M. Mathias, R. Benenson, M. Pedersoli, and L. V. Gool, "Face detection without bells and whistles," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2014, pp. 720–735.
[50] C. Oleari, *Standard Colorimetry: Definitions, Algorithms and Software*. Hoboken, NJ, USA: Wiley, 2016.
[51] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.
[52] R. Mantiuk, K. J. Kim, A. G. Rempel, and W. Heidrich, "HDR-VDP-2: A calibrated visual metric for visibility and quality predictions in all luminance conditions," *ACM Trans. Graph.*, vol. 30, no. 4, pp. 1–14, 2011.
[53] B. Joseph, I. Joseph, and D. Frese. *Pexels*. Accessed: Apr. 28, 2022. [Online]. Available: https://www.pexels.com/
[54] B. Funt, F. Ciurea, and J. McCann, "Retinex in MATLAB," in *Proc. Color Imag. Conf.*, 2000, pp. 112–121.

Alice Plutino received the Ph.D. degree from Università degli Studi di Milano in 2021. She is a Postdoctoral Fellow with Università degli Studi di Milano, where she is also an Adjunct Professor with Centro Sperimentale di Cinematografia, teaching digital film restoration and digital media conservation. In 2023, she was awarded a Marie Skłodowska-Curie Postdoctoral Fellowship, which will support her research with the University of Amsterdam. She is the author of the book *Tecniche di Restauro Cinematografico* and several journals and conference papers of national and international relevance. Her research interests include color science, colorimetry, image enhancement, image digitization and archiving, with a particular interest in cultural heritage applications. She is a member of the Italian Color Group (Gruppo del Colore), the Deputy Editor of the *Color Culture and Science Journal* (CCSJ), and the Vice-Coordinator of the Division one and eight of NC CIE Italy.

Marco Tarini received the Ph.D. degree from Università degli Studi di Pisa, in 2003. He is an Full Professor with Università degli Studi di Milano, where he teaches courses on computer graphics and video games. His research activity spans several fields within computer graphics, image and geometry processing, and their applications, including texture mapping, mesh processing, computer animation, real-time rendering, scientific visualization, and video games technologies. He has authored or coauthored more than 50 articles on these subjects on international journals and conferences. He is a Marie Skłodowska-Curie Fellow in 2001. He is also a core developer in many open source projects for libraries and tools, widely used by the scientific community. He received several awards, including the Young Researcher Award by Eurographics in 2006, for these activities. He serves as an associate editor, the chair, and an IPC/ITC member for several journals and conferences.

Open Access funding provided by 'Università degli Studi di Milano' within the CRUI CARE Agreement