Published in Image Processing On Line on 2011–04–05.
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
http://dx.doi.org/10.5201/ipol.2011.lmps_rpe

Retinex Poisson Equation: a Model for Color Perception

Nicolas Limare[1], Ana Belén Petro[2], Catalina Sbert[3], Jean-Michel Morel[4]

[1] CMLA, ENS Cachan (Paris) (nicolas.limare@cmla.ens-cachan.fr)
[2] University of Balearic Islands (Spain) (anabelen.petro@uib.es)
[3] University of Balearic Islands (Spain) (catalina.sbert@uib.es)
[4] CMLA, ENS Cachan (Paris) (morel@cmla.ens-cachan.fr)

Abstract

In 1964 Edwin H. Land formulated the Retinex theory, the first attempt to simulate and explain how the human visual system perceives color. Unfortunately, the Retinex Land-McCann original algorithm is both complex and not fully specified. Indeed, this algorithm computes at each pixel an average of a very large set of paths on the image. For this reason, Retinex has received several interpretations and implementations which, among other aims, attempt to tune down its excessive complexity. But, Morel et al. have shown that the original Retinex algorithm can be formalized as a (discrete) partial differential equation. This article describes the PDE-Retinex, a fast implementation of the Land-McCann original theory using only two DFT's.

Source Code

The source code, the code documentation, and the online demo are accessible at the IPOL web part of this article[1]. In this link an implementation is available for download. Compilation and usage instruction are included in the README.txt file of the compressed archive.

Keywords: PDE; Retinex; color

1 Overview

In 1964 Edwin H. Land [1] formulated the Retinex theory, the first attempt to simulate and explain how the human visual system perceives color. His theory and an extension, the "reset Retinex" [2] were further formalized by Land and McCann in 1971. Several Retinex algorithms have been developed ever since. These color constancy algorithms modify the RGB values at each pixel to give an estimate of the physical color independent of the shading.

The Retinex original method was complex and imprecise. Indeed, this algorithm computes at each pixel an average of a very large and unspecified set of paths on the image. For this reason, Retinex has received several interpretations and implementations which, among other aims, attempt to tune down its excessive complexity.

[1] http://dx.doi.org/10.5201/ipol.2011.lmps_rpe

But, Morel et al. have shown [3] that the original Retinex algorithm can be formalized as a (discrete) partial differential equation. More precisely, it can be shown that if the Retinex paths are interpreted as symmetric random walks, then Retinex is equivalent to a Neumann problem for a Poisson equation. This result gives a fast algorithm involving just one parameter, also present in the original theory.

The Retinex Poisson equation (given below) is very similar to Horn's [4] and Blake's [5] equations, which were proposed as alternatives to Retinex. It is also one of the "Poisson editing" equations proposed in Perez et al. [6]. The final principle of the algorithm is extremely simple. Given a color image I, its small gradients (those with magnitude lower than a threshold t) in each channel are replaced by zero. The resulting vector field is no more the gradient of a function, but the Poisson equation reconstructs an image whose gradient is closest for the quadratic distance to this vector field. Thus, a new image is obtained, where small details and shades of the original have been eliminated. The elimination of the shades creates more homogeneous colors. This fact, according to Land and McCann, models the property of our perception to perceive constant colors regardless of their shading.

The formalization proved by Morel et al. [3] yields a fast implementation of the Land-McCann original theory using only two DFT's.

2 The PDE-Retinex Model

It is proven [3] that the output of the Retinex algorithm proposed by Land and McCann is the solution of the discrete partial differential equation with Neumann boundary conditions

$$-\Delta_d u(i,j) = F(i,j),$$

where

$$\Delta_d u(i,j) = u(i+1,j) + u(i-1,j) + u(i,j+1) + u(i,j-1) - 4u(i,j),$$

is the discrete Laplacian, $dim = NM$ is the size of the image,

$$F(i,j) = f(I(i,j)-I(i+1,j))+f(I(i,j)-I(i-1,j))++f(I(i,j)-I(i,j+1))+f(I(i,j)-I(i,j-1))$$

and $f(x)$ is a threshold function, whose value is zero if $|x| < t$ and the identity in other case and I is the image to process. This function f eliminates the small variations of the intensity image I.

The parameter t (the threshold) is by default $t = 4$ but one can choose the value depending of the variations that want to eliminate. When I is a gray level image, the algorithm applies to I. When I is a color image, the algorithm is applied to each scalar channel separately.

3 The Algorithm

The output of the algorithm is an image which is the result of the Retinex PDE applied separately to the three channels of the color image, completed by a normalization using the mean and the variance of the original image.

The discrete partial differential equation is easily solved by Fourier transform. To enforce the Neumann boundary condition, the image is first mirrored across its right, and bottom sides to obtain an image four times larger, which is symmetric with respect to its vertical and horizontal medial axes.

The discrete Fourier transform of a two-dimensional function $u(n,m)$ defined on a $N \times M$ grid is defined for (k,l) in $\{0,\dots,M-1\} \times \{0,\dots,N-1\}$ by

$$\widehat{u}(k,l) = \frac{1}{NM} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} u(n,m) e^{-i\frac{2\pi kn}{N}} e^{-i\frac{2\pi lm}{M}},$$

and the discrete inverse Fourier transform for (m,n) in $\{0,\dots,M-1\} \times \{0,\dots,N-1\}$ by

$$u(n,m) = \sum_{k=0}^{N-1} \sum_{l=0}^{M-1} \widehat{u}(k,l) e^{i\frac{2\pi kn}{N}} e^{i\frac{2\pi lm}{M}}.$$

The discrete Fourier transform has the following property

$$u(n-n_0, m-m_0) = \sum_{k=0}^{N-1} \sum_{l=0}^{M-1} \widehat{g}(k,l) e^{i\frac{2\pi kn}{N}} e^{i\frac{2\pi lm}{M}},$$

where

$$\widehat{g}(k,l) = \widehat{u}(k,l) e^{-i\frac{2\pi kn_0}{N}} e^{-i\frac{2\pi lm_0}{M}}.$$

Applying the discrete Fourier transform to the discrete Poisson equation and using this last property yields

$$\widehat{u}(k,l) \left(4 - 2\cos\frac{2\pi k}{N} - 2\cos\frac{2\pi l}{M} \right) = \widehat{F}(k,l),$$

which entails

$$\widehat{u}(k,l) = \frac{\widehat{F}(k,l)}{4 - 2\cos\frac{2\pi k}{N} - 2\cos\frac{2\pi l}{M}}, \text{ for } (k,l) \neq (0,0). \tag{1}$$

Using the inverse Fourier transform we obtain the value of u at each point of the grid, defined up to a constant since the constant Fourier coefficient is arbitrary. The values of u are finally normalized and receive the mean and the variance of the original image. After this normalization some values may fall outside the interval $[0, 255]$. This values are saturated to 0 or 255. All of the above computations are performed on the extended symmetric image F defined on the $2N \times 2M$ grid. F being symmetric, its Fourier coefficients are real. This property is transferred by the equation to the Fourier coefficients of u, and u is therefore also symmetric and verifies the Neumann boundary condition. All of these operations are performed for each channel of the color image, u being in turn the red, green and blue channel.

The algorithm (applied to each channel) therefore is:

Algorithm 1 PDE Retinex

input : t threshold, original image (1 channel)
output: Solution image of the PDE Retinex
Compute $F(i,j)$;
Compute the Fourier transform of F by DFT (symmetrization is handled by the fftw library);
Deduce the Fourier transform of u using Formula (1);
Compute the final solution u by the inverse DFT and apply the normalization.

4 Implementation

The `retinex_pde` implementation is available at the IPOL web part of this article[2].

[2] http://dx.doi.org/10.5201/ipol.2011.lmps_rpe

It should compile on any system since it is only ANSI C. This implementation is used in the demo.

This code requires `libpng`[3] for PNG file input/output and `libfftw3`[4] to process the Fourier transforms.

- Linux. You can install `libpng` and `libfftw3` with your package manager.
- Mac OSX. You can get `libpng` and `libfftw3` from http://www.finkproject.org/, the Fink project.
- Windows. Precompiled DLLs are available online for `libfftw3` and `libpng`; note that `libpng` requires `zlib`[5].

Compilation and usage instructions are provided in the README.txt file.

Implementation notes: the `fftw3 library` supports several Fourier transform types, in particular discrete Fourier transforms of input real data with even/odd symmetry (i.e. cosine/ sine transform). With this kind of mirror symmetries across the boundary there is no need for complex input/output. Moreover, one gains a factor of two in computational time and space. Because of the discrete sampling, this library permits to choose the type of symmetry. The mirror symmetry can be alternatively made with respect to the boundary sample points, or with respect to the points obtained by shifting a half pixel toward the exterior boundary samples. In our implementation we use this second symmetry, which duplicates exactly the image size. After this mirror symmetry the cosine transform implements our equation.

5 Online Demo

An online demo allows you to try Retinex with your own images. The demo has only one parameter, the contrast threshold t present in the original theory.

The uploaded images will be converted to color PNG format and may be resized for an efficient Fourier transform. The images dimensions are kept under 1024 and adjusted to the nearest multiple of 2, 3, 5 and 7 to avoid large primes, and the image is resampled using a cubic spline interpolation. The original non resized images are kept and available in the demo archive. This pre-processing is not in the implementation, it is only added to the demo to ensure fast results.

The aim of the Retinex algorithm is to simulate and explain how the human visual system perceives color, *it is not to improve the image quality*. In the last decade, the "Retinex" trademark has been extended to many color contrast algorithms which actually deviate from the initial Retinex scope. These algorithms successfully enhance the local image contrast and also perform a color balance. The paper of Bertalmio et al. [7] is an example of a fast color enhancement algorithm. You can use the "Simplest Color Balance" algorithm [8] previously to Retinex, for improving the contrast of your image.

The result image, after the Retinex algorithm, is normalized using the mean and the variance of the original image. Thus, the output image and the original image have the same mean and variance and therefore the same global contrast.

[3]PNG Development Group *PNG Reference Library*. http://libpng.sourceforge.net/index.html
[4]Matteo Frigo and Steven G. Johnson. *FFTW package*. http://www.fftw.org/
[5]Greg Roelofs *Zlib homepage*. http://www.zlib.net/

6 Examples

Here are some examples. Note that Retinex is not a model conceived for image enhancement or image improvement; it is only an algorithm to mimic our color perception. Thus, Retinex will enhance an effect that our perception does anyway.

The role of the t threshold is to eliminate the small intensity variations due to shading. Thus t cannot be too large (less than 10 typically) to avoid removing significant details. But is must be large enough to eliminate light shading effects. This is not always possible, since shadows can be very contrasted.

6.1 Adelson's Checker

A first classic example shows the effect of Retinex on the Adelson's checker shadow illusion (Figure 1). In the left image a green cylinder standing on a black and white checker-board casts a diagonal shadow across the board.

The image has been so constructed that the white squares in the shadow, one of which is labeled "B", have actually the very same gray value as the black squares outside the shadow, one of which is labeled "A."

If Retinex is faithful to human perception, it should make B much brighter than A, and it does.

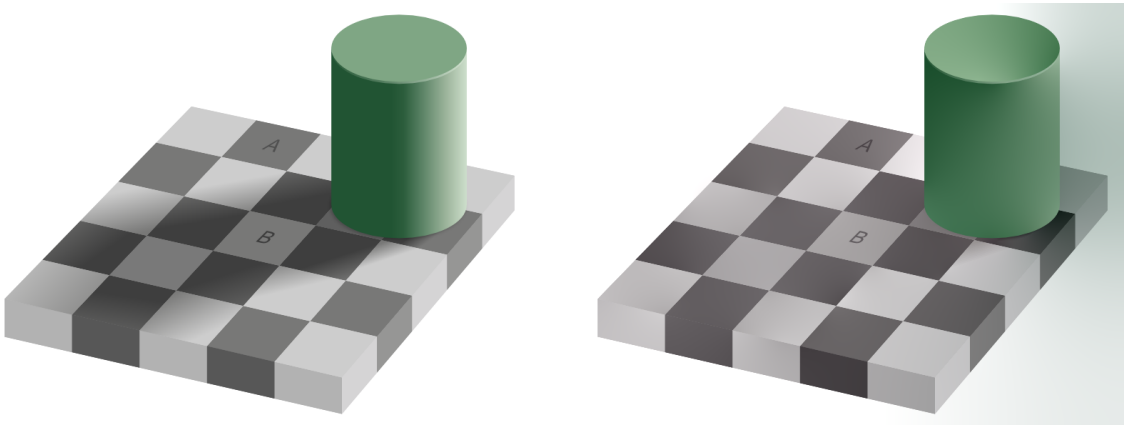


Figure 1: Left: Original image (A and B have a gray value of 120), Right: Retinex result with $t = 3$ (A has a gray value of 120 and B of 160)

6.2 Circles on a Gradient

Simultaneous contrast is a name for the fact that the appearance of a color depends on the colors surrounding it. Figure 2 shows a background with a smooth, but intense, variation and two circles with the same gray value (170). One of them is placed in the darker part of the image, and the other one in the brighter part.

The usual perception is that the circle in the darker part looks conspicuously brighter than the other. If we use a threshold $t = 3$ larger than the background variation, the result is an image with nearly constant background (from 105 to 140). The left circle gets a 0 gray value and the right circle a 255 gray value, which predicts well our perception.

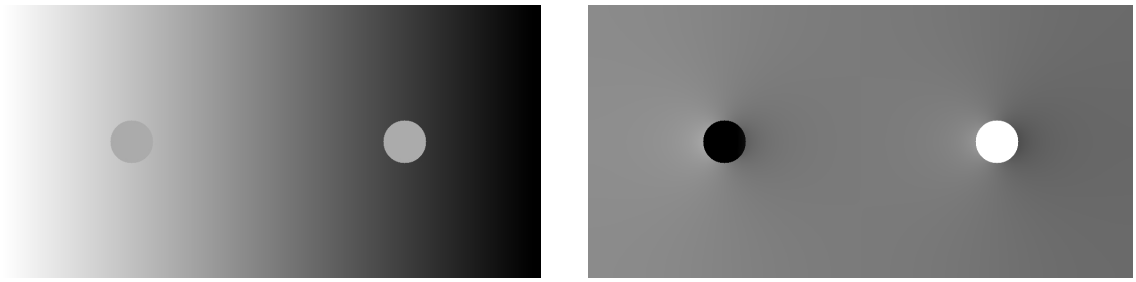


Figure 2: Left: Original image, white balanced. Right: Retinex result.

6.3 Noisy Image

To understand the effect of the threshold t Figure 3 shows a noisy original and the result of Retinex with increasing threshold values $t = 1$, $t = 3$ and $t = 5$.

The background clutter and the shades are progressively filtered out when t increases, but the main edges are kept. At $t = 5$, however, edges start loosing contrast and low contrasted details start disappearing.

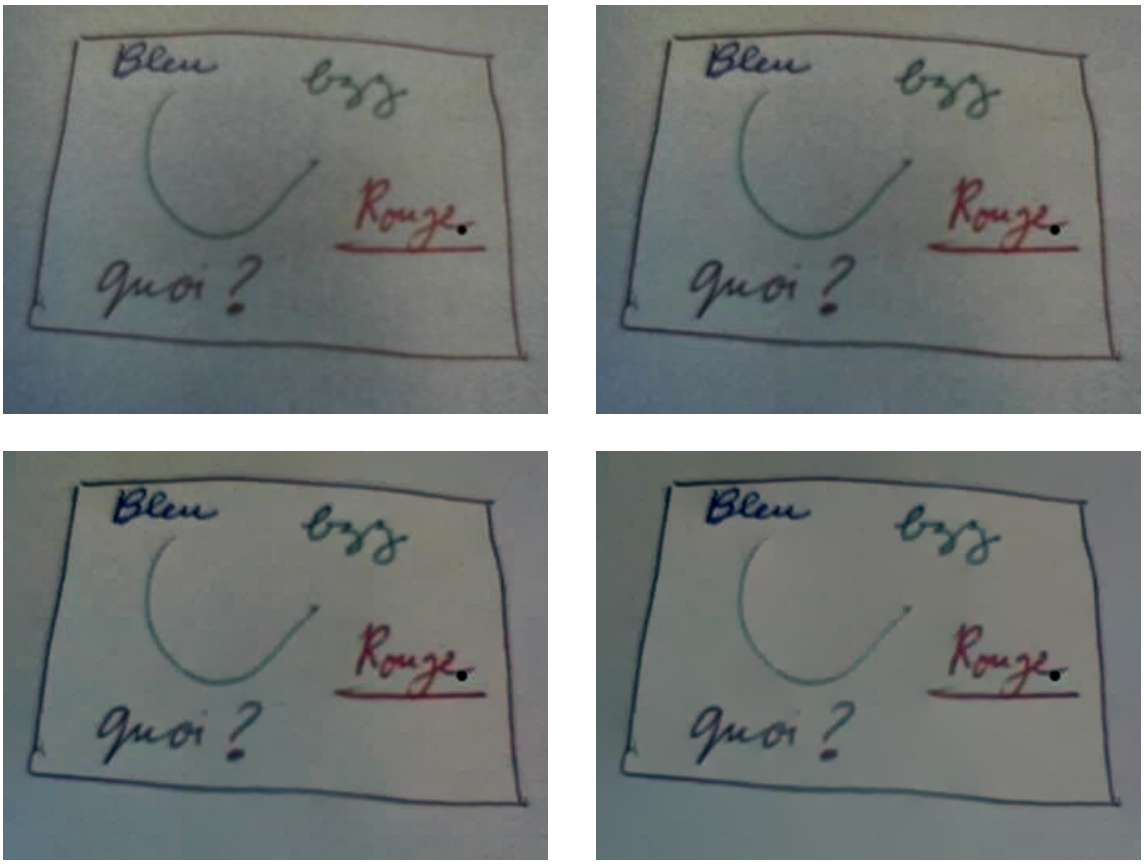


Figure 3: Comparison between the results with increasing threshold values. Top left: original image. Top right: Retinex result with $t = 1$. Bottom left: Retinex result with $t = 3$. Bottom right: Retinex result with $t = 5$.

As we have mentioned before, the Retinex method is not a model for image enhancement or image improvement. We could apply other methods for image enhancement, previously to Retinex,

to obtain better results. For example, to Figure 3 we can apply the "Simplest Color Balance" algorithm [8] and we can observe the better results (Figure 4).

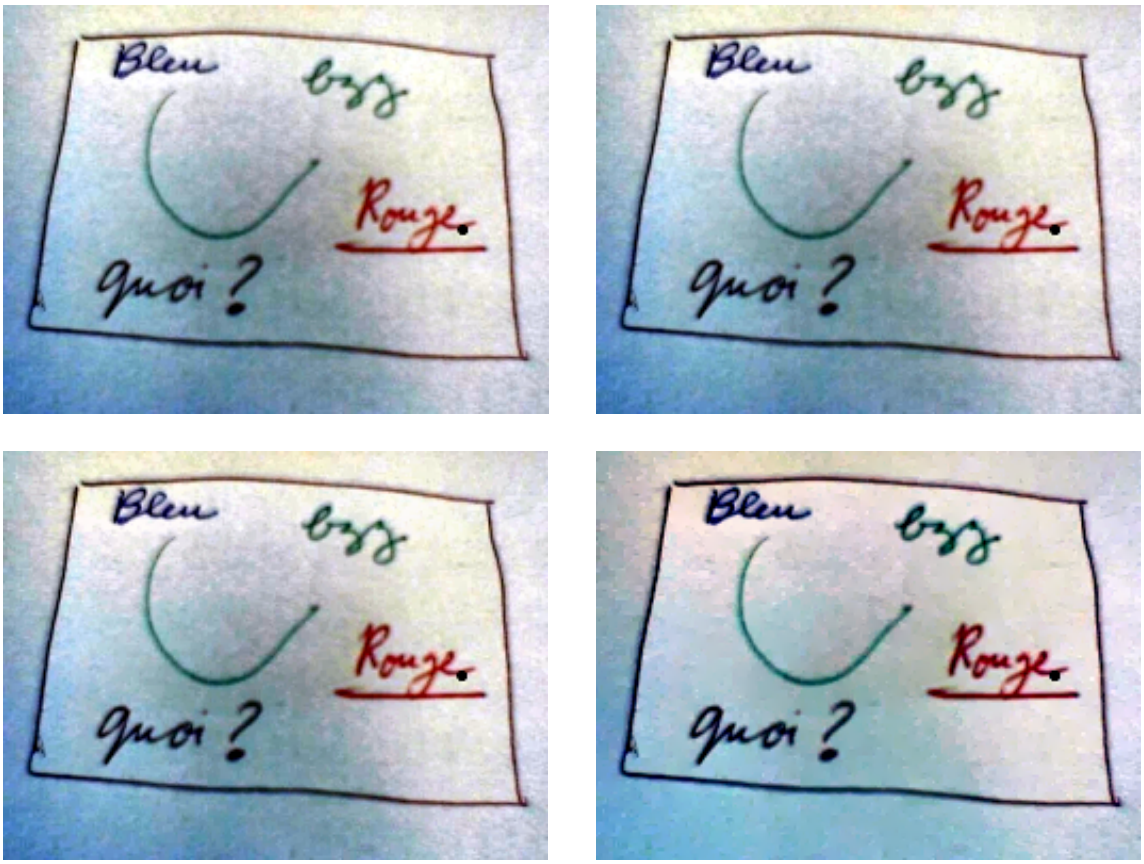


Figure 4: Comparison between the results with increasing threshold values. Top left: White balanced image. Top right: Retinex result with $t = 3$. Bottom left: Retinex result with $t = 5$. Bottom right: Retinex result with $t = 10$.

6.4 Shadows Removal

Figure 5 and Figure 6 demonstrate how Retinex can be used for removing shadows. This is not always effective, but here are two good examples. The shadow removal works only if the boundary of the shadow is blurry, and therefore has a small gradient.

6.5 Lena

Figure 7 is the Retinex application to the image "Lena". The smooth shading variations on the shoulder or the face and in the background fade out when the threshold t increases.

6.6 Limitation of the Method

The Retinex method has been used by several authors as a model for image enhancement or image improvement. But this apparent improvement when it occurs is simply due to the image normalization or the color restoration applied after having applied the Retinex algorithm. Thus the very same contrast improvement can be obtained by a simple color balance algorithm. To avoid mixing up any

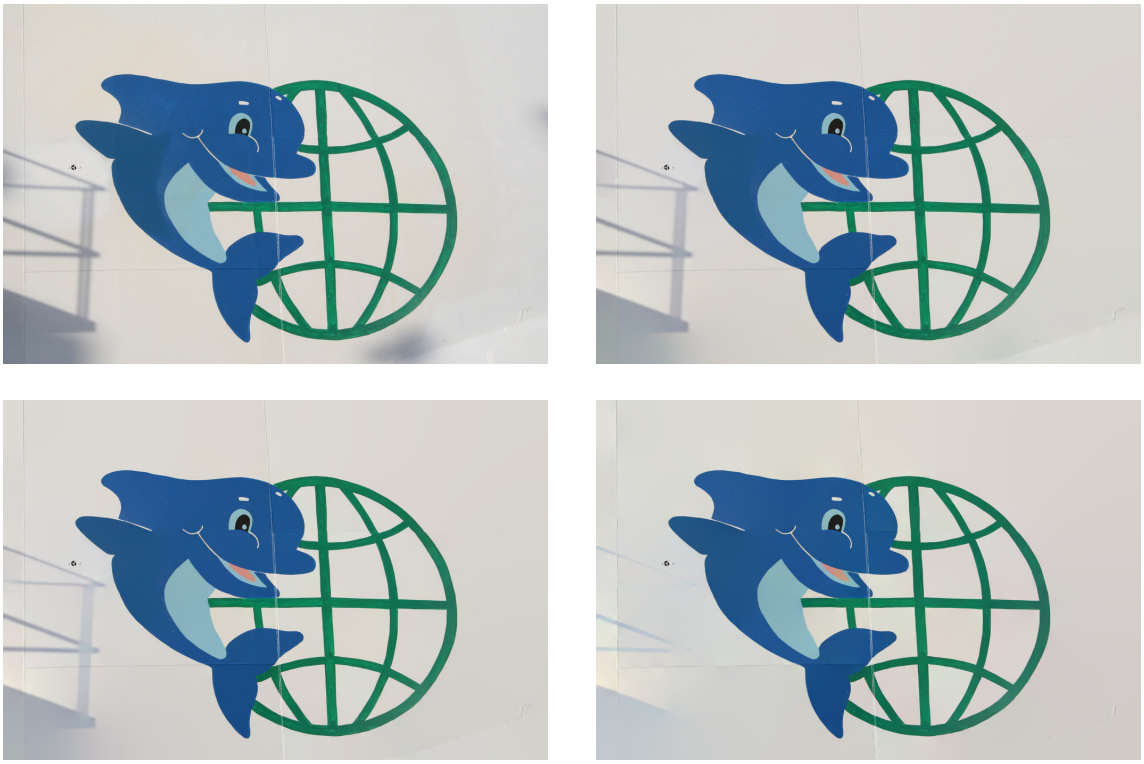


Figure 5: Retinex can be used for removing shadows. Top left: Original image. Top right: Retinex result with $t = 3$. Bottom left: Retinex result with $t = 5$. Bottom right: Retinex result with $t = 10$.

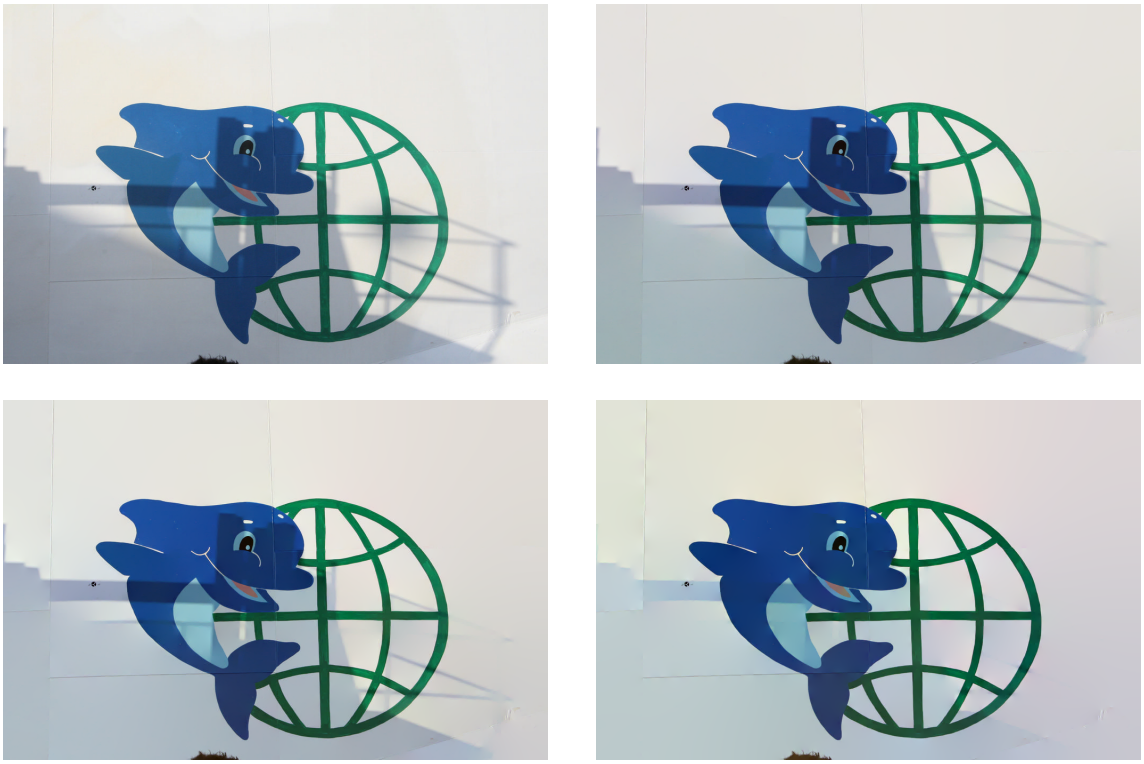


Figure 6: Retinex can be used for removing shadows. Top left: Original image. Top right: Retinex result with $t = 3$. Bottom left: Retinex result with $t = 5$. Bottom right: Retinex result with $t = 10$.

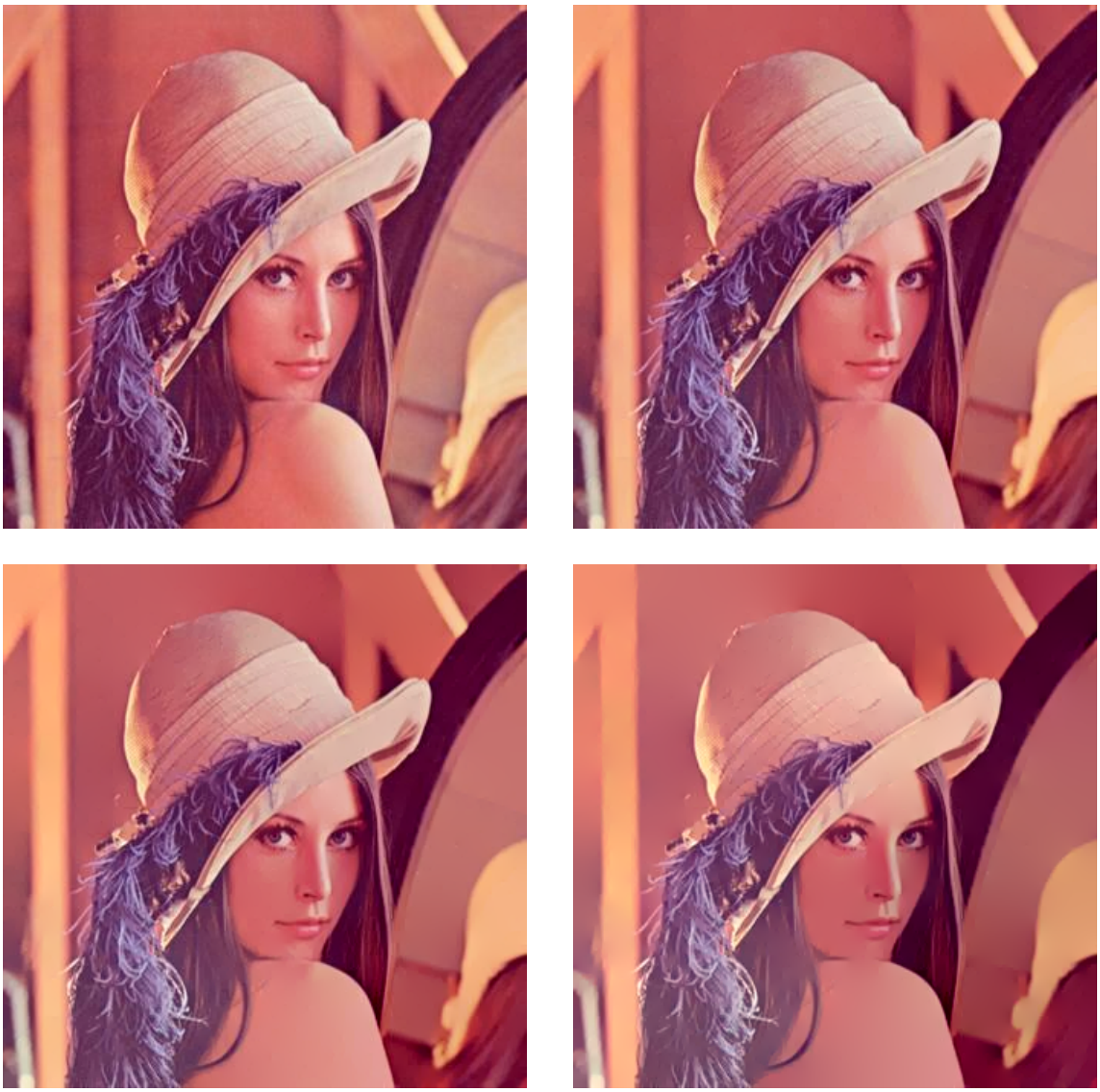


Figure 7: Application to "Lena". Top left: Original image. Top right: Retinex result with $t = 3$. Bottom left: Retinex result with $t = 5$. Bottom right: Retinex result with $t = 10$.

color balance effect with the Retinex effect, the algorithm always applies a normalization keeping for the Retinex result the mean and the variance of the original image. Retinex should *never* improve the contrast of the image; its perceptual effect is "color constancy": it simply flattens the color in low gradient regions. That's all.

Thus, dark images or bad quality images should not improve with the mere application of the Retinex method. An example of these effects can be observed in Figure 8. The application of Retinex only "flattens" the image. On the other hand, the simplest color balance algorithm yields a significant improvement of the image.

Figure 8: Limitation of the method. Left: Original image. Center: Retinex result with $t = 3$. Right: White balance image.

Figure 9 is another example with a dark image. There is no significant difference between the original image and the Retinex result. The result of the simplest color balance algorithm instead is a serious improvement of the image quality.

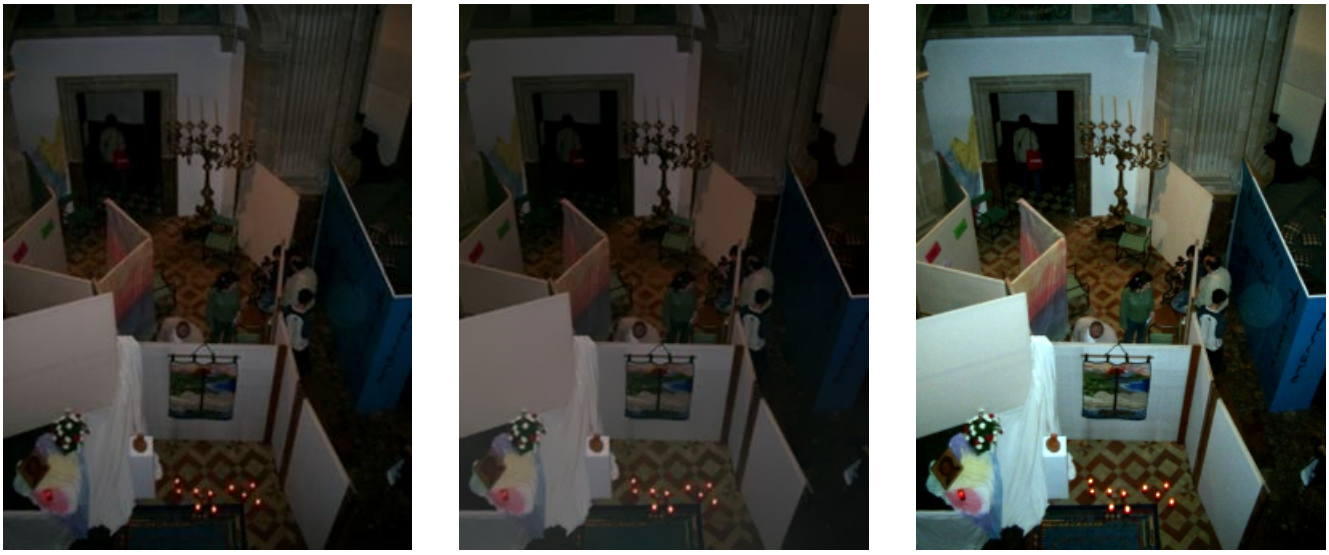


Figure 9: Limitation of the method. Left: Original image. Center: Retinex result with $t = 3$. Right: White balance image.

Since it only alters small non zero gradients, the Retinex algorithm does not produce any change in images having only completely flat areas. The synthetic image in Figure 10 shows this phenomenon. Observe that the original image is identical to the Retinex result.

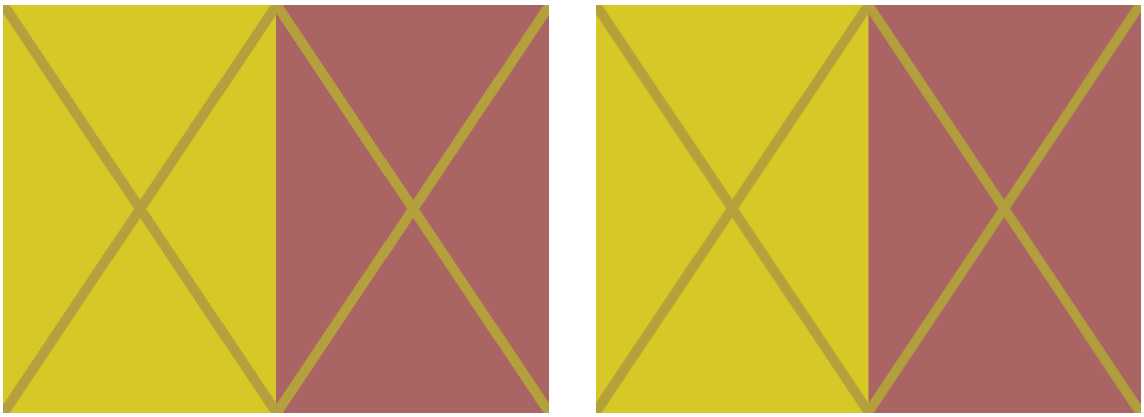


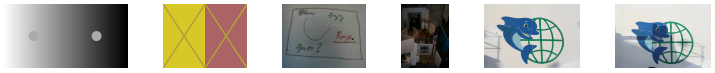
Figure 10: Limitation of the method. Left: Original image. Right: Retinex result with $t = 4$.

Acknowledgment

The authors thank the referees, Gabriele Facciolo and Vicent Caselles, for their very valuable corrections and comments.

Image credits

 Edward H. Adelson[6]

 The authors CC-BY.

 Courtesy Philip Greenspun[7]

 Standard test image

References

[1] E.H. Land, *The Retinex*. American Scientist. 52(2): 247-64. (1964).

[2] E.H. Land and J.J. McCann, *Lightness and Retinex Theory*. Journal of the Optical Society of America, 61, 1–11 (1971).

[3] J.-M. Morel, A.B. Petro and C. Sbert, *A PDE Formalization of the Retinex Theory*, IEEE Transactions on Image Processing, 19 (11), 2825–2387, (2010). http://dx.doi.org/10.1109/TIP.2010.2049239

[4] B.K. Horn, *Determining Lightness from an Image*. Computer Graphics and Image Processing, 3, 277–299 (1974). http://dx.doi.org/10.1016/0146-664X(74)90022-7

[6] http://web.mit.edu/persci/people/adelson/checkershadow_illusion.html
[7] http://philip.greenspun.com/

[5] A. Blake, *Boundary Conditions of Lightness Computation in Mondrian World*. Computer Vision Graphics and Image Processing, 32, 314–327 (1985). http://dx.doi.org/10.1016/0734-189X(85)90054-4

[6] P. Pérez, M. Gangnet and A. Blake, *Poisson Image Editing*. ACM Transactions on Image Processing. Proceedings of ACM SIGGRAPH, 22 (3), 313–318 (2003). http://doi.acm.org/10.1145/882262.882269

[7] M. Bertalmio, V. Caselles, E. Provenzi and A. Rizzi, *Perceptual Color Correction Through Variational Techniques.* IEEE Transactions on Image Processing. 16(4), 1058–1072, (2007). http://dx.doi.org/10.1109/TIP.2007.891777

[8] N. Limare, J.L. Lisani, J.-M. Morel, A.B. Petro, C. Sbert, *Simplest Color Balance*, Image Processing On Line, Vol. 1, (2011), http://dx.doi.org/10.5201/ipol.2011.llmps-scb