IPOL IMAGE PROCESSING ON LINE

Published in Image Processing On Line on 2011–09–13.
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.blmv_ct

Cartoon+Texture Image Decomposition

Antoni Buades[1], Triet Le[2], Jean-Michel Morel[3], Luminita Vese[4]

[1] TAMI, Universitat de les Illes Balears, SPAIN (toni.buades@uib.es)
[2] Mathematics Department, Yale University, USA ({triet.le@yale.edu)
[3] CMLA, ENS Cachan, France (moreljeanmichel@gmail.com)
[4] Mathematics Department, University of California at Los Angeles, USA (lvese@math.ucla.edu)

Abstract

In this article we give a thorough description of the algorithm proposed in [A. Buades, T. Le, J.M. Morel and L. Vese, Fast cartoon + texture image filters, IEEE Transactions on Image Processing, 2010] for cartoon+texture decomposition using of a nonlinear low pass-high pass filter pair.

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: cartoon; texture; nonlinear filters

1 Introduction

The algorithm first proposed in [3] stems from a theory proposed by Yves Meyer in [1]. The cartoon+texture algorithm decomposes any image f into the sum of a cartoon part, u , where only the image contrasted shapes appear, and a textural v part with the oscillating patterns. Such a decomposition $f = u + v$ is analogous to the classical signal processing low pass-high pass filter decomposition. However, the cartoon part of an image actually contains strong edges, and therefore all frequencies, up to the high ones, while a texture can also contain middle and high frequencies. Thus, linear decomposition algorithms cannot make a clear cut separation between cartoon and textures. They blur out edges and take their high frequencies into the texture part. Conversely, they leave behind some texture in the low pass filtered cartoon part. Yves Meyer proposed to solve the problem by a variational problem containing two norms: the right decomposition $f = u + v$ is the one where the cartoon part u has minimal total variation while the oscillatory component has a minimal norm in a dual space of BV. This second norm does not penalize oscillation: the higher the frequency of the oscillation of v, the smaller its norm. In [3] one can find a detailed history and analysis of variational algorithms and variants for the original Meyer formulation, which seems to start with [2].

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

The many variants proposed in the literature consider various functional spaces for the textural part (dual of BV, Besov spaces, etc.).

In this article we give a thorough description of the algorithm proposed in [3]. It is a fast approximate solution to the original variational problem obtained by applying a nonlinear low pass-high pass filter pair. The algorithm proceeds as follows. For each image point, a decision is made of whether it belongs to the cartoon part or to the textural part. This decision is made by computing a local total variation of the image around the point, and comparing it to the local total variation after a low pass filter has been applied.

Edge points in an image tend to have a slowly varying local total variation when the image is convolved by a low pass filter. Textural points instead show a strong decay of their local total variation by convolution with a low pass filter. The cartoon+texture filter pair is based on this simple observation.

The cartoon part keeps the original image values at points termed as non-textural points. At points identified as texture points, the cartoon part takes the filtered value. At points where the cartoon/texture decision is ambiguous, a weighted average of them is given. The texture part simply is the difference between the original image and its cartoon part. As pointed out in [3], although the algorithm does not claim to exactly solve the original variational problem, it retains its inspiration, and brings in a transparent user parameter, the scale of the texture, to specify the decomposition.

2 The Scale Parameter

There is no unique decomposition of an image into texture and cartoon. A texture seen at close range is just a set of well-distinguished objects, such as leaves, bubbles, stripes, or straws. Thus, it can be kept in the cartoon part for low values of the scale parameter, and passes over to the textural part for larger scales.

The scale parameter in the algorithm is therefore crucial, and must be chosen by the user. However, a default value is proposed in the first trial. The scale parameter is measured in pixel size. Thus $\sigma = 2$ means roughly that the texture half-period is 2 pixels. With $\sigma = 2$, only the finest textures are distinguished.

In general, humans perceive image regions as textures for values ranging from $\sigma = 3$ to 6. Over this last value, the textures are made of well distinguished and contrasted objects, and the decision to view them as a texture is definitely subjective.

3 Algorithm

The main characteristics of a textured region is its high total variation. The formalization of this remark leads to define the local total variation (LTV) for every pixel x

$$LTV_\sigma(x)(f) := G_\sigma * |\nabla f|(x),$$

where G_σ is a Gaussian kernel with standard deviation σ.

The relative reduction rate of LTV is defined by

$$\lambda(x) := \frac{LTV_\sigma(x)(f) - LTV_\sigma(x)(L_\sigma * f)}{LTV_\sigma(x)(f)},$$

being L_σ a low pass filter. As LTV decreases very fast under low pass filtering, $\lambda(x)$ gives us the local oscillatory behavior of the function.

$$\frac{LTV_\sigma(x)(f) - LTV_\sigma(x)(L_\sigma * f)}{LTV_\sigma(x)(f)} = \lambda \quad \Longleftrightarrow \quad LTV_\sigma(x)(L_\sigma * f) = (1-\lambda) LTV_\sigma(x)(f).$$

If λ is close to 0, there is little relative reduction of the local total variation by the low pass filter. If instead λ is close to 1 the reduction is strong, which means that the considered point belongs to a textured region. Thus, a fast nonlinear low pass and high pass filter pair can be computed by doing weighted averages of f and $L_\sigma * f$ depending on the relative reduction of LTV

$$u(x) = \omega(\lambda(x))L_\sigma * f + (1 - \omega(\lambda(x)))f, \qquad v(x) = f(x) - u(x),$$

where $w(x) : [0, 1] \to [0, 1]$ is a nondecreasing piecewise affine function that is constant and equal to zero near zero and constant and equal to 1 near 1 (Figure 1).

In all experiments the soft threshold parameters defining w have been fixed to $a_1 = 0.25$ and $a_2 = 0.5$. If $\lambda(x)$ is small the function f is non-oscillatory around x and therefore the function is BV around x. Thus $u(x) = f(x)$ is the right choice. If instead λ is large, the function f is locally oscillatory and locally replaced by $L_\sigma * f$.

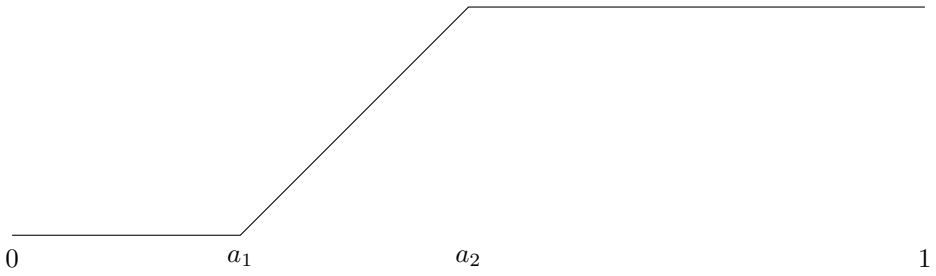

Figure 1: The soft threshold function $w(s)$.

The choice of $\lambda = 1/2$ as underlying hard threshold is conservative: it ensures that all step edges stay on the cartoon side, but puts all fine structures on the texture side, as soon as they oscillate more than once. Since it is desirable to have a one-parameter method, λ is fixed once and for all. In that way the method keeps the scale of the texture as the only method parameter.

4 Implementation

As indicated in the algorithm, the cartoon+texture decomposition only requires the application on the gradient image of two low-pass filters, which are performed directly by a discrete convolution. The gradient is computed by the simplest centered difference scheme. The main steps are:

1. Apply a low pass filter to the initial image f.

   The low pass filtered image $L_\sigma * f$ is obtained by convolving f with the low pass filter $L_\sigma = (Id - (Id - G_\sigma)n)$, indicating n that the convolution is iterated n times and being n fixed to 5. Convolutions are computed in space with mirror boundary conditions, that is, the image is symmetrized out of its domain. In the current implementation this low pass filtered image is obtained iterativelly (see Algorithm 1).

   We preferred this low pass filter to a Fourier based filter as proposed in [3], for simplicity in the coding. Figure 2 illustrates the low pass-high pass behavior of the proposed filter L_σ and the corresponding $H_\sigma = Id - L_\sigma$ for several values of n.

2. Compute the Euclidian norm of the image gradients of f and $L_\sigma * f$.

Algorithm 1: Iterative low-pass filtering.

low $\leftarrow G_\sigma * f$
high $\leftarrow f$ - low
for $i = 1 \dots n$ **do**
  high $\leftarrow$ high - $G_\sigma *$ high
  low $\leftarrow f$ - high

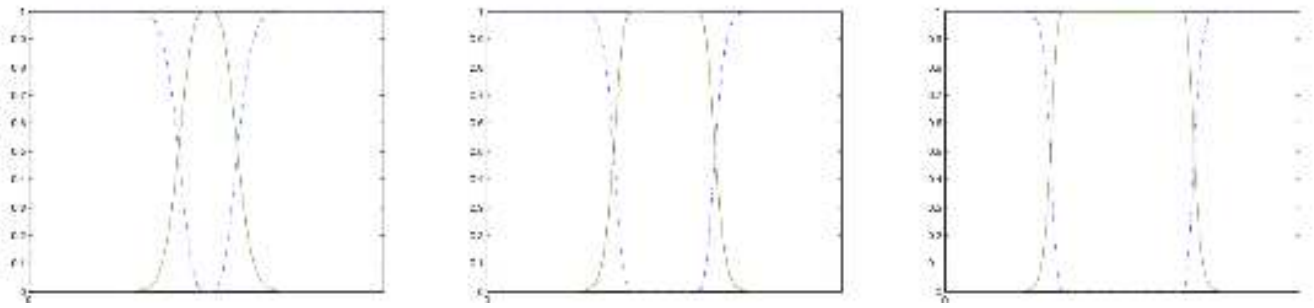

Figure 2: L_σ and the corresponding $H_\sigma = Id - L_\sigma$ for several values of n.

The vertical and horizontal derivatives are computed by a centered two point scheme and the modulus of the gradient with an Euclidean norm.

$$u_x(i,j) = u(i+1,j) - u(i-1,j)$$

$$u_y(i,j) = u(i,j+1) - u(i,j-1)$$

$$|\nabla u| = \sqrt{u_x(i,j)^2 + u_y(i,j)^2}$$

3. Convolve these moduli with the Gaussian G_σ to get the local total variation of f and $L_\sigma * f$. Convolutions are computed in space with mirror boundary conditions.
4. Deduce the value of $\lambda(x)$ at each point in the image.
5. Deduce the value of the cartoon image as a weighted average of f and $L_\sigma * f$.
6. Compute the texture as the difference $u - f$.

The color implementation for a color image $f = (r, g, b)$ is as follows:

1. Apply a low pass filter independently to each channel of the image f.
2. Compute the local total variation of each channel of the original and low pass filtered images. Compute the color local total variation as the average of the red, green and blue local total variations.
3. Deduce the value of $\lambda(x)$ at each point in the image by using the color local total variation. That is, the same function $\lambda(x)$ is used for the three channels.
4. Deduce the value of the cartoon image as a weighted average of each channel of f and $L_\sigma * f$.
5. Compute the texture as the difference $u - f$.

5 Examples

5.1 Cactus

Examples in Figure 3 illustrate several aspects of the decomposition. On the cactus image, which is large, a scale 5 is just enough to remove some detail. The borders of the cactus leaves are in no way step edges. In fact for many of them the color oscillates strongly near the edges, and is considered as texture.

Figure 3: From left to right: original, cartoon, texture. Computed with scale parameter 5.

5.2 Noisy Square

The second image (Figure 4), a noisy square on noisy background, is correctly divided into a smooth cartoon with sharp boundary and a noise texture. Notice an undesirable adhesion effect: points near the edges are considered edge points, and therefore their texture is kept in the cartoon part. This adhesion effect is observable in all images and would only disappear if the isotropic filters were replaced by anisotropic filters.

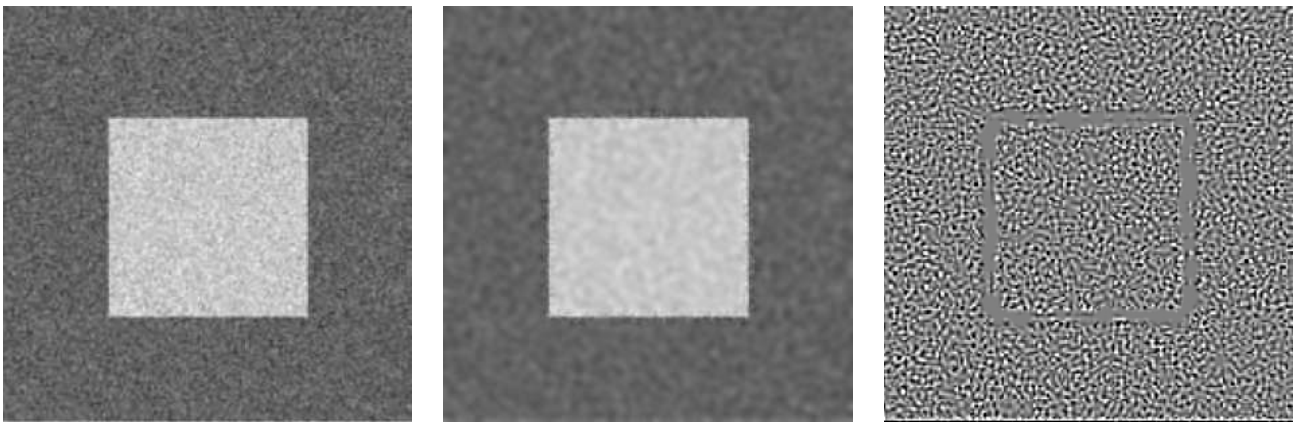

Figure 4: From left to right: original, cartoon, texture. Computed with scale parameter 3. Observe the adhesion problem near the edge.

5.3 Dolphin

This photograph of a logo on a boat (Figure 5) is an almost perfect cartoon! The algorithm does well in keeping it almost entirely in the cartoon part.

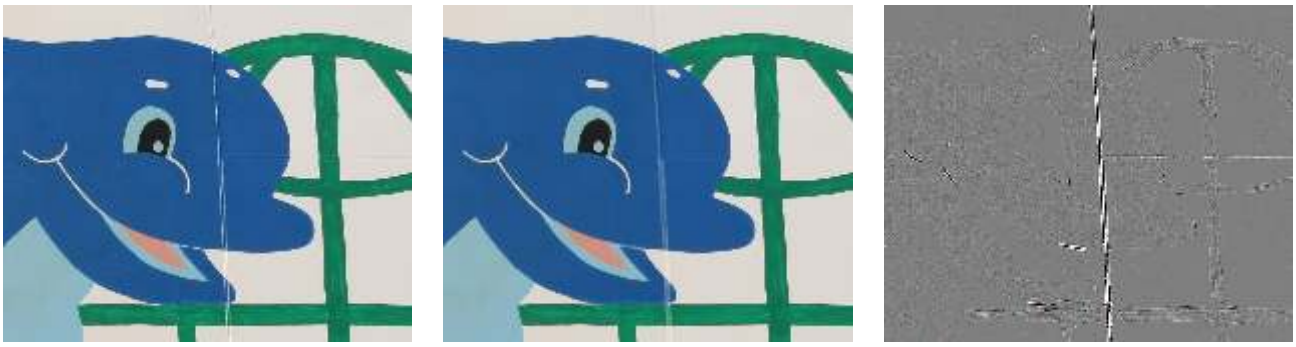

Figure 5: From left to right: original, cartoon, texture. Computed with scale parameter 4. The cartoon image is almost identical to the original.

5.4 Fingerprint

In this fingerprint image (Figure 6) the cartoon part only contains, as expected, a step function indicating the location of the fingerprint.

Figure 6: From left to right: original, cartoon, texture. Computed with scale parameter 2.5. The texture image contains almost all the image details.

5.5 Textured Square

In this image, everything is texture. So the cartoon only contains a smoothed version of the original, and all details move to the textured part (Figure 7).

5.6 Other examples

Figure 8 displays other examples of cartoon+texture decomposition with the proposed algorithm.

Image Credits

Fazen, Djfrank, CC-BY-NC-SA, Flickr[2].

[2] https://www.flickr.com/

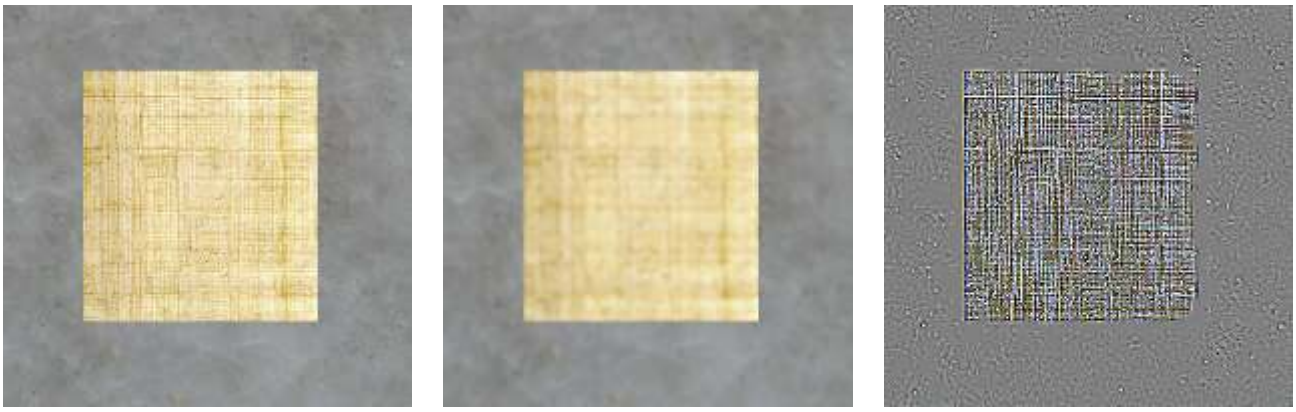

Figure 7: From left to right: original, cartoon, texture. Computed with scale parameter 3. The cartoon image is blurred, while the texture image contains almost everything.

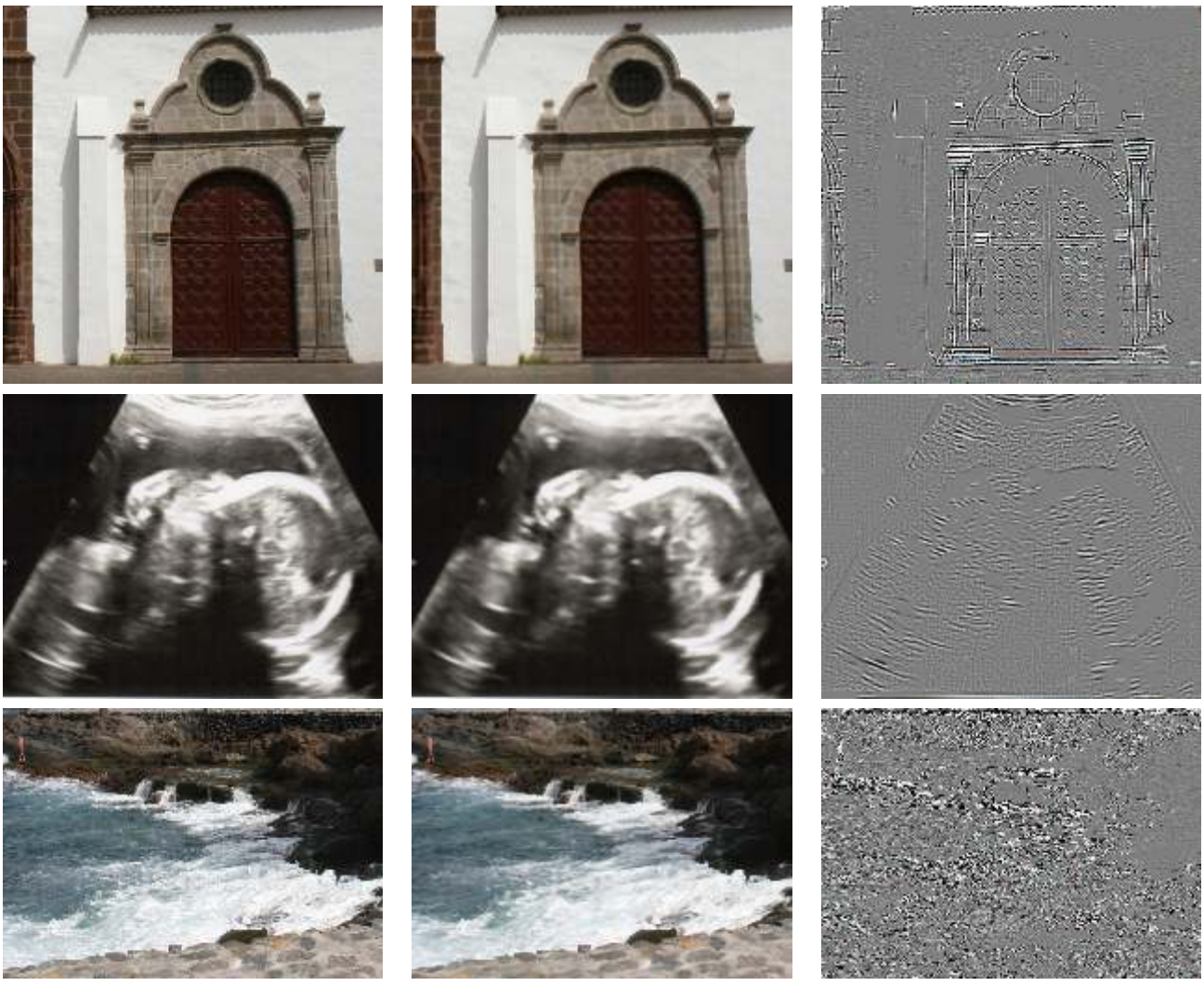

Figure 8: For each row, from left to right: original, cartoon, texture. The scale parameter is 5 in the first two rows and 4 in the last row.

References

[1] Y. Meyer, Oscillating patterns in image processing and nonlinear evolution equations: the fifteenth Dean Jacqueline B. Lewis memorial lectures, American Mathematical Society, 2001.

http://www.worldcat.org/isbn/0821829203.

[2] L.A. Vese and S.J. Osher. Modeling Textures with Total Variation Minimization and Oscillating Patterns, Image Processing. Journal of Scientific Computing,19(1):553-572, 2003. http://dx.doi.org/10.1023/A:1025384832106.

[3] A. Buades, T. Le, J.M. Morel and L. Vese, Fast cartoon + texture image filters, IEEE Transactions on Image Processing, Vol. 19 (18), pp: 1978-1986, 2010. http://dx.doi.org/10.1109/TIP.2010.2046605.