https://doi.org/10.2352/ISSN.2470-1173.2017.18.COLOR-028

Gamut Mapping in RGB Colour Spaces with the Iterative Ratios Diffusion Algorithm

Carlo Gatta, Scientific Consultant, Barcelona, Spain
Ivar Farup, Professor, NTNU – Norwegian University of Science and Technology, Gjøvik, Norway

Abstract

Different reproduction devices can have different sets of reproducible colours. These sets are called gamuts. The process of transforming colours from one device (or image) gamut to another is called gamut mapping. Gamut mapping has many technical issues to be considered: the used colour space, direction and magnitude of the mapping and whether and to which extent in-gamut colours should be altered. Spatially invariant algorithms treat all the pixels independently on their position in the image. Spatially variant (local) algorithms allows a better rendition but introduces the problem of artefacts and/or haloing in the resulting image. In this paper we propose a spatially variant gamut mapping algorithm that creates virtually no artefacts nor haloing in the resulting image. We start from an analysis of the Retinex algorithm and devise proper functionals to build an algorithm which tries to maintain spatial ratios in the image while mapping it into the gamut and, at the same time, avoids all drawbacks of Retinex approaches. We suggest to perform the mapping in an RGB colour space rather than one of the perceptually more homogeneous ones. Although less homogeneous, we experimentally show that RGB colour spaces actually have better hue constancy according to a certain criterion.

Introduction

Colour gamut mapping denotes the process of adjusting a set of colour data – typically from an image – to fit the colour gamut of a given destination device. Historically, gamut mapping algorithms were pure colour mappings, i.e. functions between colour spaces, independent of the content of the image to be reproduced [1]. Such gamut mapping algorithms by now constitute fairly mature technology, and the main work in this area is now on standardising representations and algorithms [2].

Today, the research on gamut mapping focuses more on other dimensions, such as the reproduction of high dynamic range images on conventional devices [3], multispectral images [4], and even goniochromatic reproduction [5]. The most active area of research on gamut mapping the later years has been the exploitation of spatial properties of the image (see, e.g., [6]). In spatial gamut mapping, there is no longer a one-to-one mapping of colours from the input gamut to the output gamut. The local context of the image is taken into account such that not only the colour, but simultaneously also the texture and the local contrast is reproduced as accurately as possible. There are two main approaches to spatial gamut mapping: In the first approach, information lost in the conventional mapping process is added back to the image, either once or iteratively [7]. In the second approach, a cost function measuring the perceptual error between the gamut mapped image and the original is minimised with respect to either the final image [8], or the parameters of the gamut mapping algorithm [9].

It is an established fact that such spatial colour gamut mapping algorithms can be superior to the conventional algorithms for many types of images [10]. However, the behaviour of the algorithms is still somewhat unpredictable in that they can produce unwanted artefacts such as hue shifts and halos for certain images. In a previous work [7], we introduced specific constraining function in order to reduce these artefacts. Here we propose a new approach. First, we suggest to perform the mapping in an RGB colour space rather than one of the perceptually more homogeneous ones. Although less homogeneous, we argue that RGB colour spaces actually have better hue constancy according to a certain criterion. Secondly, we avoid haloing by introducing an iterative diffusion scheme motivated by retinex type algorithms [11, 12] using a $1/r$ dependency of spatial distance in the image.

On the Use of RGB Colour Spaces for Gamut Mapping

Having unitary steps in a colour space that can represent unitary perceived difference between colours is a very relevant property of a colour space w.r.t. the gamut mapping main goal. This property is named perceptual homogeneity. The studies of McAdams have shown that, based on his experimental data, to build a perceptually homogeneous colour space, the colour space should have a dimension greater than 3, or the 3D space should be curved. More than one attempt has been done to build Euclidean 3D colour spaces that approximate perceptual homogeneity. Some example are CIELAB, CIELUV, OSA UCS Ljg. These colour spaces are extremely important and successful for practical use; however, they cannot be considered perfectly perceptual homogeneous. In almost all the papers on gamut mapping, homogeneity is the property that is mandatory for the proposed methods to work properly. The work in [13] by McCann is the most notable exception.

To overcome the problem of having a perceptually homogeneous colour space for gamut mapping three ways can be followed: (1) use differential geometry to build a real perceptually homogeneous colour space with appropriate bijection to convert colours from and back to CIEXYZ [14]; (2) assume that perceptual based colour spaces, like e.g. Munsell book, are perceptually homogeneous and isotropic and then build the appropriate functions or look up tables to convert from and back to CIEXYZ; (3) design a gamut mapping algorithm that does not require a perceptually homogeneous colour space.

This work builds on the third strategy, proposing to perform gamut mapping in linear RGB colour spaces. The main reasons for choosing a linear RGB colour space against other colour

spaces are:

1. The RGB colour space is the basis for ratio-based computational models of colour appearance, like e.g. Retinex [15]. In this case, we can modify ratios between values without using perceptual unitary steps thus without needing a perceptually homogeneous colour space.
2. The RGB colour spaces have constant hue half-planes that are mandatory for gamut mapping since one of the paramount rules is to preserve hue. These half-planes are not perfect, as for other colour spaces derived from XYZ; nonetheless, they provide a good approximation.

The next subsection provides empirical evidence that linear RGB constant hue half-planes are actually sufficiently good to be used in a hue preserving mapping strategy.

Constant Hue Half-Planes in CIELAB, RGB and CIECAM02

Our proposal needs a colour space that has half-planes that represent constant hue colours. Since the majority of gamut mapping algorithms work in the CIELAB colour space, and that CIECAM02[16] is proposed as a colour appearance model with potential applications to colour management[17], it is mandatory to show that the RGB constant hue half planes are good enough or better than constant hue half-planes of CIELAB and CIECAM02.

To this aim, the Munsell book can be considered an optimal source for hue constancy data since it has been proved independently in [18] and [19] that, when combined with a spatially invariant gamut mapping algorithm, the Munsell book performs better than other colour spaces.

We used the Newhall, Nickerson and Judd's data of the Munsell book, which has 40 pages that represent constant hue half-planes. Assuming that these half-planes are correct, CIELAB, RGB and CIECAM02 can be compared in a quantitative way. The 2745 Munsell chips have been transformed to the corresponding CIEXYZ values, then linear RGB triplets are obtained (using sRGB primaries) by the following transformation:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 3.240 & -1.537 & -0.498 \\ -0.969 & 1.876 & 0.0415 \\ 0.0556 & -0.204 & 1.057 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (1)$$

Then, we removed the chips that are out of the RGB cube, obtaining 1842 chips. Starting from RGB triplets, two chromatic coordinates can be obtained as follows: (1) project RGB triplets onto a plane $P_\perp$ that is orthogonal to the gray axis $\vec{g} = [1,1,1]$; (2) project these values to two orthogonal axis that belong to $P_\perp$. In this way, two chromatic coordinates (*RGBa* and *RGBb*) representing the hue of a colour in the RGB colour space are defined[1].

The CIECAM02 chromaticity attributes a and b has been calculated starting from CIEXYZ tristimulus following the paper [16], without applying the chromatic adaptation[2], with the following parameters: $L_A = 100$, $Y_b = 100$, $Y_w = 100$.

If we accept the Munsell book as the source of perfect hue constant planes, we should expect that, for a given Munsell page and a Munsell value, a colour space with perfect hue half-planes should place the Munsell chips on a line that points toward the achromatic axis. The more the colours deviate from a perfect straight line, the less suitable is a colour space for a gamut mapping that projects colours toward the achromatic axis. We use this idea to measure how good are CIELAB, CIECAM02 and RGB. For each Munsell page and Munsell value we compute the average inter-colours angular difference in the three spaces.

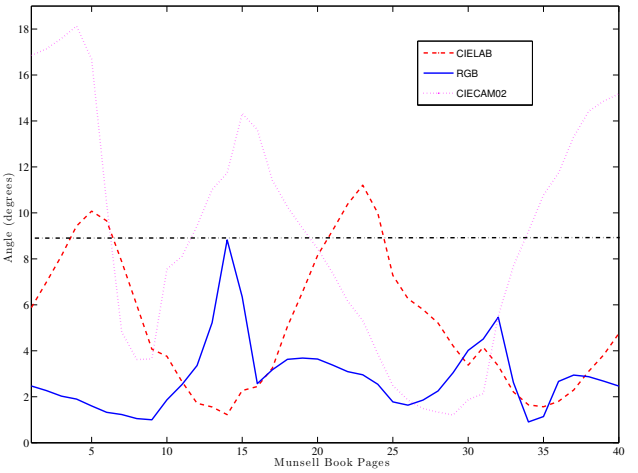

Figure 1. *Angular hue error as a function of Munsell Hue Pages*

For the sake of simplicity, then we averaged these numbers for the same Munsell Hue. Figure 1 shows these numbers for CIELAB, RGB and CIECAM02 on 40 Hue pages. The numbering of pages is from 1 (2.5R) to 40 (10RP) in steps of 2.5 Munsell Hue. The dashed, full and dotted lines represent average angular errors (in degrees) produced by CIELAB, RGB and CIECAM02 respectively, averaged on Munsell Value. The horizontal line represents the 9° limit, that is the resolution of the used Munsell chips in terms of Hue. It can be noticed that RGB performs better than both CIELAB and CIECAM02. In particular the CIELAB curve slightly overcomes the 9° boundary for pages from 4 to 6 (10.0R to 2.5YR) and from 21 to 24 (2.5BG to 10.0BG). These data confirm the well known hue shift of CIELAB in the regions of red/orange and blue/purple. Results from CIECAM02 are surprisingly not as good as both CIELAB and RGB.

Following this results, if a perceptually homogeneous colour space is not necessary for the purposes of gamut mapping, RGB performs better than both CIELAB and CIECAM02.

From Retinex to the Gamut Mapping Iterative Ratios Diffusion Algorithm

In this paper we develop an algorithm that, taking inspiration from Retinex-based colour appearance models, performs gamut mapping in a novel way. The main idea we follow is that local ratios between pixels values in the RGB colour space are important features for image appearance, as Land demonstrated in his excellent experimental investigation [15] [20]. This approach to gamut mapping has, in our knowledge, only one important predecessor [13].

In this section we start from the analysis of the Retinex algorithm proposed by Provenzi et al. in [21, 11] to understand clearly which properties of a Retinex algorithm are suited (or not) for the

[1] These two chromatic coordinates, despite the name, are not correlated with the CIE a^* and b^* coordinates

[2] Omitting this step of the CIECAM02 does not change the hue constancy property.

gamut mapping goals. Because of the technical depth and the mathematical rigorous treating of these two papers, we can consider the results they present as a milestone. The work in [21] provides proofs and experimental results on some expected Retinex behaviours and proves some well known conjectures about the Retinex algorithm. The work in [11] investigates the local behaviour of Retinex proposing a unified point of view to analyse and exploit locality in perceptually inspired algorithms.

To avoid confusion, with the word Retinex we refer only to algorithms that derive from the original Land approach in [15] and not to other centre/surround Retinex (e.g. [22]) derived from the idea proposed by Land in [23]. In this section, all the claims regarding Random Spray Retinex (RSR) are excerpted by the above mentioned papers. To better understand the properties of RSR we will state at least its mathematical formulation. Retinex algorithms compute *lightness* of every single pixel i collecting data potentially from the whole input image. The employed strategy to sample pixels in the image can be very different from algorithm to algorithm. RSR samples data from the image by using a Random Spray centred in the pixel i. More formally, the spray is obtained as follows (mainly excerpted from [11]).

We can describe a spray as a collection of pixels coordinates. If (i_x, i_y) are the spatial coordinates of i, we can define the coordinates of a generic pixel $j \equiv (j_x, j_y)$ belonging to the spray $S_k(i)$ in this way:

$$\begin{cases} j_x = i_x + \rho \cos(\theta) \\ j_y = i_y + \rho \sin(\theta) \end{cases} \tag{2}$$

where $\rho \in \text{RAND}_n[0,R]$, $\theta \in \text{RAND}_n[0,2\pi)$. Here $\text{RAND}_n[0,x]$ is a uniform pseudo-random generator of values in the interval $[0,x]$ and R is a constant that represents the maximum radius of the Spray, which is set to the image diagonal length.

Generating N sprays for every pixel i, the RSR computation can be described as:

$$L_c(i) = \frac{1}{N} \sum_{k=1}^{N} \frac{I_c(i)}{I_c(x_{c,H_k})} \tag{3}$$

where I_c represent the input intensity values of the image in the channel c of the RGB colour space, L_c the respective computed *lightness* image and x_{c,H_k} is the pixel with highest intensity in the spray $S_k(i)$, for every $k = 1,\ldots,N$.

It can be noticed that the RSR algorithm essentially computes the lightness of a pixel by taking the ratio of its intensity and the intensity of the highest pixel in a neighbourhood described by the spray geometry. The highest pixel is named *local maximum*. The operation is repeated N times to reduce the noise due to random distribution of the Spray.

Our goal here is to understand which properties of RSR are potentially good for the goal of gamut mapping. Summarising, the papers [21, 11] show (or in some cases demonstrate) the following properties of Retinex and/or RSR:

A) Retinex and RSR computes *lightness* separately on the three RGB colour channels.
B) Retinex and RSR computation never decreases pixel values. It can be easily mathematically derived from the analysis of Equation (3).
C) Multiple applications of RSR (the lightness output is used as intensity input in the next application) reach quickly (typically 10-20 applications) a convergence point in which further applications do not change the image. The image of convergence is characterised by heavy white speckling noise.
D) Locality in Retinex can be exploited by 2D pixel spray. The spray density in RSR decreases as $1/r$, where r is the Euclidean distance from the spray centre (see Equation 2). This Spray radial distribution performs better respect to other spray densities in terms of colour correction, contrast treatment and detail enhancement [11]. The possible noise introduced by RSR is mainly due to the randomness of the 2D sampling spray geometry. Another important source of noise is the presence of flat areas (constant colour) in the input image.

The following subsections discuss these properties and claim the choice we took in developing the Gamut mapping Iterative Ratios Diffusion Algorithm.

Retinex Independent RGB Computation

The first Retinex property is very well known since its definition [15]. This fact, together with other Retinex properties, gives Retinex the ability to remove an over-imposed unknown colour cast if present. This is a negative property for gamut mapping since one of the most important goals of gamut mapping is to preserve the input image hues. For this reason, we will treat the relative luminance and chrominance components of the colour separately.

Retinex Never Decreases Values

The second property is good for gamut mapping, since the gamut of a printer in the linear RGB colour space usually misses dark tones while the triplet $RGB = (1,1,1)$, working in relative colorimetry, always belongs to the printer gamut. Since we decided to work separately on luminance and chrominance, the Retinex "toward white" behaviour has to be modified in "toward maximum luminance". The fact that Retinex never decreases pixel values means that they can be left unchanged in some cases. This is unacceptable for gamut mapping, since final image colours must be in gamut.

Convergence Point of Multiple Retinex Applications

As stated above, multiple applications of RSR generate an image of convergence. This condition is reached when, separately in the R, G and B colour channels, every random spray samples at least one pixel with maximum intensity, namely $I_c(x_{c,H_k}) = 1$. It is easy to understand that, under this condition, the image has a lot of white and/or maximum saturated red, green, and blue pixels. This convergence image is unacceptable for gamut mapping.

Locality in Retinex

The fourth property gives us important hints on how to implement the concept of local ratios. However, this point suggests that using random sprays can raise the problem of enhancing high frequency noise if present. In the algorithm we propose, ratios between pixels intensity are not computed directly as in the Retinex

algorithm, but diffused isotropically.

The informal discussion in previous subsections shows some important guidelines in developing a Retinex-inspired Gamut Mapping algorithm. The next section will describe how these guidelines have been formalised and implemented. Prior to this, we briefly claim our final decisions:

- Sequential mapping: relative luminance first using ratios, then clipping saturation.
- The direction of mapping is toward 1 during luminance treatment, and toward the gray axis during saturation treatment (on constant hue half-planes maintaining the previously computed luminance).
- Use of a diffusing function that decreases as $1/r$, where r is the Euclidean distance between pixels.

G-IRDA: The Gamut Mapping Iterative Ratios Diffusion Algorithm

One of the important characteristics of the proposed method is that a change to a pixel colour is propagated to its neighbourhood trying to preserve the ratios values. To this aim, we developed an iterative algorithm that changes pixels values according both to the fact that they are out-of-gamut and that some neighbourhood pixels values have been changed in the previous iteration.

The main reason we change pixels values is that some pixels colours are out of gamut. For this reason we need an operator that discriminates in/out-of-gamut for every pixel in the image. The following Q operator is designed to satisfy this requirement:

$$Q_p(I_p(x,y)) = \begin{cases} 0 & \text{if} \quad I_p(x,y) \text{ is in-gamut} \\ 1 & \text{if} \quad I_p(x,y) \text{ is out-of-gamut} \end{cases} \tag{4}$$

where the index p indicates the iteration step and the pair (x,y) is the discrete spatial coordinates in the image support $\Im$.

As stated before, we want that changes in the previous iteration are diffused to neighbourhood pixels in the current iteration. To obtain this we perform the convolution between the magnitude of the changes applied in the previous iteration p and a convolving function d:

$$M_p(I_p, I_{p-1}, d) = \|I_p - I_{p-1}\| * d \tag{5}$$

where $\|\cdot\|$ is the L2 norm and $*$ denotes the two dimensional convolution operator, and I_1 is the input image. The norm $\|I_p - I_{p-1}\|$ represents the magnitude of the changes applied to every pixel in the last iteration[3].

It is now straightforward to understand the important role that d plays in the computation, since the degree and shape of diffusion depends on it. As stated above, we want to diffuse changes in the neighbourhood with a function that decrease as $1/r$ where r is the Euclidean distance between the origin $(0,0)$ of the convolving function and a pixel coordinates (x,y). To keep the general form of IRDA, we set $d = d_E$, where d_E is the following discrete convolving function:

$$d_E(x,y) = \begin{cases} 0 & \text{if} \quad (x,y) = (0,0) \\ k_E^{-1} \frac{1}{\sqrt{x^2+y^2}} & \text{if} \quad (x,y) \neq (0,0) \end{cases} \tag{6}$$

[3]For $p = 1$, $\|I_1 - I_0\| = 0$ by definition, since I_0 is not defined.

where k_E is such that $d_E(x,y)$ has unitary integral. It is important that $d_E(0,0) = 0$ since a change to a pixel value has to be propagated to the neighbourhood pixel but not to the pixel itself. If not differently specified d_E is the convolving function used by the gamut mapping algorithm developed in the next section.

Now that we have the Q operator to identify in and out of gamut pixels and the M operator that, by means of the convolving function d, diffuses changes of previous iteration to neighbourhood pixels, we can state the iterative formula of our proposal, called IRDA (Iterative Ratio Diffusion Algorithm), omitting the spatial variables for clarity:

$$I_{p+1} = I_p + \alpha V \left((1-\beta) Q_p + \beta \frac{M_p}{\alpha} \right) \tag{7}$$

where V is the direction of mapping, $\left((1-\beta) Q_p + \beta \frac{M_p}{\alpha} \right)$ is the magnitude of mapping for every pixel, scaled by the constant value $\alpha \in (0,1]$. We intentionally do not discuss the matrix V. It will be done extensively later. Here $\beta \in [0,1)$ is the algorithm parameter that controls the diffusive behaviour of M. The setting $\beta = 1$ must be avoided since it rules out the contribution of the clipping, thus vanishing the gamut mapping component of the algorithm. The division by α applied to M operator is important since the quantities in M derive from the previous iteration that was scaled in magnitude by α. Thus, the division by α here assures that the quantities in Q and M are comparable, and thus the β parameter in this case implements a convex linear combination between Q and M since $Q \in [0,1]$ and $M \in [0,1]$. Further discussion on this point, showing it's importance, will be provided later. As it is stated, Equation (7) propagates magnitude difference and not ratios. However, with a simple mathematical trick, magnitude of ratios changes can be propagated easily; as shown later in Section .

The direction of mapping is defined only by the matrix V, while the magnitude is decided by the Q and M operators through β. The constant value α can be seen as the step Δt of a numerical approximation algorithm. In fact, high values of α correspond to fast convergence but lower quality while small values give better quality but slower convergence.

As every iterative (or recursive) algorithm, we need to set the stop condition. In the case of gamut mapping we need that the resulting image is completely in-gamut, thus the stop condition is

$$\forall (x,y) \in \Im, \; Q_p(x,y) = 0 \tag{8}$$

It can be demonstrated (see Appendix) that, under the following conditions, the algorithm converges: the matrix V, defining the direction of mapping, has to be devised such that the pixels colours are directed toward the destination gamut, and, at the same time, $\|V\| \leq 1$.

Being an iterative algorithm, the parameter α is very important to the final result. Analysing Equation (7), it is possible to note that the maximum magnitude of mapping during an iteration is α times the norm of vector valued matrix $\vec{V}$. We can impose that the maximum magnitude at each iteration is fixed to a specific value. Since the maximum norm of V must be 1, the optimal setting of α comes from the precision we ask the iterative algorithm to work. Considering an image quantised with 2^n steps, the optimal value is $\alpha = 2^{-(n+1)}$ so that the maximum change during an iteration is less or equal than half the quantisation step. Thus,

we can guarantee at least the sufficient precision before the final re-quantisation of data prior to sending to printer or other devices. Using RGB images with 8 bits per channel, the optimal value for α is 0.0019531. Smaller values can provide even better results at the cost of an higher number of iterations before convergence.

Sequential Gamut Mapping

We can now address more in detail the technical problems of gamut mapping. As stated above, we perform gamut mapping in two separate steps: first we treat relative luminance Y, then saturation. This decision is important in our proposal for two reasons: firstly it permits not to change the hue, even though we are taking inspiration from Retinex, secondly it avoids the possible inversion of lightness typical of the one step mappings directed toward the centre of gamut. This second motivation is significant since the luminance component is very important for details perception and for the global contrast of the image. This problem has been addressed by Farup et al. in [7] in Section VI.C. From their analysis, we have to take a critical decision: either preserve luminance gradients or preserve saturation gradients. We decide to preserve luminance gradients since they are far more important than saturation gradients.

Diffusing Relative Luminance Ratios

Figure 2 shows the first part of the G-IRDA algorithm. The algorithm takes as input a sRGB coded input image $\vec{I}_{sRGB}$ and produces an intermediate linearly coded RGB image $\vec{R}_{RGB}$, where the Y component has been treated by the IRDA algorithm to achieve gamut mapping goals. To clearly describe and define all the details of the algorithm the following subsections describe every block of the scheme in Figure 2.

We perform a standard gamma decoding assuming $\gamma = 2.2$. The following projection is used to have CIE Y values from linearly encoded RGB values:

$$I_Y = 0.212 I_R + 0.715 I_G + 0.072 I_B \tag{9}$$

where R, G and B index indicate the respective colour component of image $\vec{I}_{RGB}$.

The IRDA algorithm is based on differences and sums. A way to treat Y ratios is to convert the data from linear to logarithmic and then use IRDA as it is. This is done using the following property of the logarithm function: $\ln(a/b) = \ln(a) - \ln(b)$, and exponentiating at the end of the IRDA computation. The two blocks "Log" and "Exp" are computed by, respectively, $l : [0,1] \mapsto [-1,0]$ and $e : [-1,0] \mapsto [0,1]$:

$$l(I_Y(x,y)) = \frac{\ln(I_Y(x,y) + \varepsilon) - \ln(K_1)}{K_2} \tag{10}$$

$$e(R_{logY}) = K_1 e^{(K_2 R_{logY})} - \varepsilon \tag{11}$$

where ε is a small value to avoid the logarithm of zero, $K_1 = 1 + \varepsilon$ and $K_2 = -ln(\varepsilon) + \ln(1+\varepsilon)$ are constants that depend only on ε. These constants assure that $e(l(z)) = z, \forall z \in [0,1]$.

To clearly define the behaviour of the IRDA algorithm we need to define V_Y, α and β. While α and β are constants that can be tuned, V_Y has to be devised univocally. It is important to note that, since I_{logY} is a scalar matrix, V_Y is a scalar matrix too. The following formula states how to devise V_Y:

$$V_Y(x,y) = \ln(1) - I_{logY}(x,y); \tag{12}$$

Here, thanks to K_2, $I_{logY}(x,y) \in [-1,0]$ and consequently $V_Y(x,y) \in [0,1]$. This ensures that $|V_Y(x,y)| \leq 1$. The other condition to have a convergent IRDA algorithm is that $V_Y(x,y)$ is devised to move pixels colours toward the destination gamut. In relative colorimetry, $Y = 1$ is in gamut, and thus the mapping toward $\ln(1) = 0$ as indicated in Equation (12) guarantees the second convergence condition.

To clarify the behaviour induced by the definition of V_Y in Equation (12), we can rewrite the IRDA Equation (7) substituting the generic vector valued matrix $\vec{V}$ with the specific case of V_Y:

$$R_{logY,p+1} = R_{logY,p} + \alpha V_Y \left((1-\beta) Q_p + \beta \frac{M_p}{\alpha} \right) \tag{13}$$

where $R_{logY,0} = I_{logY}$. Substituting V_Y with its definition (see Equation (12)) and assuming $((1-\beta)Q_p + \beta \frac{M_p}{\alpha}) = 1$ for simplicity, we obtain:

$$R_{logY,p+1} = R_{logY,p} + \alpha \ln(1) - \alpha I_{logY} \tag{14}$$

Exponentiating both sides of this formula and using the properties of exponential function we have:

$$e^{R_{logY,p+1}} = e^{R_{logY,p}} \frac{e^{\alpha \ln(1)}}{e^{\alpha I_{logY}}} \tag{15}$$

Here the index $logY$ denotes quantities that are logarithmic encoded, thus we can rewrite the formula using the respective linear quantities, adding spatial coordinates (x_i, y_i) of a generic pixel i:

$$R_{Y,p+1}(x_i,y_i) = R_{Y,p}(x_i,y_i) \left(\frac{1}{I_Y(x_i,y_i)} \right)^{\alpha} \tag{16}$$

This formula clearly shows that during the IRDA computation, at each iteration, every pixels value $R_{Y,p}(x_i,y_i)$ is multiplied by the constant quantity $(1/I_Y(x_i,y_i))^{\alpha}$ that depends only on $I_Y(x_i,y_i)$ and α. This quantity is always greater than one since $I_Y(x_i,y_i) \in [0,1]$ and $\alpha > 0$. This respects the Retinex property of never decreasing pixel values.

Moreover, this formula can be seen as an iterative version of the Retinex formula in Equation (3) where the spray sampling has been removed and $I_Y(x_i,y_i)$ plays the role of the local maximum. Thus, the locality of IRDA is devised using the diffusion operator instead of sampling the neighbourhood pixels. Moreover, the spatially variant nature of IRDA is not present in formula (16) because, for simplicity, we removed the diffusion component in the step between Equations (13) and (14) without loosing validity of the mathematical explanation.

Once the grayscale gamut mapped image R_Y is obtained, we need to re-add the chromaticity component of the input image. Since the chromatic component is two dimensional we need to extend to the RGB colour space the grayscale images R_Y and I_Y as following:

$$\vec{R}^{\text{gray}}_{RGB} = R_Y \vec{v}_R + R_Y \vec{v}_G + R_Y \vec{v}_B \tag{17}$$

$$\vec{I}^{\text{gray}}_{RGB} = I_Y \vec{v}_R + I_Y \vec{v}_G + I_Y \vec{v}_B \tag{18}$$

where $\vec{v}_R$, $\vec{v}_G$ and $\vec{v}_B$ are the unitary vectors representing respectively the red, green and blue axis. Then, to re-add the chromaticity we simply substitute the Y component of input image with the new Y component of R_Y as follows:

$$\vec{R}_{RGB} = \vec{I}_{RGB} - \vec{I}^{\text{gray}}_{RGB} + \vec{R}^{\text{gray}}_{RGB} \tag{19}$$

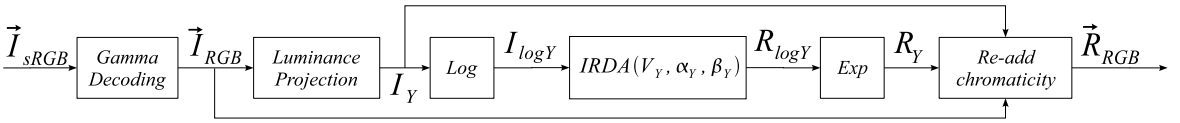

Figure 2. *A schematic view of the Y treatment of the G-IRDA algorithm.*

Clipping Saturation

The second step of the algorithm modifies the saturation of the image $\vec{R}_{RGB}$ to achieve an in-gamut output image $\vec{O}_{RGB}$. As stated above we want to modify saturation without changing both the hue and the previously computed relative luminance R_Y. To this aim we perform the clipping toward the achromatic axis, on lines orthogonal to the Y axis. We called this clipping $Y_\perp$. The study performed in Section ensures that the hue will be modified less or as much as algorithms that employ constant hue lines in CIELAB. However, depending on the shape of the destination gamut, the $Y_\perp$ can decrease the saturation too much, especially for very dark colours. To favour saturation instead of intensity, the clipping could be performed toward the gray axis on lines orthogonal to it; we name this clipping as $G_\perp$. In the result section we show some comparison between this two clipping strategies.

After the clipping, the output in-gamut image $\vec{O}_{RGB}$ can then be encoded in *sRGB* or alternatively linearly or non-linearly transformed into a desired colour space.

Speed-Up via Multi-Resolution Processing

The computational cost of the proposed method is $\mathcal{O}(KN\log_2(N))$, where K is the required number of iterations and N is the number of image pixels. The cost of each iteration is dominated by the filtering, which is $\mathcal{O}(N\log_2(N))$ since we use the Fast Fourier Transform (FFT) to compute it. A way to speed up the method is to increase the α value, which unfortunately causes a degradation of the final result as demonstrated later in Subsection . Instead, we use a classical multi-resolution approach where the input image is subsampled 2^S times and gamut mapped using IRDA; subsequently, high-frequency content is added to the upscaled mapped image and, again, mapped using IRDA. This repeats until a full resolution image is obtained. We performed various experiments[4] with a set of 10Mpixels images and measured the speed up and degradation of results compared to the single scale algorithm, measured in average ΔE^*_{ab} error[5]; results are summarised in Table 1.

Discussion of the G-IRDA Algorithm

Estimation of Induced Ratio Errors

It is important to understand how and how much can the proposed approach induce ratio errors between pixel values during the computation. It has been already discussed that, for the purpose of gamut mapping, it is impossible to preserve all the ratios between Y values. To give an estimation on how the algorithm induces ratios errors to the image and how the α parameter can influence this error, the following proof, using a simplified notation, is presented.

[4]The algorithm is implemented in MATLAB. Experiments have been conducted using a 2,3 GHz Intel Core i7. Using GPU-based FFT and/or a more performing language, we suppose another order of magnitude speed-up could be easily achieved.

[5]The JND (Just-Noticeable Difference) corresponds to $\Delta E^*_{ab} \approx 2.2$

Let a_0 and b_0 be scalars representing Y values of two pixels and $\delta_0 = a_0/b_0$ their initial ratio (here the subscript denotes the iteration step p). After one iteration of the IRDA algorithm, as described in Equation (16), and thus without the diffusion part, a_1 and b_1 will be:

$$a_1 = a_0\left(\frac{1}{a_0}\right)^\alpha, \quad b_1 = b_0\left(\frac{1}{b_0}\right)^\alpha. \tag{20}$$

The ratio at iteration $p = 1$ is:

$$\delta_1 = \frac{a_1}{b_1} = \left(\frac{a_0}{b_0}\right)^{1-\alpha} = \delta_0^{1-\alpha} \tag{21}$$

This shows that the smaller α the smaller the induced error during an iteration step. This confirms the fact that smaller α results in better images.

Effect of β Parameter

By a simple analysis of the main IRDA Equation (7), it can be seen that $\beta = 0$ converts the proposed method into a spatially invariant clipping. Increasing $\beta \in [0,1)$, increases the diffusion effect of IRDA and diminishes the clipping component of the equation. This results in a more prominent Retinex-like behaviour. Figure 3 shows the result of G-IRDA while increasing β.

As it can be seen, increasing β helps in improving visibility in dark areas at the cost of less saturated colours and over-compression of the image gamut. It is also evident that for $\beta = 0.99$ the algorithm tends to create "local whites" almost in every image region.

G-IRDA Behaviour Using Alternative $d(x,y)$ Functions

The shape of function $d(x,y)$ can heavily influence the result since the diffusion depends directly on it. An alternative function can be a modified Gaussian:

$$d_G(x,y) = \begin{cases} 0 & \text{if} \quad (x,y) = (0,0) \\ k_G^{-1} e^{-\left(\frac{x^2+y^2}{2\sigma_G^2}\right)} & \text{if} \quad (x,y) \neq (0,0) \end{cases} \tag{22}$$

where where k_G is such that $d_G(x,y)$ has unitary integral. After some preliminary tests we noticed that d_G induces the haloing effect in the final result. Moreover, the use of d_G requires the tuning of σ_G and thus increases the number of parameters without adding improvements. In our experimentation we noticed that the "edge" of the halo varies for different settings of σ_G. This observation prompted us the following question.

The Gaussian function in Equation (22) has inflection points when the radius $r = \sqrt{x^2+y^2}$ is equal to σ. The change of sign in the second order derivative at the inflection point makes the Gaussian function decreasing quicker when $r > \sigma$ than it does when $r \in [0,\sigma]$. This means that the diffusion action to pixels

Speed-up ratio, average time of the multi-resolution algorithm, and ΔE^*_{ab} error as a function of scales S. Uncertainty represents standard deviations of values on the set of used images.

S	5	6	7	8
Speed-up	6.2±1.4	10.4±6.2	12.1±7.74	14.1±3.4
Time (sec)	382±76	267±104	235±97	170±43
ΔE^*_{ab}	1.66±0.49	1.72±0.56	1.9±0.69	1.7±0.69

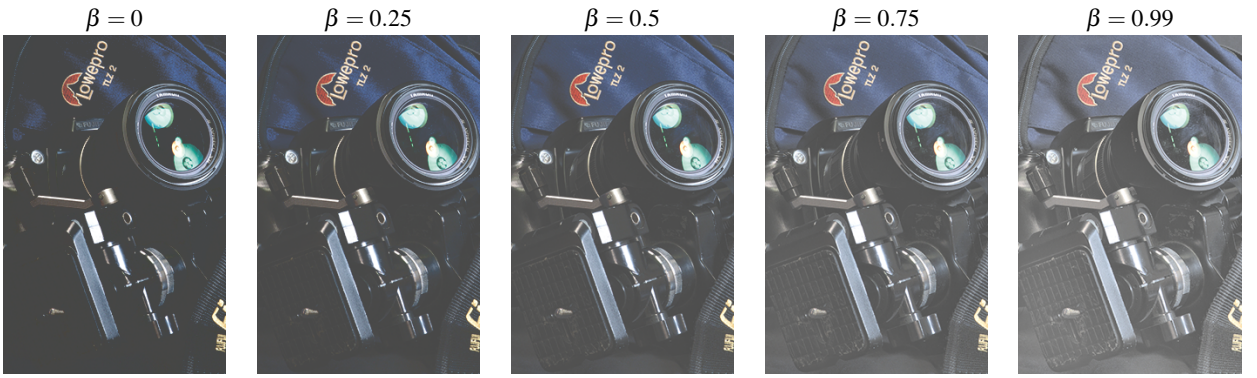

Figure 3. *G-IRDA behaviour varying β.*

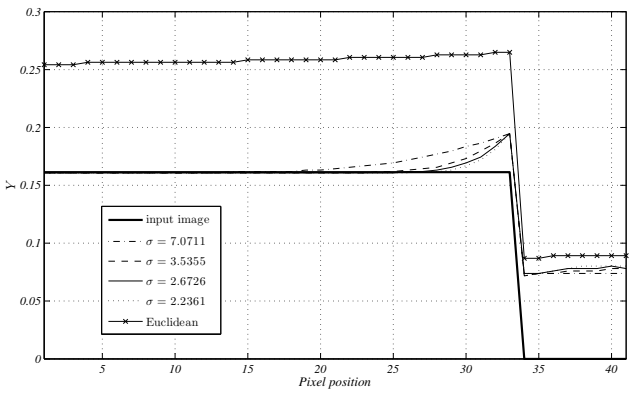

Figure 4. *Scanlines of the result of IRDA using Euclidean and Gaussian diffusion functions on a toy problem.*

that are located outside the circle of radius $r = \sigma$ is conspicuously minor than the action applied to pixels inside this circle. Is the *haloing* the visible "signature" of the presence of an inflection point in the diffusion function?

To support this hypothesis we performed a preliminary test with a toy image composed by a strong edge in which the lower value is $Y_l = 0$, and thus out of gamut, while the upper value is $Y_h = 0.1614$ (in-gamut). Figure 4 shows the scan-lines of the input toy image and the results with different σ values plus the result using the inverse Euclidean function of Equation (6). Using this toy image and varying σ, we obtained that the Pearson's linear correlation coefficient between the inflection point of the Gaussian function (σ) and the point in which the derivative of scanlines reach the zero derivative, e.g. when the halo disappears, is 0.9989.

This discussion is not meant to be exhaustive, rather it wants to raise a question that requires more investigation: is the Gaussian function the only (or major) cause of haloing artefact during isotropic diffusion or multi-level/scale computation?

Results and Comparison

Figure 5 shows the result of gamut mapping the five images shown in the leftmost column to the ISO uncoated gamut using three existing spatially variant gamut mapping algorithms, and two versions of G-IRDA proposed here. The images in the second column are gamut mapped by the Retinex-based algorithm by McCann [13]. The third column shows the results of the multi-scale approach by Farup et al. [7], which encompasses the Morovic [24] approach and moreover solves some of the drawbacks and also reduces the haloing effect. The parameter settings proposed in the paper are used as is. The fourth column shows the results of the approach presented by Zolliker et al. [25] in which the method by Bala et al. [26] has been modified with a bilateral filter to reduce the haloing effect. The two rightmost columns show the results of the G-IRDA algorithm with $\beta = 0.5$ ($\alpha = 1/512$) with both the $Y_\perp$ and $G_\perp$ saturation clipping strategies, respectively.

The topmost image is a toy image that was created by the authors in order to exaggerate some of the problems commonly encountered in spatially variant gamut mapping algorithms – hue shifts and halos [7]. These types of artefacts are very clearly seen for the McCann approach. For the Farup algorithm the hue shift are removed, but the halos are still visible. In the Zolliker reproduction and the two G-IRDA ones, these artefacts are more or less completely eliminated. There are other obvious differences between the results of these three algorithms also, in particular when it comes to relative and absolute saturation and lightness. This can be seen by comparing the red, green, blue and yellow central patches, and their relative contrast with the central gray patches.

For the camera image, all algorithms but McCann perform fairly well. The other ones mainly differ in how well they repro-

duce the details of the tripod in the lower left part, and also in how they reproduce the red and blue colours of the camera bag. For the G-IRDA approaches, the behaviour with respect to details can be tuned by changing the β parameter, as already discussed.

Looking at the picnic image, the importance of the choice of colour space becomes particularly apparent. Where the colour of the sky gets a hint of purple by the Farup algorithm and slightly so also by the Zolliker algorithm that both use the CIELAB space, it is much better rendered by G-IRDA. The direction of the mapping clearly affects the rendering of the sky for the two versions of the G-IRDA approach, where the luminance contrast of the clouds becomes much more apparent in the $Y_\perp$ vs. the $G_\perp$ strategy. Finally, the details of the grass are clearly best rendered by G-IRDA.

The final image is different in that it contains large homogeneous areas with low gradients and almost no texture. Such images often do quite well with spatially invariant gamut mapping, whereas spatially variant algorithms tend to introduce evident artefacts and haloing. Artefacts are clearly seen for the McCann algorithms, and halos are also visible in the Farup approach. For G-IRDA and Zolliker, it is not possible to identify any such artefacts for this image. When it comes to overall lightness contrast and rendering, the two G-IRDA versions are clearly the best performing ones.

Conclusion

In this paper we introduced a simple yet effective spatially variant gamut mapping algorithm that does not generate visible artefacts and/or haloing. We argue that this behaviour is mainly due to the use of a diffusion function that has no inflection point. Additionally, we show that RGB spaces could be appropriate for gamut mapping due to good hue constant half-planes. The proposed algorithm has only one free parameter (β), which regulates the behaviour of the algorithm from pure clipping to a Retinex-like enhancement. This is a clear advantage with respect to state-of-the-art methods, which usually requires more parameters, frequently not directly related to gamut mapping aspects. We shown that the proposed algorithm can be incorporated into a multi-resolution scheme to speed it up consistently, making it suitable for very large images.

Appendix: Convergence of IRDA

To demonstrate that the iterative algorithm of Equation (7) converges we have to demonstrate that:

$$\lim_{p\to\infty} \alpha V \left((1-\beta) Q_p + \beta \frac{M_p}{\alpha} \right) = 0 \tag{23}$$

i.e. for all the pixels $(x,y) \in \mathfrak{I}$, the magnitude of the change tends to zero while the number of iteration tends to infinity. The first thing we have to guarantee is that the matrix V is designed correctly to force the pixels colour to progressively move into the destination gamut; in this way the Q operator must assume the zero value for all the pixels after a certain iteration P_c that depends on the destination gamut, input image, and on V. Thus, if V is properly defined we can rewrite the limit as:

$$\lim_{p\to\infty} \alpha V \left(\beta \frac{M_p}{\alpha} \right) = 0 \tag{24}$$

i.e. for all the positions (x,y), M_p has to reach the zero value. Now it's easy to substitute M_p using its definition (see Equation (5)), simplify α, obtaining the following equation:

$$\lim_{p\to\infty} V(\beta \| I_p - I_{p-1} \| * d) = 0 \tag{25}$$

Now, since the convolving function d has an integral normalisation, this quantity is forced to decrease if $\beta V < 1$. Since $\beta \in [0,1)$ the constraint we should respect is that $\|V\| \leq 1$ for all the coordinates (x,y). This is sufficient to have an algorithm that converge to a solution.

References

[1] Ján Morovič and M. Ronnier Luo. The fundamentals of gamut mapping: A survey. *Journal of Imaging Science and Technology*, 45(3):283–290, 2001.

[2] Phil Green. Baseline gamut mapping method for the perceptual reference medium gamut. In *IS&T/SPIE Electronic Imaging*, pages 93950M–93950M. International Society for Optics and Photonics, 2015.

[3] Jens Preiss, Mark D Fairchild, James A Ferwerda, and Philipp Urban. Gamut mapping in a high-dynamic-range color space. In *IS&T/SPIE Electronic Imaging*, pages 90150A–90150A. International Society for Optics and Photonics, 2014.

[4] Philipp Urban and Roy S Berns. Paramer mismatch-based spectral gamut mapping. *Image Processing, IEEE Transactions on*, 20(6):1599–1610, 2011.

[5] Thiago Pereira and Szymon Rusinkiewicz. Gamut mapping spatially varying reflectance with an improved brdf similarity metric. *Computer Graphics Forum*, 31(4):1557–1566, 2012.

[6] Ali Alsam and Ivar Farup. Spatial colour gamut mapping by orthogonal projection of gradients onto constant hue lines. In *8th International Symposium on Visual Computing*, pages 556–565, Rethymnon, Crete, Greece, July 2012.

[7] Ivar Farup, Carlo Gatta, and Alessandro Rizzi. A multiscale framework for spatial gamut mapping. *IEEE T. Image Process.*, 16(10), 2007. doi: 10.1109/TIP.2007.904946.

[8] Syed Waqas Zamir, Javier Vazquez-Corral, and Marcelo Bertalmio. Gamut mapping in cinematography through perceptually-based contrast modification. *IEEE Journal of Selected Topics in Signal Processing*, 8(3):490–503, 2014.

[9] Peter Zolliker, Zofia Baranczuk, Iris Sprow, and Joachim Giesen. Conjoint analysis for evaluating parameterized gamut mapping algorithms. *Image Processing, IEEE Transactions on*, 19(3):758–769, 2010.

[10] Fabienne Dugay, Ivar Farup, and Jon Y. Hardeberg. Perceptual evaluation of color gamut mapping algorithms. *Color Res. Appl.*, 33(6):470–476, 2008.

[11] Edoardo Provenzi, Massimo Fierro, Alessandro Rizzi, Luca De Carli, Davide Gadia, and Daniele Marini. Random spray retinex: A new retinex implementation to investigate the local properties of the model. *IEEE Transactions on Image Processing*, 16:162–171, January 2007.

[12] Øyvind Kolås, Ivar Farup, and Alessandro Rizzi. STRESS: A framework for spatial color algorithms. *J. Imaging. Sci. Technology*, 55(4):040503, 2011.

[13] John J. McCann and P. M. Hubel. In-gamut image reproduction using spatial comparisons. United States Patent, US 6,516,089 B1, 2003.

[14] Ivar Farup. Hyperbolic geometry for colour metrics. *Optics Express*, 22(10):12369–12378, 2014.

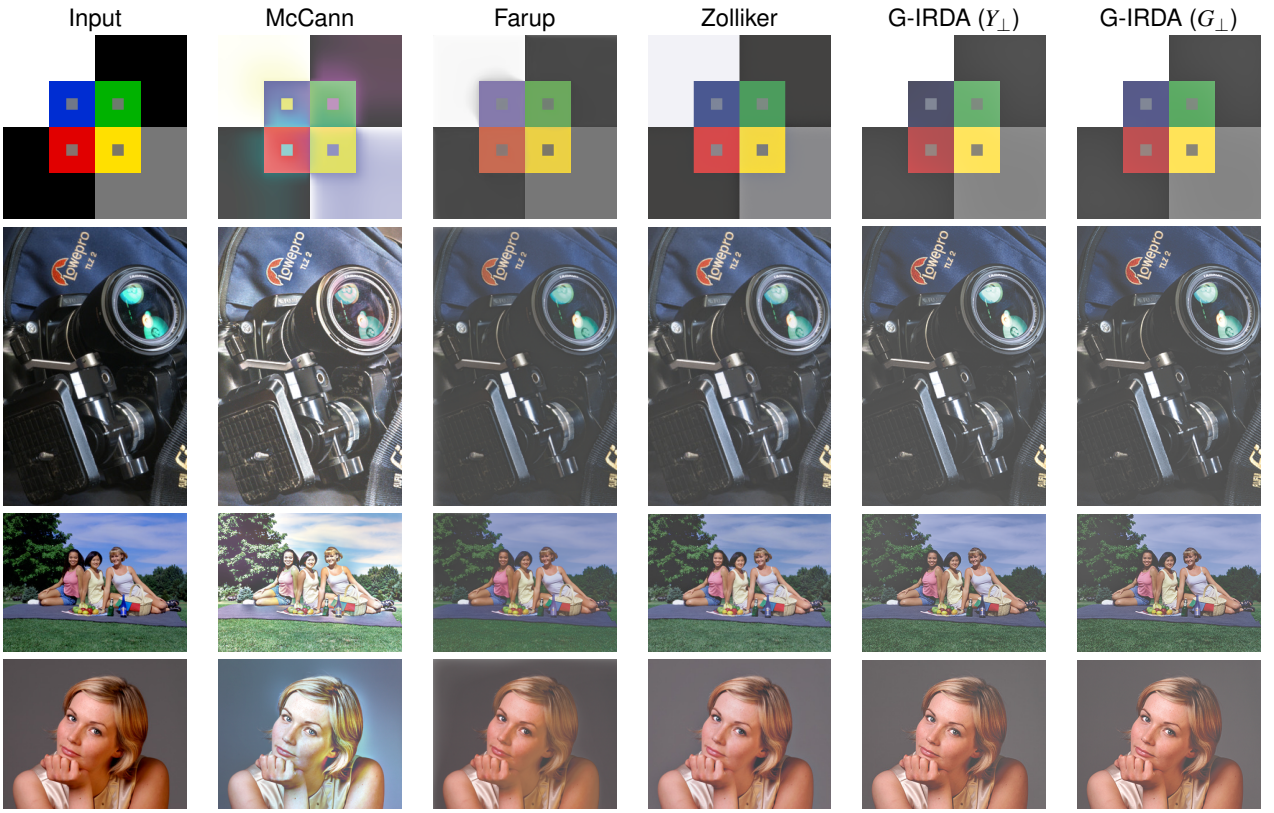

Figure 5. *Comparison of several state-of-the-art algorithm and the GIRDA algorithm with both $Y_\perp$ and $G_\perp$ clipping strategies.*

[15] Edwin H. Land and John J. McCann. Lightness and retinex theory. *Journal of the Optical Society of America*, 61(1):1–11, January 1971.

[16] Nathan Moroney, Mark D. Fairchild, Robert W. G. Hunt, Changjun Li, M. Ronnier Luo, and Todd Newman. The CIECAM02 color appearance model. In *Proceedings of IS&T and SID's 10th Color Imaging Conference: Color Science and Engineering: Systems, Technologies, Applications*, pages 23–27, Scottsdale, Arizona, 2002.

[17] Ingeborg Tastl, Miheer Bhachech, Nathan Moroney, and Jack Holm. Icc color management and ciecam02. In *Color and Imaging Conference*, volume 2005, pages 217–223. Society for Imaging Science and Technology, 2005.

[18] Gabriel G. Marcu. Munsell constant hue sections. In *Proc. IS&T/SID's 6th Color Imaging Conference*, pages 159–162, 1998.

[19] John J. McCann. Uniform color spaces: 3d luts vs. algorithms. In *Proceedings of IS&T's 1999 PICS Conference*, pages 204–208, 1999.

[20] Edwin H. Land. The retinex theory of color vision. *Scientific American*, 237:108–128, 1977.

[21] Edoardo Provenzi, Luca De Carli, and Alessandro Rizzi. Mathematical definition and analysis of the retinex algorithm. *J. Opt. Soc. Am. A*, 22(12):2613–2621, December 2005.

[22] Daniel J. Jobson, Zia-ur Rahman, and Glenn A. Woodell. A multiscale retinex for bridging the gap between color images and the human observation of scenes. *IEEE Transactions on image processing*, 6(7):965–976, July 1997.

[23] Edwin H. Land. An alternative technique for the computation of the designator in the retinex theory of color vision. *PNAS*, 83(10):3078–3080, May 1986.

[24] Ján Morovič and Yu Wang. A multi-resolution, full-colour spatial gamut mapping algorithm. In *Proceedings of IS&T and SID's 11th Color Imaging Conference: Color Science and Engineering: Systems, Technologies, Applications*, pages 282–287, Scottsdale, Arizona, 2003.

[25] Peter Zolliker and Klaus Simon. Retaining local image information in gamut mapping algorithms. *IEEE Trans. Image Proc.*, 16(3):664–672, March 2007.

[26] Raja Bala, Ricardo deQueiroz, Reiner Eschbach, and Wencheng Wu. Gamut mapping to preserve spatial luminance variations. *Journal of Imaging Science and Technology*, 45(5):436–443, September/October 2001.

Author Biography

Carlo Gatta obtained the degree in Electronic Engineering in 2001 from the Università degli Studi di Brescia (Italy), and in 2006 the Ph.D. in Computer Science at the Università degli Studi di Milano (Italy). From 2007 to 2016 he was at the Computer Vision Center at Universitat Autonoma de Barcelona mainly working on medical imaging. Currently, he is an independent scientist consultant, mainly focusing on image processing, computer vision, machine learning, and deep learning.

Ivar Farup received a MSc in physics from the Norwegian Institute of Technology in 1994 and a PhD in applied mathematics from the University of Oslo, Norway in 2000. He is currently a professor of computer science at NTNU – Norwegian University of Science and Technology, mainly focusing on colour science and image processing.