Published in Image Processing On Line on 2011–09–27.
Submitted on 2011–00–00, accepted on 2011–00–00.
ISSN 2105–1232 © 2011 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials, software, datasets and online demo at
https://doi.org/10.5201/ipol.2011.gl_lcc

Local Color Correction

Juan Gabriel Gomila Salas, Jose Luis Lisani

Universitat de les Illes Balears (UIB), Spain
(juan.gabriel@me.com, joseluis.lisani@uib.es)

Communicated by Jean-Michel Morel *Demo edited by* Jose-Luis Lisani Roca

Abstract

In this paper we present a local algorithm for contrast enhancement developed by N. Moroney at Hewlett-Packard Laboratories and presented at the IS&T/SID Eight Color Imaging Conference, in 2000. The algorithm uses a non-linear masking, is fast and does not require any manual parameter adjustments.

Source Code

The source code (ANSI C), its documentation, and the online demo are accessible at the IPOL web page of this article[1].

Keywords: contrast enhancement; local method; gamma correction

1 Introduction

In the context of this paper, by color correction techniques we refer to methods that increase the contrast of digital images. When images are either too dark or too bright a classical gamma correction is enough to increase their dynamic range and improve their contrast.

Figures 1 and 2 display two examples of contrast enhancement using gamma correction. In the first case a dark image is processed with $\gamma = 0.5$, while in Figure 2 $\gamma = 2.5$ is used to process a bright image. The histograms of the resulting images show a clear increase of the dynamic range. In these examples is also shown that global histogram equalization doesn't perform well, since it increases excessively the dynamic range of the original images.

However, when images contain both dark and bright regions gamma correction techniques perform poorly (see Figure 3). The reason is that gamma correction is a global technique. All pixels having a particular input intensity level are assigned the same output intensity, independent of the local context.

Results in Figure 3 show that it is not possible to simultaneously improve the contrast of dark and bright regions using gamma correction. A compromise solution for choosing γ is to compute

[1]https://doi.org/10.5201/ipol.2011.gl_lcc

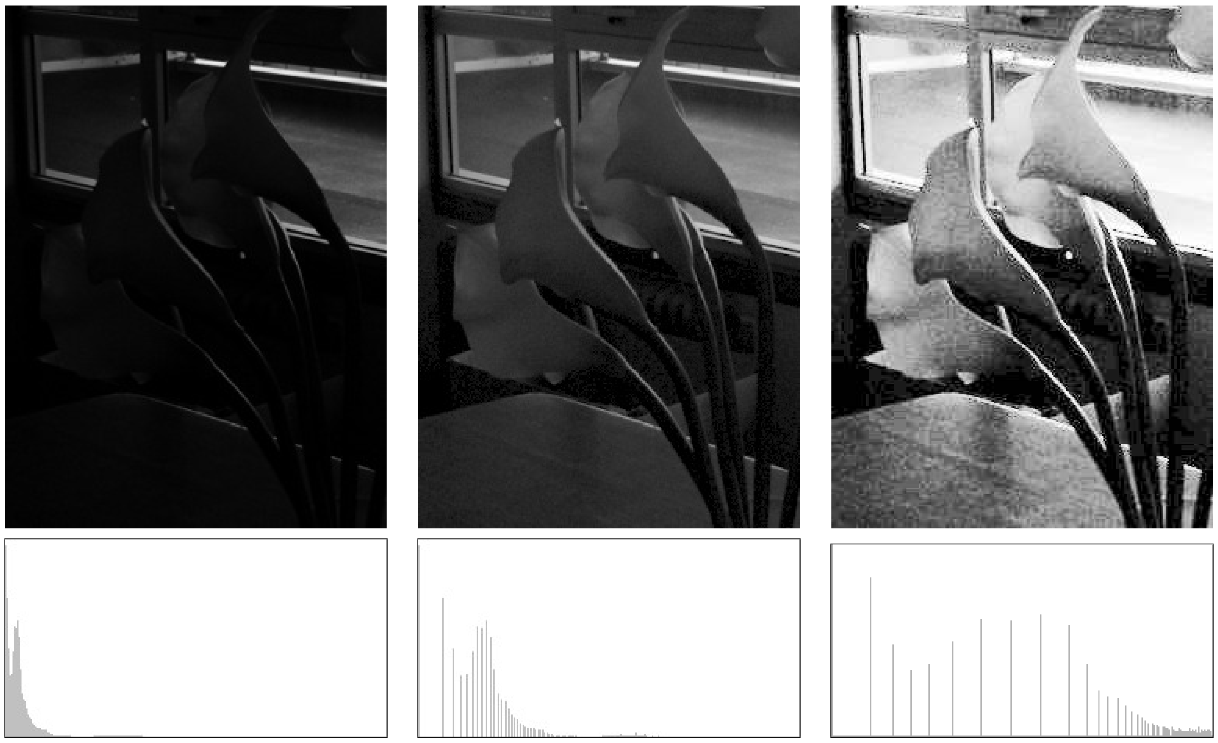


Figure 1: Effect of gamma correction with $\gamma = 0.5$ on a dark image. Top row, from left to right: original image, gamma correction result, histogram equalization result. The respective intensity histograms are shown below.

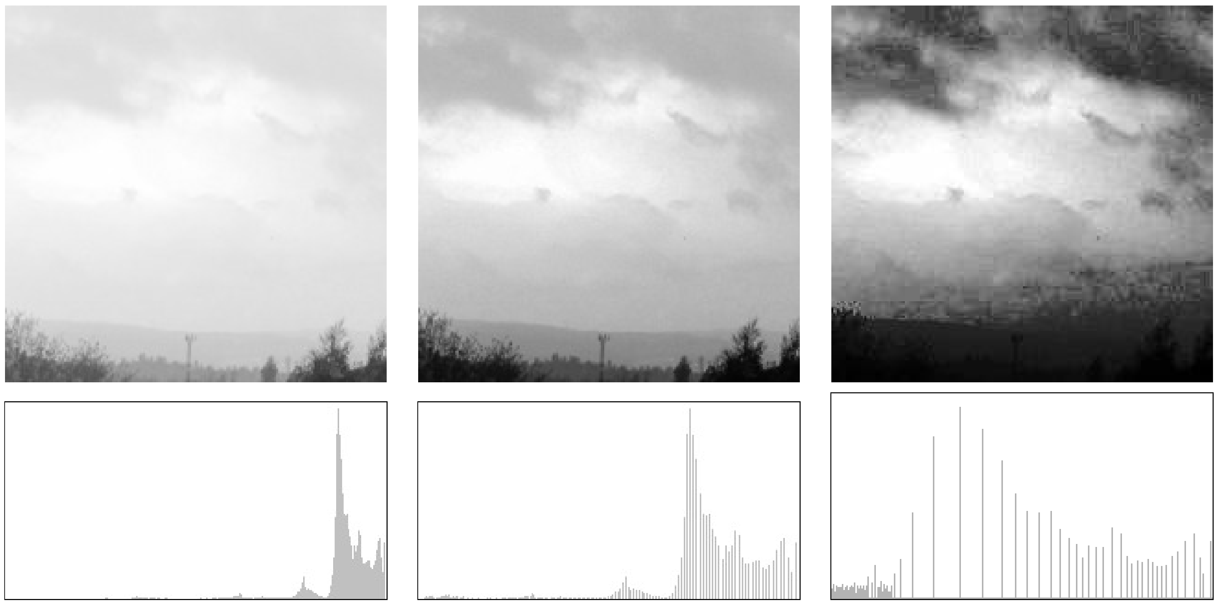


Figure 2: Effect of gamma correction with $\gamma = 2.5$ on a bright image. Top row, from left to right: original image, gamma correction result, histogram equalization result. The respective intensity histograms are shown below.

the mean grey level μ of the image and then use $\gamma > 1$ (implying attenuation of values) or $\gamma < 1$ (implying amplification of values) depending on whether μ is above 127.5 or below 127.5, respectively (we assume 8-bits images), according to the following formula (*default* γ), inspired by [1]

$$\gamma = 2^{(\mu-127.5)/127.5}. \tag{1}$$

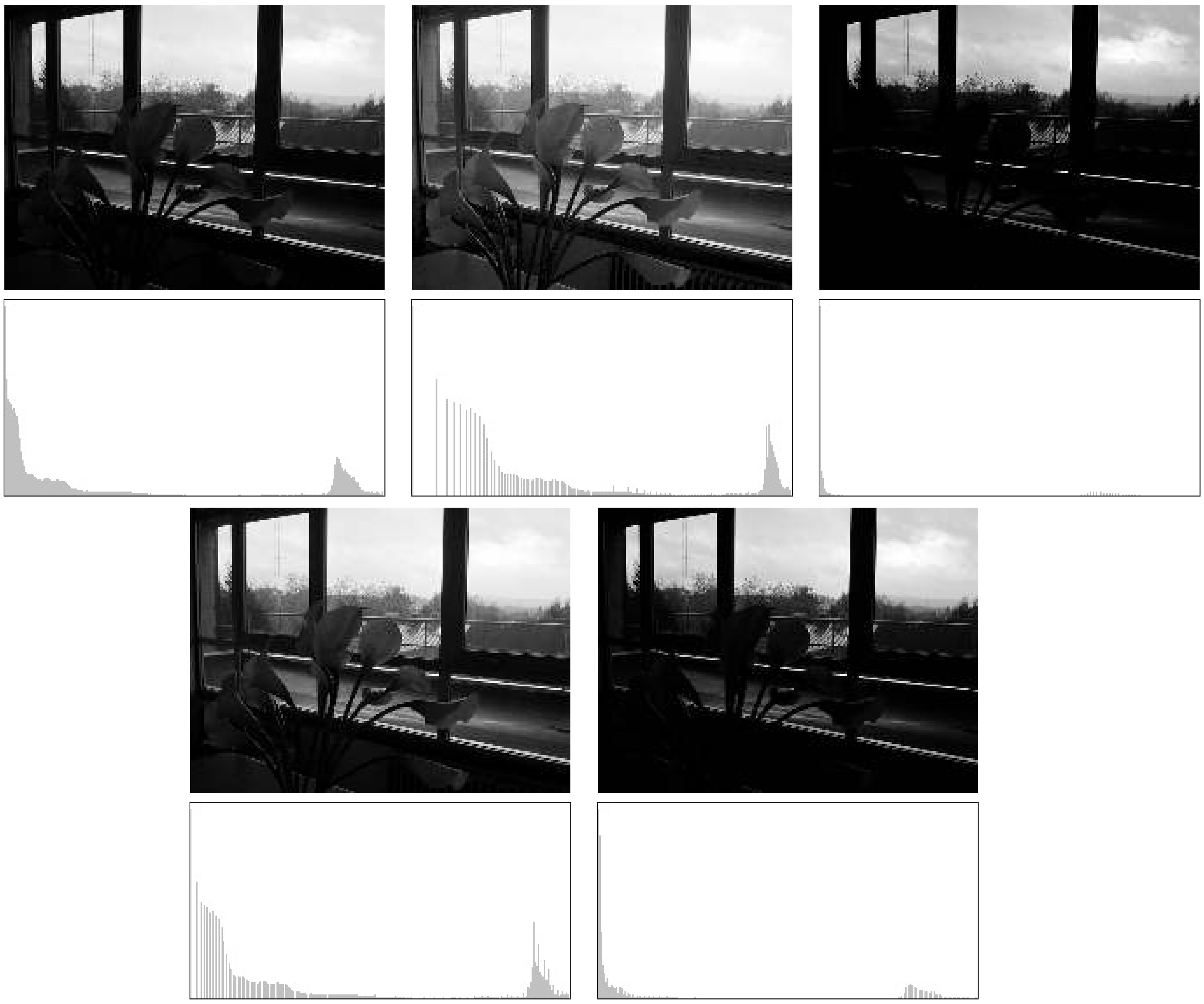


Figure 3: Gamma correction of an image with dark and bright regions with different values of γ. First row, from left to right: original image and gamma correction results with $\gamma = 0.5$ and $\gamma = 2.5$. Third row, gamma correction results with $\gamma = 0.75$ and $\gamma = 1.5$. Below each image its intensity histogram is displayed.

For the original image in Figure 3, the default γ is 0.74, which gives almost the same result shown in Figure 3 for $\gamma = 0.75$.

In situations when shadows and highlights are present in the image, local techniques outperform global techniques. Local techniques can map one input value to many different possible output values, depending on the values of the neighboring pixels. This allows simultaneous shadow and highlight adjustment.

In this paper we present a local algorithm for contrast enhancement developed by N. Moroney at Hewlett-Packard Laboratories and presented at the IS&T/SID Eight Color Imaging Conference, in 2000 (US Patent 6,822,762, 2004) [1, 2]. The algorithm uses a non-linear masking, is fast and does not require any manual parameter adjustments.

2 Algorithm (LCC Algorithm)

Assume 8-bits RGB color images (R, G, B values in the range [0, 255]). The algorithm is computed in two steps:

1. A mask image is computed from the input image.
2. The input and mask images are combined to get the result.

The mask image is computed from the intensity component of the color image, defined as the average of R, G and B values i.e. $I = (R + G + B)/3$. The use of intensity information avoids distortions of the chroma. The mask image is obtained by inverting and then blurring the intensity component of the input image

$$M(x, y) = (\text{Gaussian} * (255 - I))(x, y). \tag{2}$$

Blurring is performed by using a Gaussian kernel of large radius, which guarantees that image contrast will not be excessively reduced along the edges (see discussion below). The resulting mask indicates which regions of the image will be lightened or darkened. For instance, a light region of the image will have a dark mask value, so it will be darkened.

The combination operation consists of a power function, where the exponent is computed using the mask value previously found. If the mask value is greater than 128, it will result in an exponent less than 1, while if the mask value is lower than 128, it will result in an exponent greater than 1. Moreover, if the mask value is precisely 128, the exponent will be 1, and it will have no effect on the input image. The operation is equivalent to a pixel-wise gamma correction and can be written as the following equation

$$\text{Output}(x, y) = 255 \left(\frac{\text{Input}(x, y)}{255}\right)^{2^{\frac{128-M(x,y)}{128}}}, \tag{3}$$

where, if (x, y) is a pixel coordinate of the image domain, $\text{Input}(x, y)$ is the input image, $\text{M}(x, y)$ is the computed mask and $\text{Output}(x, y)$ is the output image.

If R, G, B are normalized in the range [0, 1], then the formulas can be simplified

$$I(x, y) = \frac{R(x, y) + G(x, y) + B(x, y)}{3},$$

$$M'(x, y) = (\text{Gaussian} * I)(x, y), \tag{4}$$

$$\text{Output}(x, y) = (\text{Input}(x, y))^{2^{2M'(x,y)-1}}. \tag{5}$$

In the case of monochrome images Input(x, y) is the intensity component of the image. In [1] only results for monochrome images are shown.

For color images we have mainly two options:

- Apply the algorithm channel by channel:
    1. compute I and M' as in the formulas above for (Input, Output) in (R, new R), (G, new G), (B, new B),
    2. apply Equation (5).

In Section 4 (figures 6 and 8) it is shown that this option may lead to changes in chrominance.

- Take a Luma+Chroma approach:
  1. convert the input RGB image to a Luma+Chroma color representation,
  2. apply LCC to the Luma component:
  $$M'(x,y) = (\text{Gaussian} * \text{Luma})(x,y),$$
  $$\text{new Luma}(x,y) = (\text{Luma}(x,y))^{2^{(2M'(x,y)-1)}},$$
  (we assume Luma values in [0, 1]),
  3. convert back to RGB using the new Luma and the original Chroma

In the second case, we have several possibilities, depending on the model for color representation that we choose. Since a discussion on which color model, if any, is better is beyond the goal of this paper we just decided to display the results of using three different models:

- HSI[2]. That is, Luma=I=(R+G+B)/3, Chroma=HS. In this case, preservation of chroma implies preservation of original R/G/B ratios.
- HSL[3]. That is, Luma=L, Chroma=HS.
- YPbPr[4]. That is, Luma=Y, Chroma=PbPr.

Refer to Pascal Getreuer's *colorspace* web site[5] for further information about color models and conversion formulas.

Comparisons of the various implementations of the algorithm are presented in Section 4.

Parameters of the algorithm. The only free parameter of the algorithm is the radius (r) of the blurring filter used to obtain the mask image. As commented above, a certain amount of blurring is needed in order to avoid low contrasted edges. In particular, the author in [1] recommends to use a large radius, in such a way that image features can no longer be recognized. However, if the radius is too big the mask image will become uniform and the algorithm will reduce to a classical gamma correction. In Section 4 we investigate the effect of the radius magnitude on various test images.

3 Implementation

Four versions of the local color correction (LCC) algorithm have been implemented (see previous section for details):

- LCC-RGB: LCC algorithm applied channel by channel on the RGB input image.
- LCC-HSI: Luma+Chroma approach using the HSI color model.
- LCC-HSL: Luma+Chroma approach using the HSL color model.
- LCC-YPbPr: Luma+Chroma approach using the YPbPr color model.

[2]http://en.wikipedia.org/wiki/HSI_color_space
[3]http://en.wikipedia.org/wiki/HSL_and_HSV
[4]http://en.wikipedia.org/wiki/YPbPr
[5]http://www.getreuer.info/home/colorspace

In all the implementations, we have programmed the masking step by assuming that the original image has been extended by even symmetry.

Concerning the range of values for parameter r, we have decided to allow values between 0 (no blurring) to half the minimum dimension of the image. The use of larger radius implies almost uniform mask images. In such cases, we have decided to compute a global gamma correction with default γ

$$\gamma = 2^{(\mu-127.5)/127.5}, \tag{6}$$

where μ is defined as follows:

- average value of I=(R+G+B)/3, in LCC-RGB and LCC-HSI
- average value of L, in LCC-HSL
- average value of Y, in LCC-YPbPr

4 Results

First, we start (Figure 4) by testing the algorithm on the image displayed in Figure 3, in order to check whether the algorithm is able to improve simultaneously the contrast of dark and bright regions.

The algorithm is run with different values of parameter r and the results are compared with a gamma correction with default γ (as defined by Equation (6)).

Results in Figure 4 show that LCC outperforms classical gamma correction when shadows and highlights are simultaneously present at the scene. Moreover, the effects of variations of parameter r are appreciated when comparing different results: as r increases objects become sharper, their contrast with respect to surrounding objects increasing. Those effects are specially visible in the leafs of the trees (see detail in Figure 5).

In figures 6 to 8 color information is added to the images and the results of algorithms LCC-RGB, LCC-YPbPr, LCC-HSI and LCC-HSL are compared. We can also compare the results with the ones in Figure 4, since a color version of the same original image is used in this test. Figures 6 and 7 show the results of the different versions of the algorithm for a fixed value of parameter r ($r = 40$). The corresponding R, G, B and I histograms are also displayed. A detail of the image is displayed in Figure 8, which permits to appreciate the differences between the results of the algorithms: LCC-HSI and LCC-HSL preserve the original chrominances (observe the yellowish colors of the flowers), while LCC-RGB and LCC-YPbPr alter this information (flowers are nearly white).

We conclude that, as expected, LCC-RGB does not preserve the chrominances of the original images. The same is true for LCC-YPbPr, even if it is based on a Luma+Chroma model. The versions of the algorithm based HSL and HSI do preserve these chrominances.

It must be remarked however that the HSI and HSL models perform poorly in dark regions, since the chrominance information in these regions is highly perturbed by noise. This can be appreciated in figures 9 and 10 where a false greenish color appears in the processed image.

Figures 11 and 12 show another example of the poor performance of LCC-HSI and LCC-HSL in dark regions. In this case the dark portion of the car is converted to very saturated red, which looks unnatural.

The problem with HSI and HSL is that chrominance information is quite unreliable for almost-black colors (small perturbations of R, G and B produce very different values of H and S).

In particular, the problem with HSI is related to the R/G/B ratios having a singularity at black. Let I and I' denote the original intensity and the LCC-corrected intensity, then the output colors are

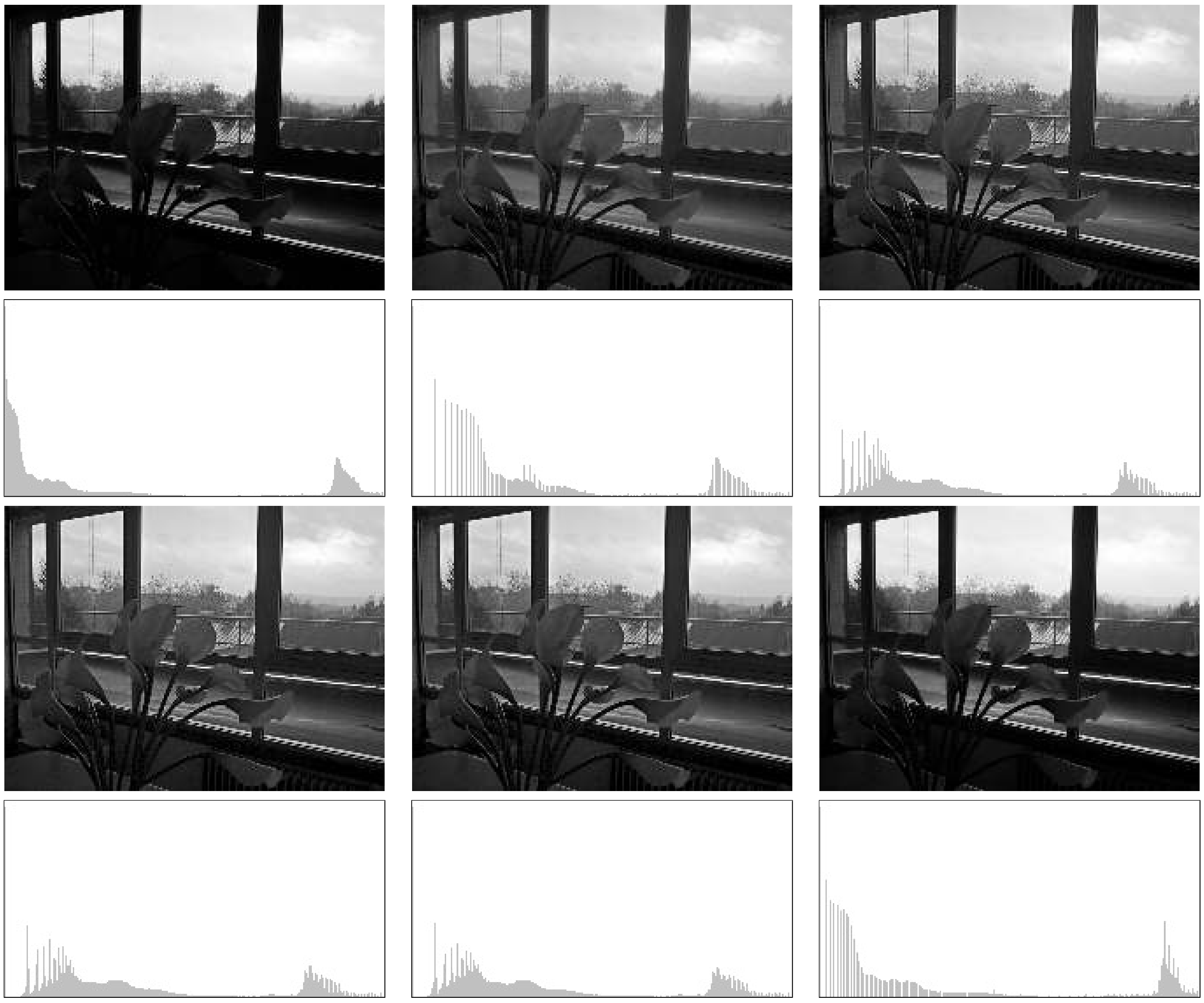


Figure 4: Local Color Correction of a monochrome image, with different values of the parameter r. The result of global correction (gamma correction with default γ computed with Equation (6)) is also displayed. First row, from left to right: original image, results with $r = 0$ and $r = 40$. Third row, results with $r = 100$, $r = 200$ and result of global gamma correction. Below each image its intensity histogram is displayed.

$$R' = \frac{I}{I'}R, \qquad G' = \frac{I}{I'}G, \qquad B' = \frac{I}{I'}B.$$

If (R, G, B) and its neighbors are almost black, then relative to intensities in [0, 1], the LCC-RGB output is approximately

$$R' = \sqrt{R}, \qquad G' = \sqrt{G}, \qquad B' = \sqrt{B},$$

while the LCC-HSI output is approximately

$$R' = \frac{R}{\sqrt{I}}, \qquad G' = \frac{G}{\sqrt{I}}, \qquad B' = \frac{B}{\sqrt{I}}.$$

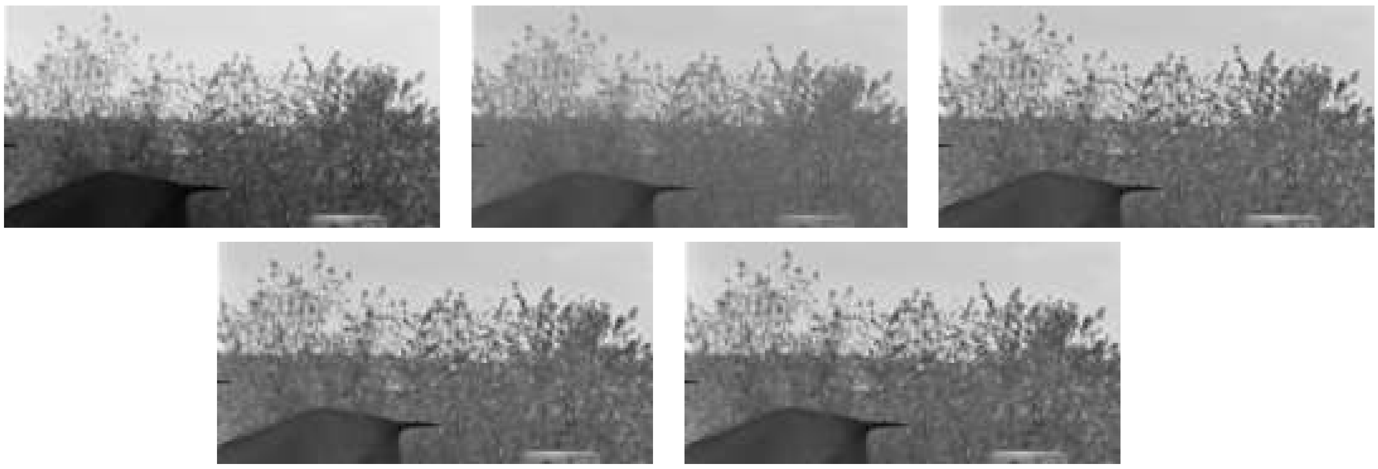


Figure 5: Detail of images in Figure 4. First row, from left to right: original image, results with $r = 0$ and $r = 40$. Second row, results with $r = 100$ and $r = 200$. Observe as contrast increases with r, although differences between $r = 40$ and $r = 100$ or $r = 200$ are small.

So the LCC-HSI output color (R', G', B') is more sensitive to small perturbations in an almost-black input color than the LCC-RGB output.

On the other hand, for an almost-black color, the output with this LCC-YPbPr procedure is approximately

$$R' = \sqrt{Y} + 1.402\mathrm{Pr}$$

$$G' = \sqrt{Y} - 0.344\mathrm{Pb} - 0.713\mathrm{Pr}$$

$$G' = \sqrt{Y} + 1.772\mathrm{Pb},$$

where (Y,Pr,Pb) are a linear transformation of the input colors (R,G,B). So LCC-YPbPr's sensitivity to perturbations is similar to LCC-RGB, and therefore it works better than LCC-HSI for dark colors.

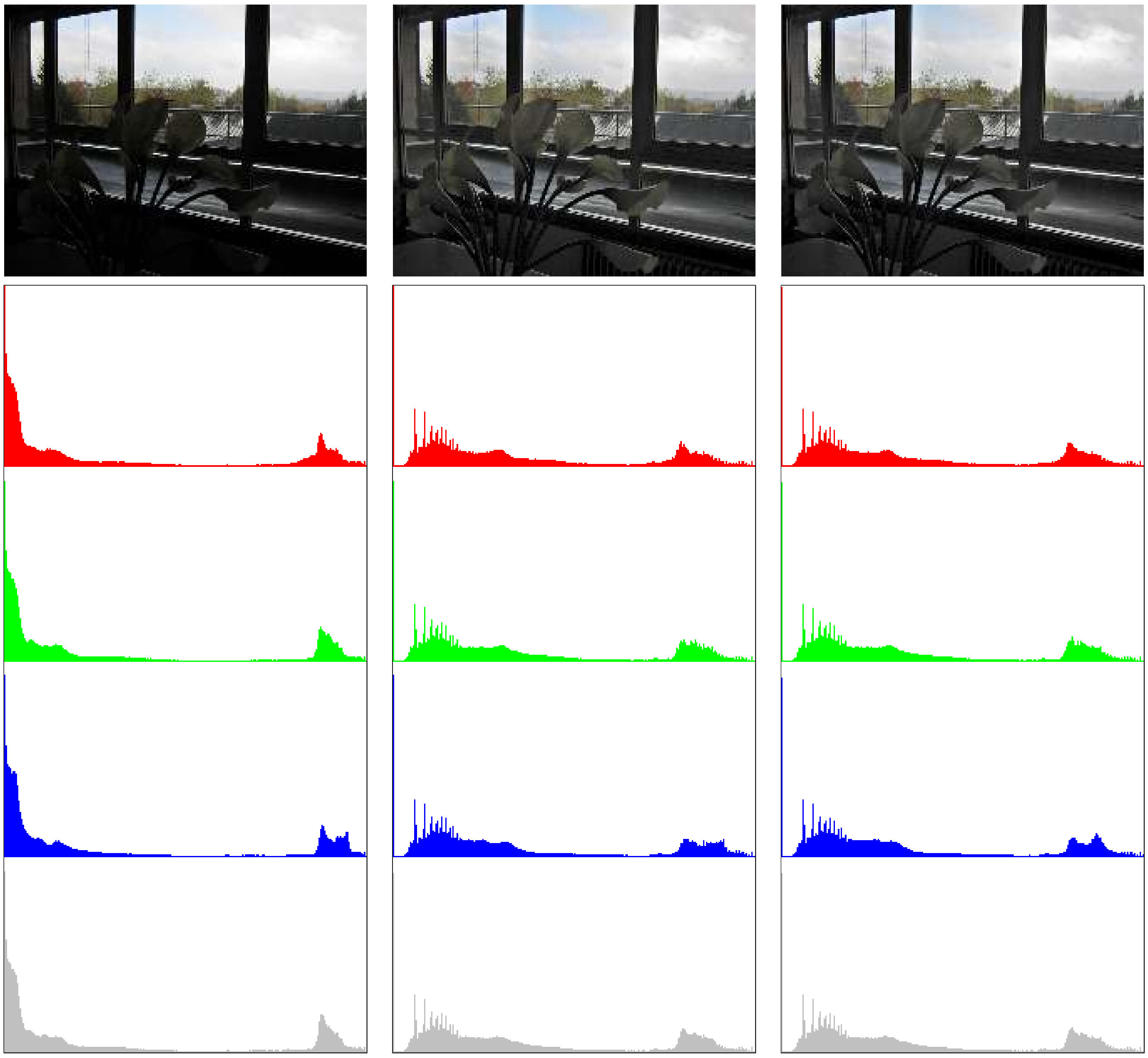


Figure 6: Local Color Correction of a color image, with a fixed value of the parameter ($r = 40$). First row, from left to right: original image, results of algorithms LCC-RGB and LCC-YPbPr. Below each image its R, G, B and intensity histograms are displayed.

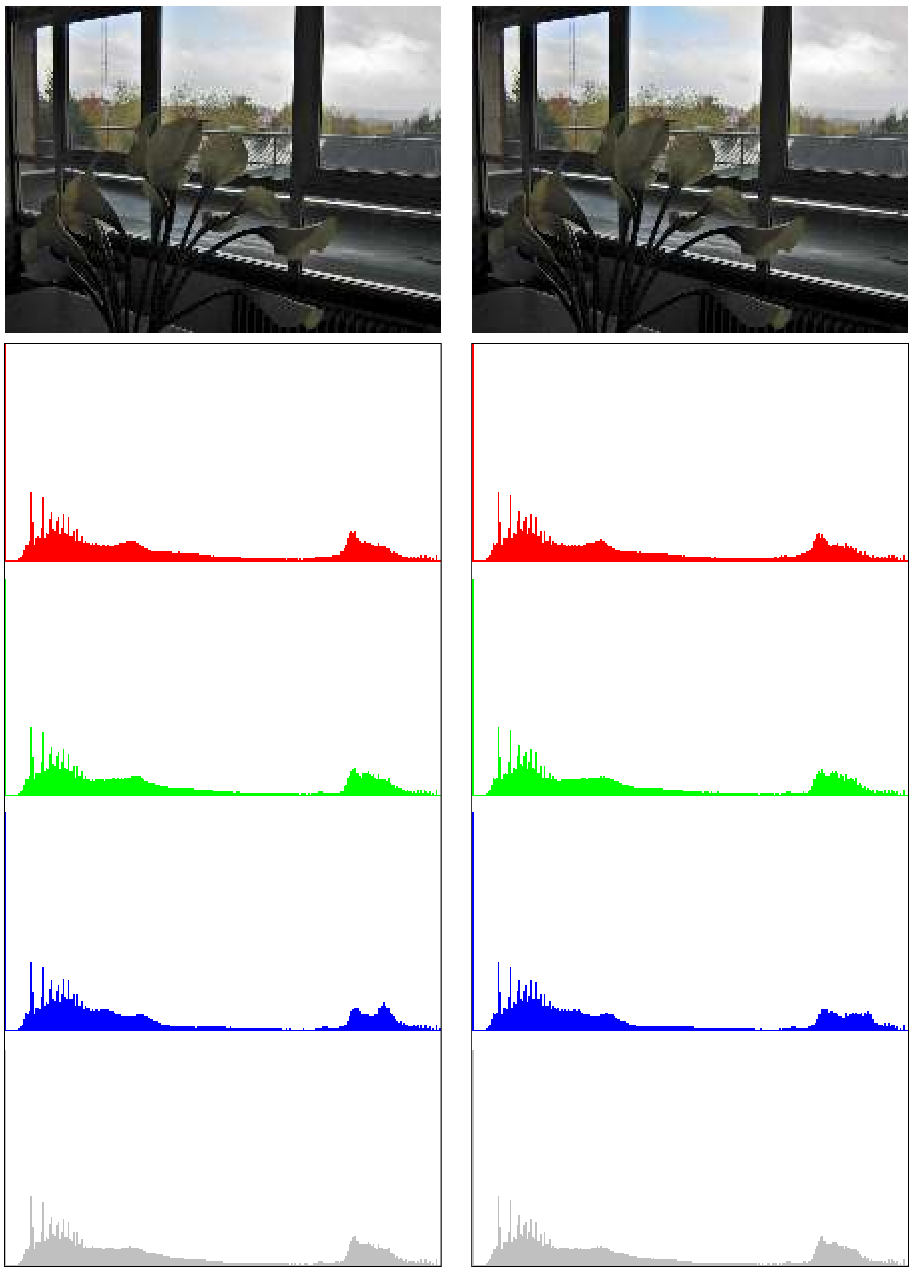


Figure 7: Local Color Correction of a color image, with a fixed value of the parameter ($r = 40$). First row, from left to right, results of algorithms LCC-HSI and LCC-HSL. Below each image its R, G, B and intensity histograms are displayed.

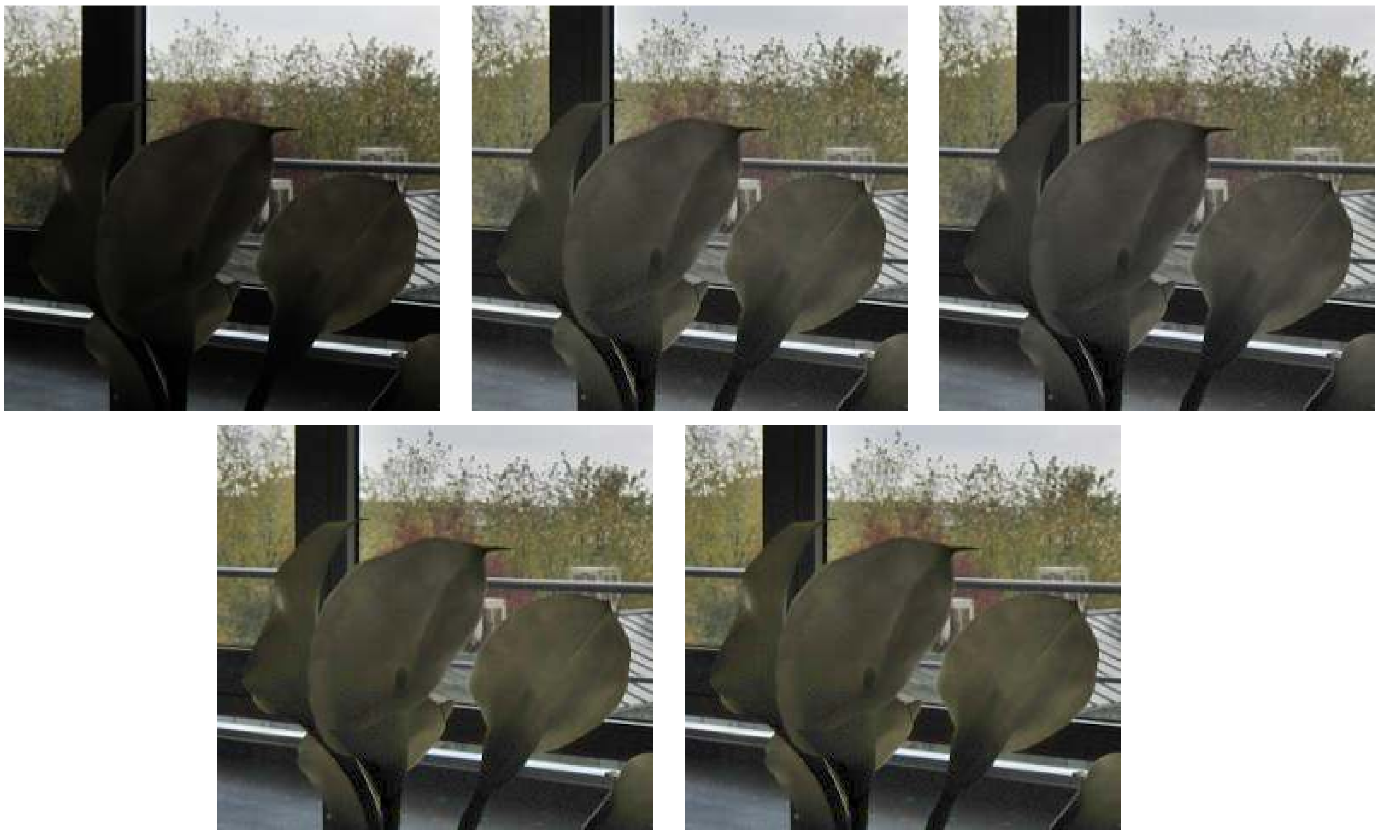


Figure 8: Detail of images in figures 6 and 7. Top, from left to right, original image and results of algorithms LCC-RGB and LCC-YPbPr. Bottom, results of algorithms LCC-HSI and LCC-HSL. Observe that LCC-HSI and LCC-HSL preserve the original yellowish colors of the flowers.

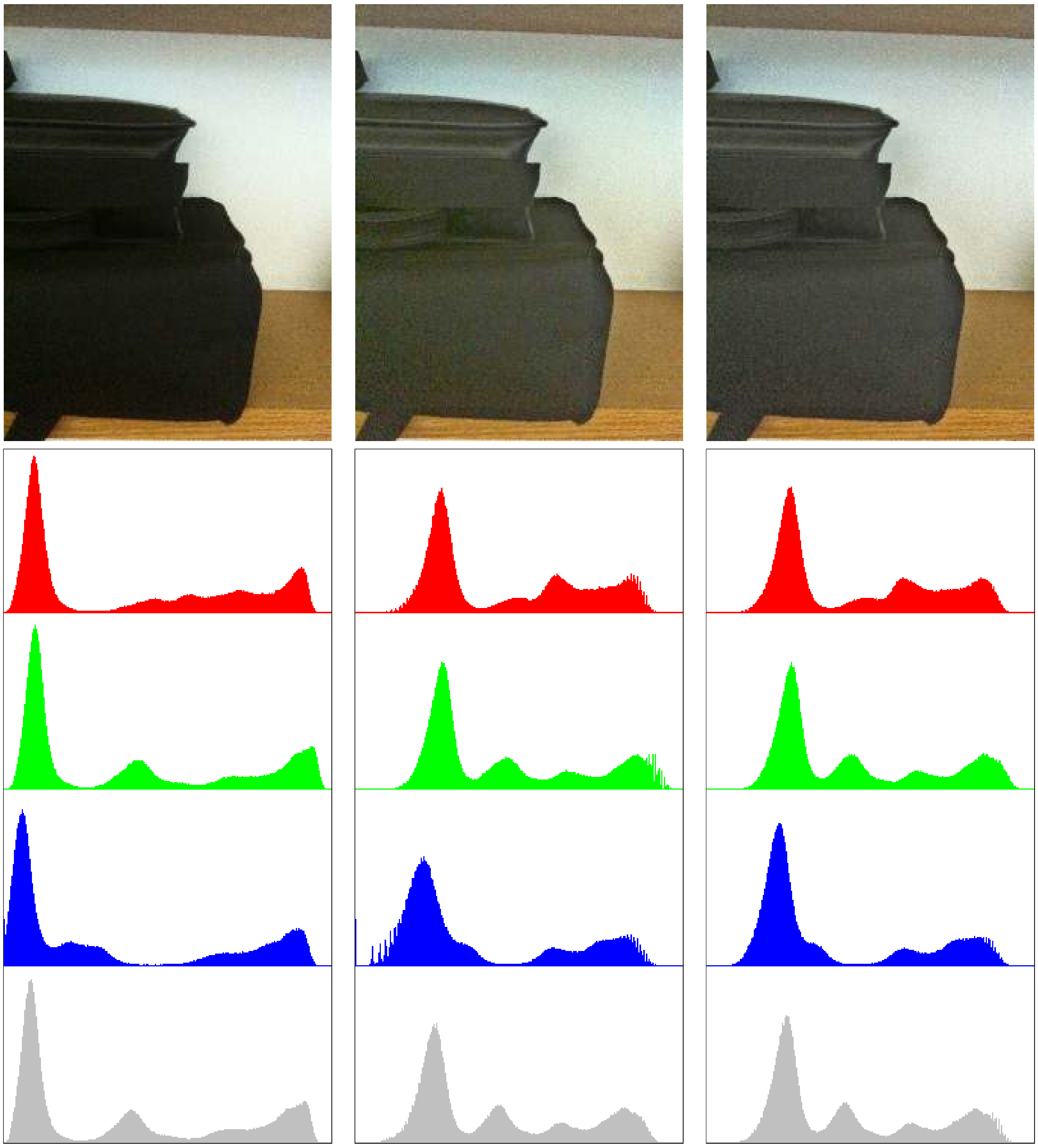


Figure 9: Local Color Correction of a color image, with a fixed value of the parameter ($r = 40$). First row, from left to right: original image, results of algorithms LCC-RGB and LCC-YPbPr. Below each image its R, G, B and intensity histograms are displayed.

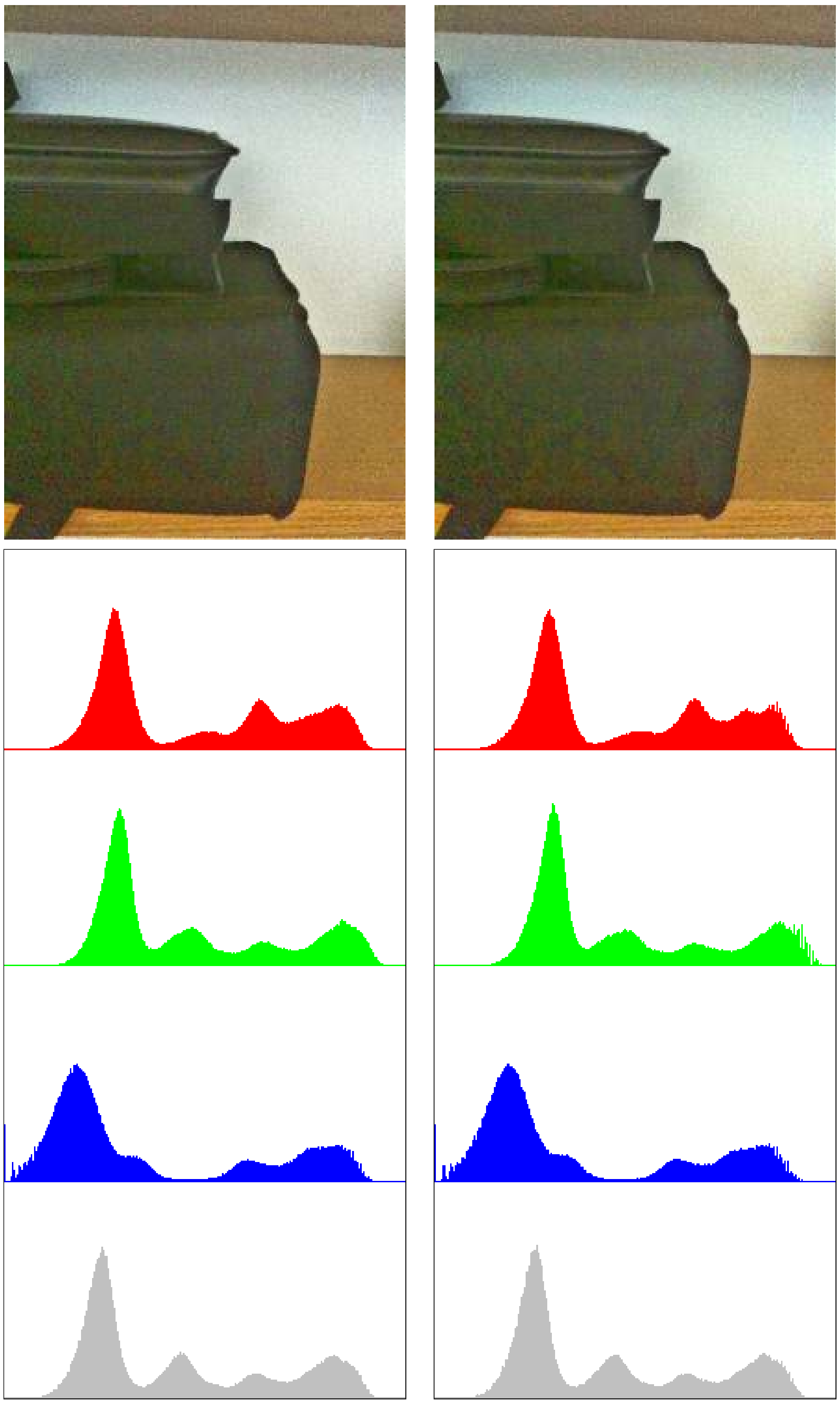


Figure 10: Local Color Correction of a color image, with a fixed value of the parameter ($r = 40$). First row, from left to right, results of algorithms LCC-HSI and LCC-HSL. Below each image its R, G, B and intensity histograms are displayed. These images illustrate the shortcomings of LCC-HSI and LCC-HSL models when applied on dark regions. In this example, a false greenish color appears in the processed image. LCC-RGB and LCC-YPbPr (see Figure 9) perform correctly.

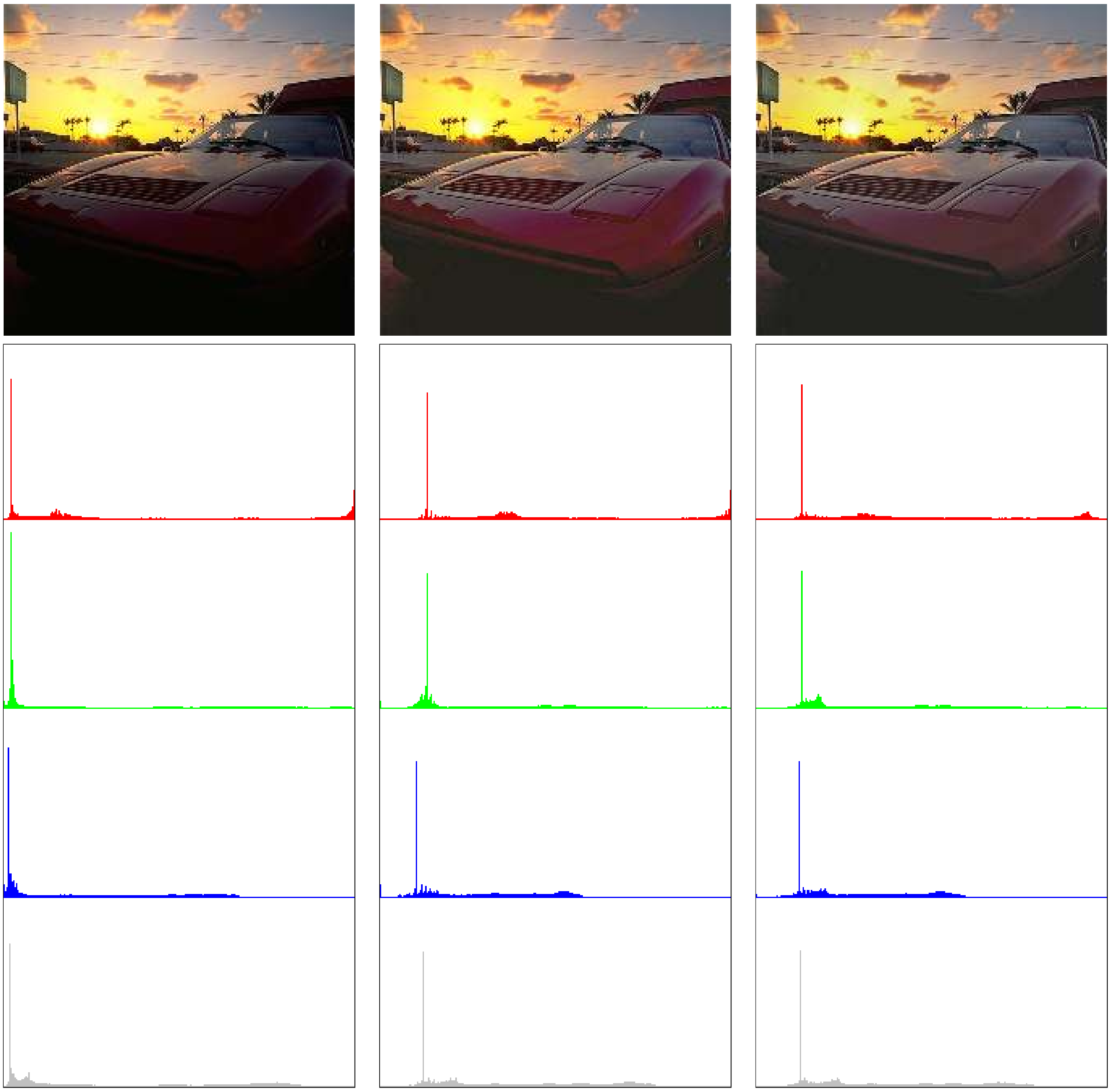


Figure 11: Local Color Correction of a color image, with a fixed value of the parameter ($r = 40$). First row, from left to right: original image, results of algorithms LCC-RGB and LCC-YPbPr. Below each image its R, G, B and intensity histograms are displayed.

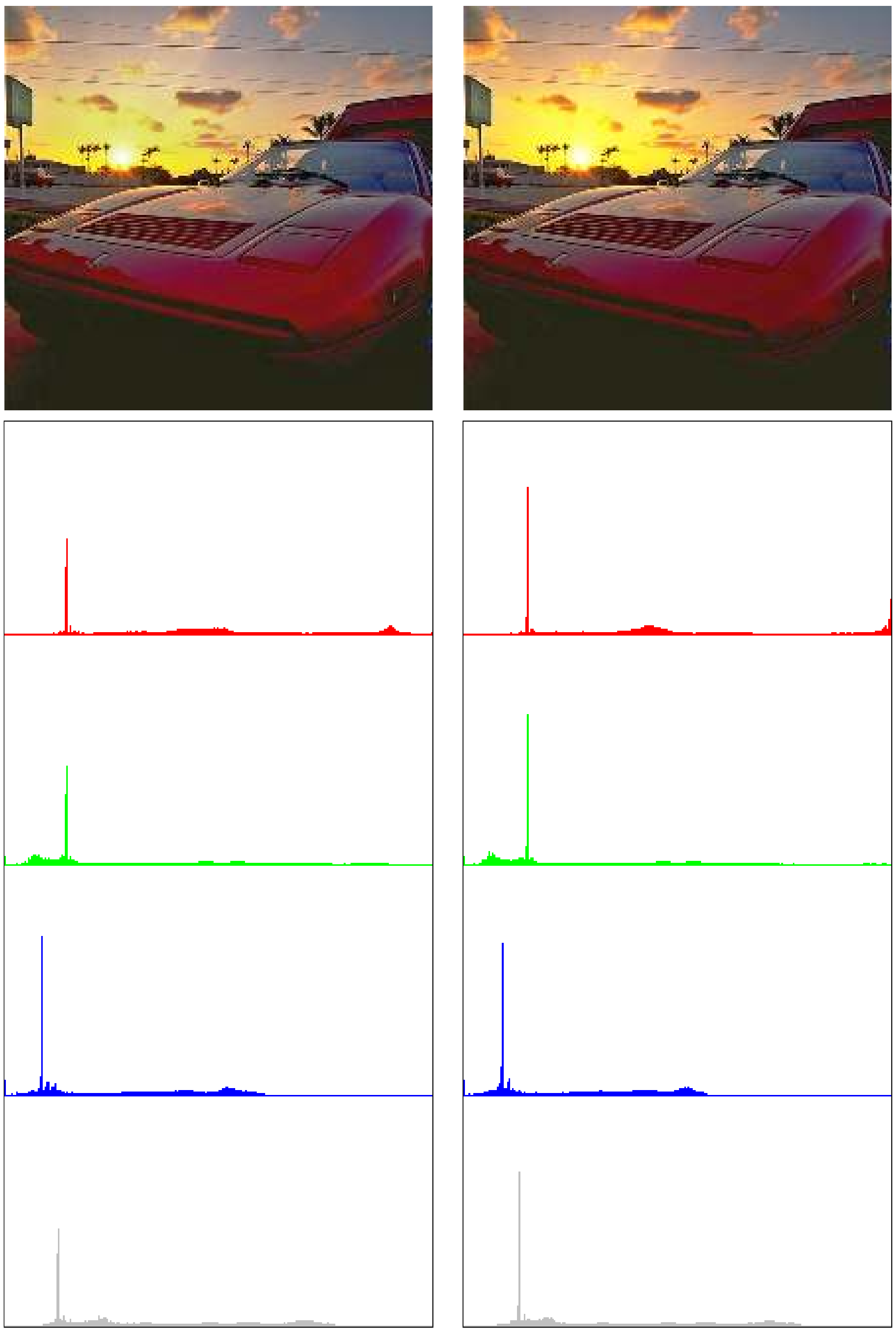


Figure 12: Local Color Correction of a color image, with a fixed value of the parameter ($r = 40$). First row, from left to right, results of algorithms LCC-HSI and LCC-HSL. Below each image its R, G, B and intensity histograms are displayed. These images illustrate the shortcomings of LCC-HSI and LCC-HSL models when applied on dark regions. Using LCC-HSI and LCC-HSL dark red colors in the car become excessively saturated and they look unnatural. In this example LCC-YPbPr (see Figure 11) preserves correctly the original chrominances.

5 Additional Results

Figures 13 to 17 display other results of the different versions of the algorithm, obtained with different values of the parameter.

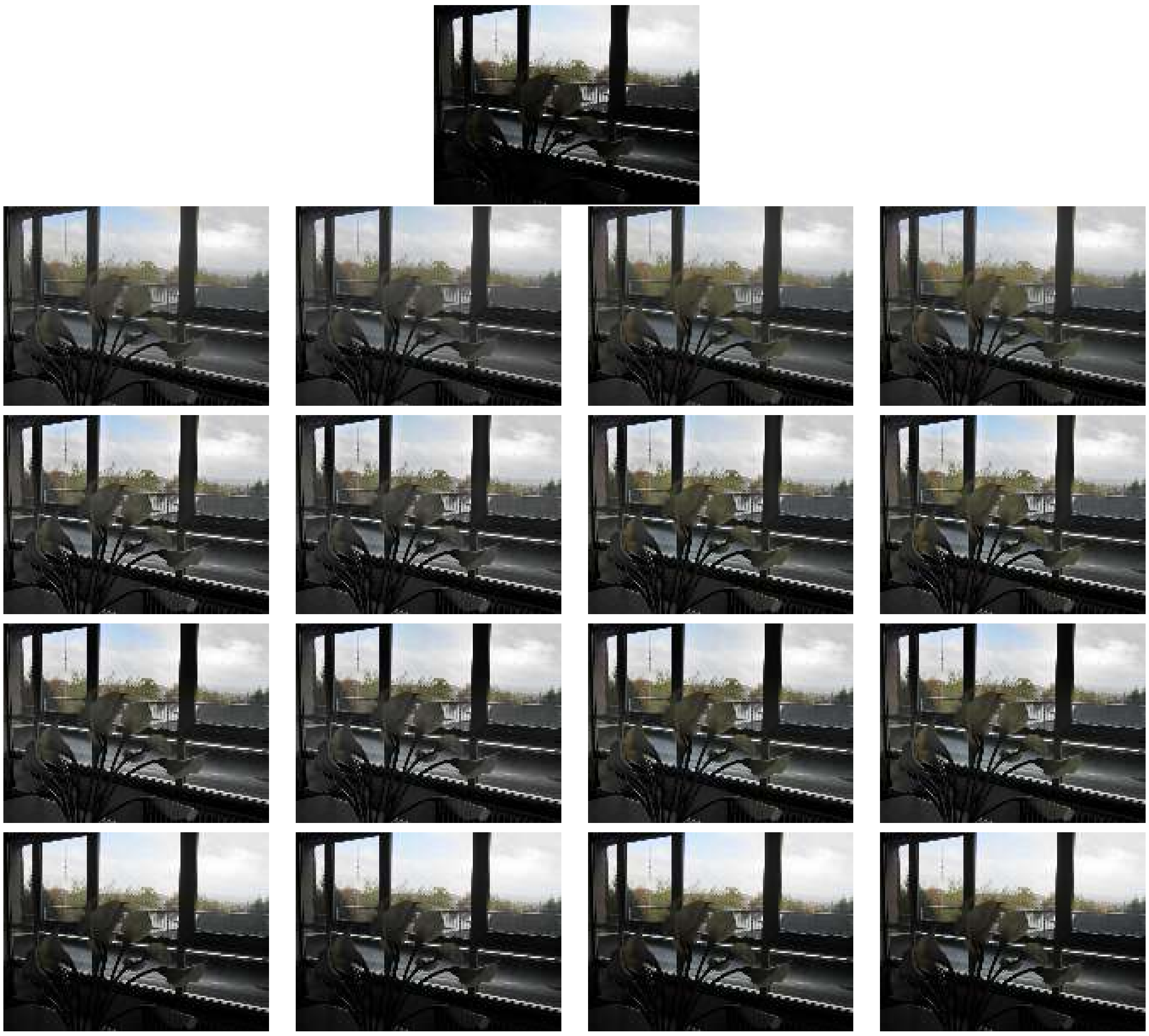


Figure 13: Top, original image. The subsequent rows display (from left to right) the results of LCC-RGB, LCC-YPbPr, LCC-HSI, LCC-HSL for increasing values of the parameter r. From top to bottom, $r = 0$, $r = 40$, $r = 100$. The last row displays the result of global gamma correction with default γ value as defined by (6).

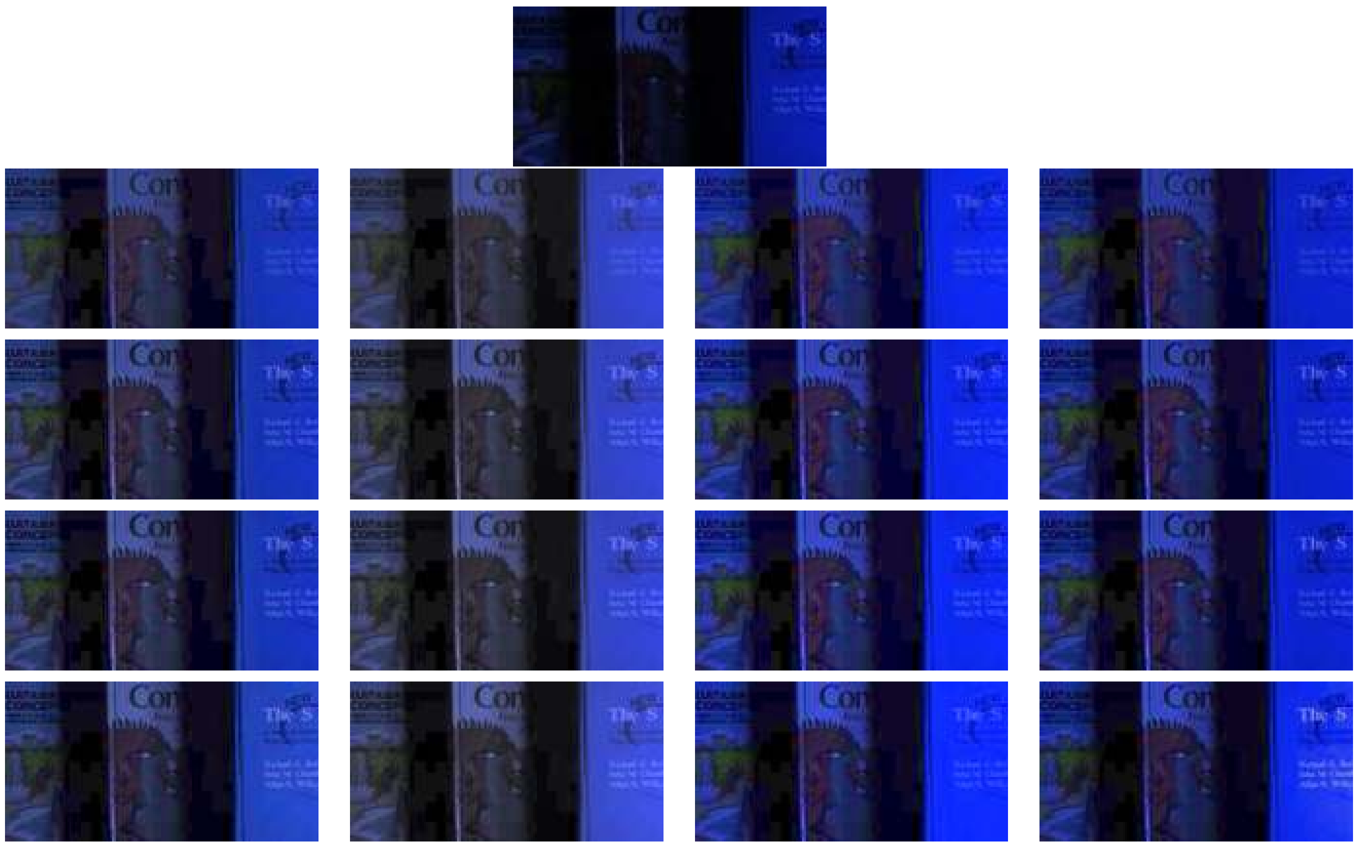


Figure 14: Top, original image. The subsequent rows display (from left to right) the results of LCC-RGB, LCC-YPbPr, LCC-HSI, LCC-HSL for increasing values of the parameter r. From top to bottom, $r = 0$, $r = 20$, $r = 40$. The last row displays the result of global gamma correction with default γ value as defined by (6).

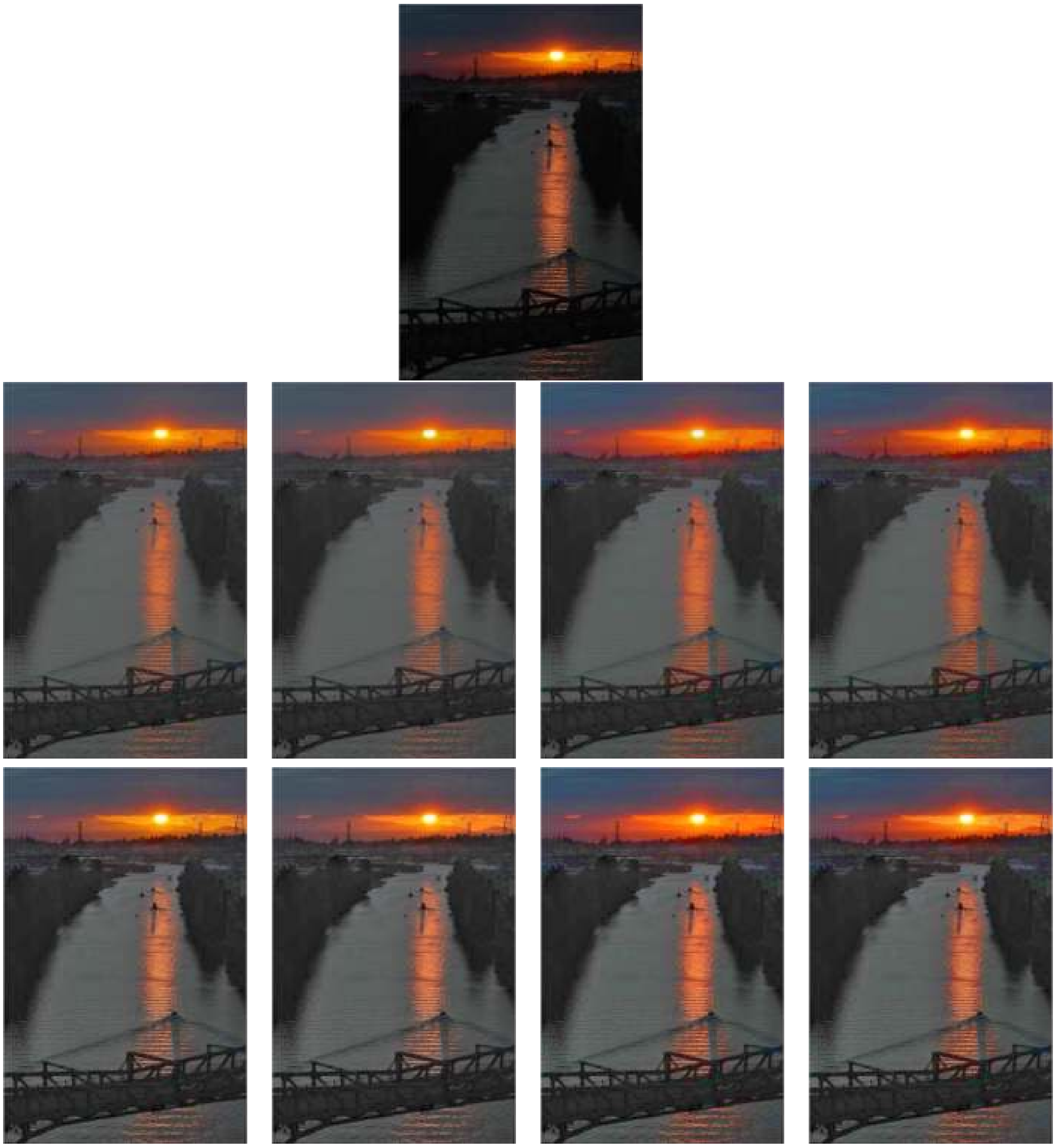


Figure 15: Top, original image. The subsequent rows display (from left to right) the results of LCC-RGB, LCC-YPbPr, LCC-HSI, LCC-HSL for increasing values of the parameter r. From top to bottom, $r = 0$ and $r = 40$.

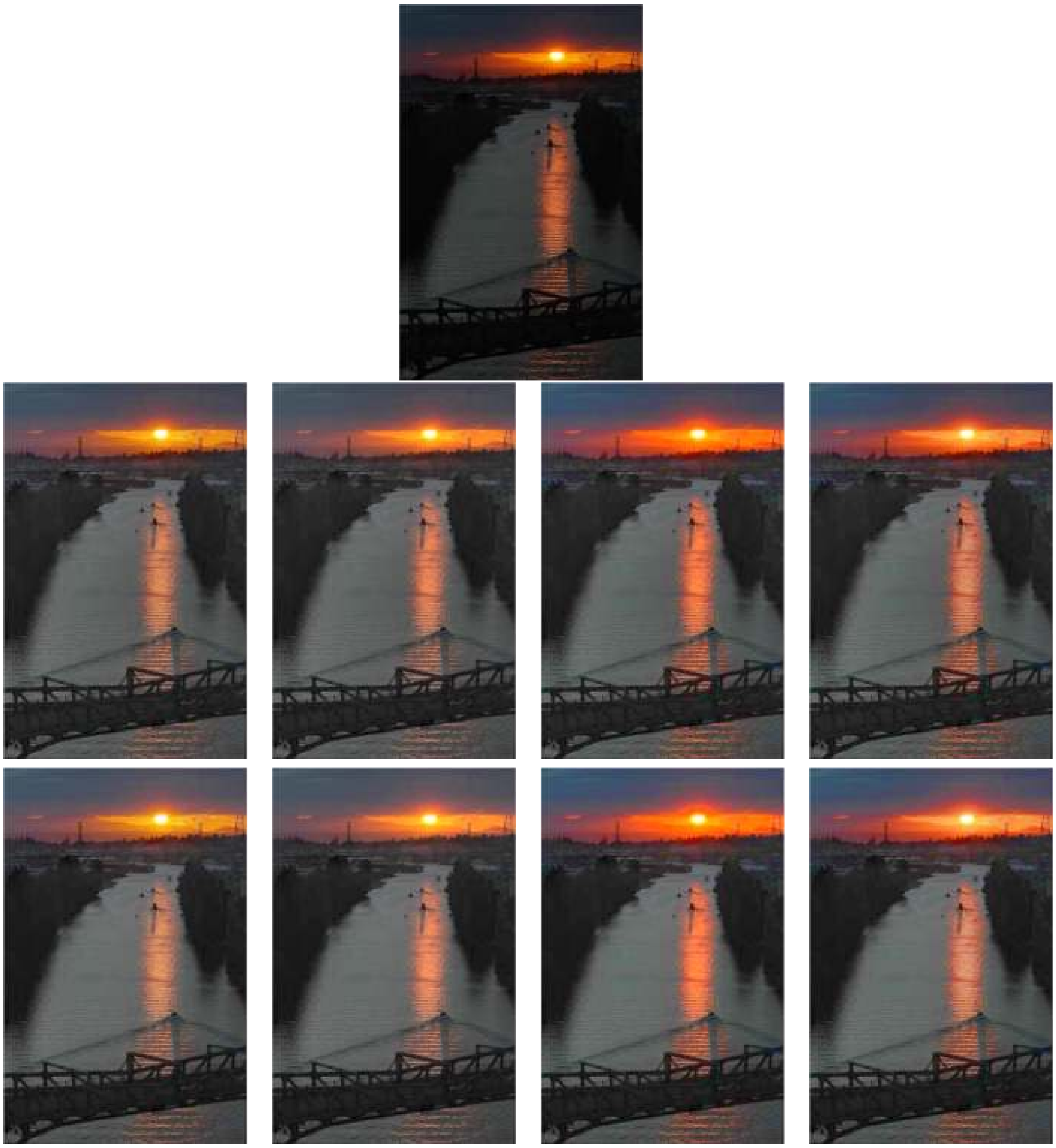


Figure 16: Top, original image. The subsequent rows display (from left to right) the results of LCC-RGB, LCC-YPbPr, LCC-HSI, LCC-HSL for increasing values of the parameter r. Second row, $r = 100$. The last row displays the result of global gamma correction with default γ value as defined by (6).

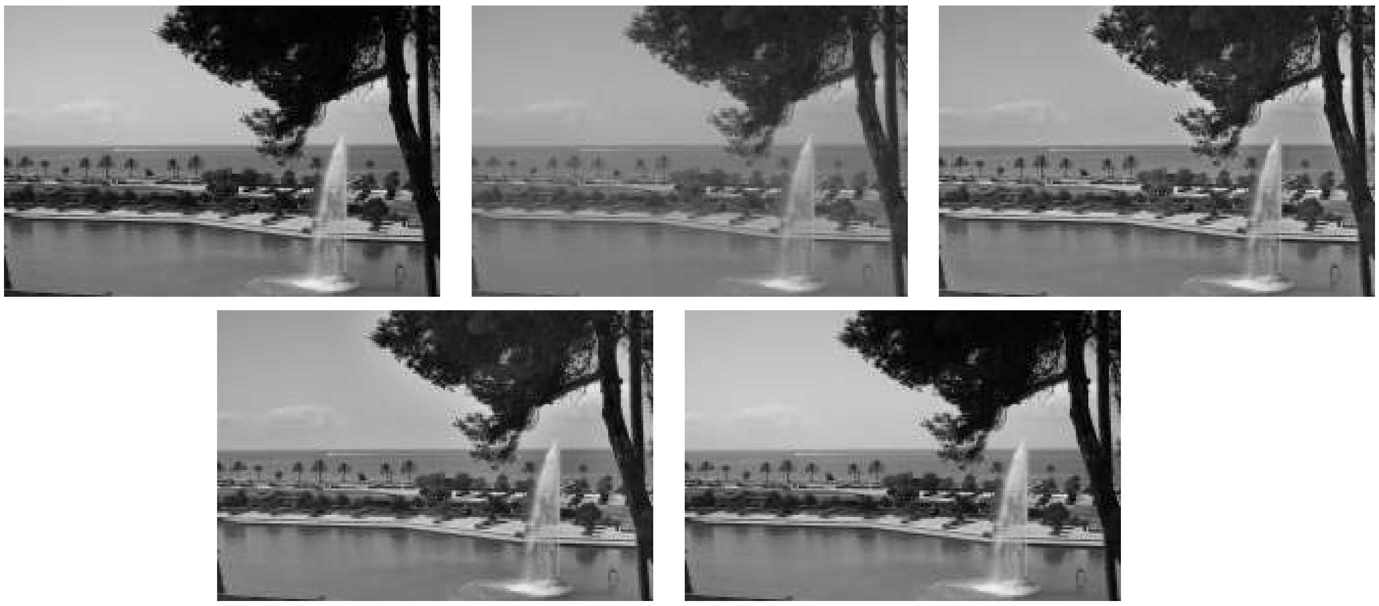
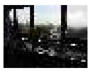
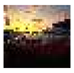


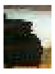
Figure 17: Top, from left to right: original grayscale image, results of LCC for $r = 0$ and $r = 40$. Bottom, result of LCC for $r = 100$ and result of global gamma correction with default γ value as defined by (6).

Image Credits

 CC-BY Juan Gabriel Gomila

 Courtesy Philip Greenspun[6]

Kobus Barnard, SFU Computational Vision Laboratory[7]

 unknown

References

[1] N. Moroney, "Local Color Correction Using Non-Linear Masking", IS&T/SID Eight Color Imaging Conference, pp. 108-111, 2000.

[2] N. Moroney et al., "Local Color Correction", US Patent 6,822,762. November 23, 2004.

[6]http://philip.greenspun.com/
[7]http://www.cs.sfu.ca/~colour/data/