2021/11/21 v0.6 IPOL article class

Published in Image Processing On Line on 2026–06–00.
Submitted on 2024–11–19, accepted on 2026–05–05.
ISSN 2105–1232 © 2026 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.2026.592

Single Image Super-Resolution Based on Anisotropic Diffusion

Pascal Monasse

LIGM, CNRS, Univ Gustave Eiffel, ENPC, Institut Polytechnique de Paris, Marne-la-Vallée, France
(pascal.monasse@enpc.fr)

Communicated by Rémy Abergel and Enric Meinhardt-Llopis *Demo edited by* Pascal Monasse

Abstract

This paper details an unsupervised algorithm for image super-resolution based on a reaction-diffusion model. The diffusion is anisotropic, inspired by Perona-Malik diffusion. The reaction term favors compatibility with the low resolution input image. The implementation is simple and mainly relies on numerical schemes to approximate the required differential operators. The different possibilities present in the literature are analyzed and tested, leading to the choice of the best ones. Qualitative experiments illustrate the results on some images.

Source Code

The ANSI C++ 03 implementation of the code that we provide is the one which has been peer reviewed and accepted by IPOL. The source code, the code documentation, and the online demo are accessible at the IPOL web page of this article[1]. Compilation and usage instructions are included in the `README.md` file of the archive.

Keywords: single image super-resolution; anisotropic diffusion; Perona-Malik

1 Introduction

1.1 Formulation of the Problem

In the field of image processing, many tasks involve getting better quality or going beyond the image acquisition constraints. For instance, denoising and deblurring are widely explored topics with a rich scientific literature. Another problem is the super-resolution, where one tries to generate a higher resolution image from the original image. Typical zoom factors are $\times 2$, where the resolution is multiplied by a factor 2 in each dimension, implying that three quarters of the pixels must be generated. Higher factors, $\times 4$ and beyond are considered extreme, and the resulting image quality cannot be expected to be very good. The ideal situation occurs when additional low resolution

[1] https://doi.org/10.5201/ipol.2026.592

shots of the same scene are available, in which case further information can be transferred in the high resolution image. The problem becomes rather an image fusion one, which comes with its own challenges.

When no additional image is available, the problem is known as single image super-resolution, often abbreviated as SISR. The most natural approach is then to first interpolate the known pixels to the requested resolution and, in a second step, to deblur and possibly to denoise the high resolution image. Though this can be performed independently of the original low-resolution image, better results could be achieved if the down-sampling of the high-resolution image is constrained to be close to the original one. Such a process can be formulated as a reaction-diffusion partial differential equation.

For diffusion, best results are obtained when it is anisotropic: it should happen mostly in the direction perpendicular to the gradient, along the level lines, and a low diffusion should occur in the gradient direction. The isotropic diffusion is driven by the heat equation, where the evolution of the image is proportional to the Laplacian, that is, the trace of the Hessian matrix of the image. However, this trace can also be computed in any other orthonormal basis, the most appropriate one for our purpose being based on an axis along the gradient direction. The diffusion along the gradient direction is weighted as a function of the gradient modulus: the stronger the gradient modulus, the lower the diffusion in its direction. This leads to the celebrated Perona-Malik diffusion [11].

1.2 Anisotropic Diffusion

The simplest filter for diffusion is governed by the heat equation

$$\frac{\partial u}{\partial t}(x,y,t) = \triangle u(x,y,t) = \operatorname{div}(Du) \text{ with } u(.,.,0) = u_0. \tag{1}$$

It admits a closed-form solution, the convolution of u_0 by a centered Gaussian of variance depending on t. For super-resolution, the idea is to first generate the high resolution image by a simple interpolation, then apply diffusion with a reaction term preventing the high-resolution image from diverging too much from the original one. Decomposing the Laplacian into the gradient direction $Du/|Du|$ and its orthogonal yields

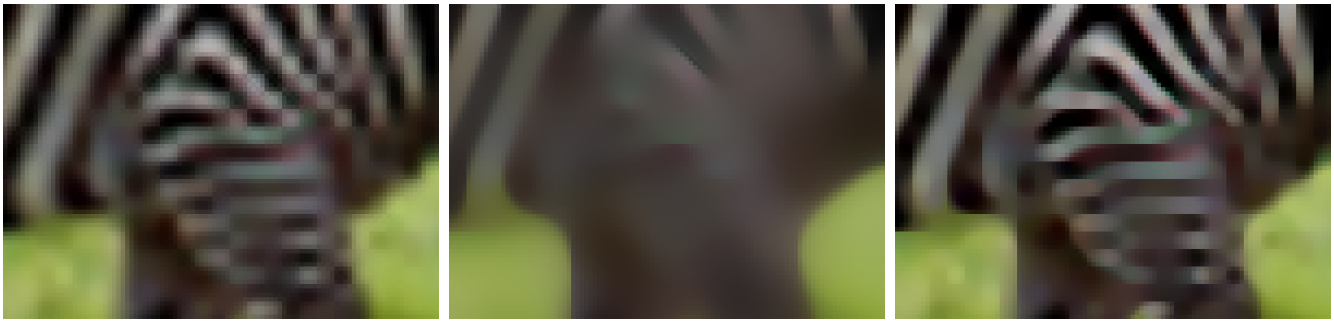
$$\frac{\partial u}{\partial t} = \frac{1}{|Du|^2} D^2u(Du^\top, Du^\top) + \frac{1}{|Du|^2} D^2u(Du, Du). \tag{2}$$

The left term of the sum regularizes the level line whereas the right term interpolates between level lines. Both happening with the same weight leads to isotropic diffusion. The idea of the Perona-Malik filter [11] is to insert a term modulating the diffusion based on the gradient magnitude

$$\frac{\partial u}{\partial t} = \operatorname{div}\left(g(|Du|)Du\right), \tag{3}$$

involving a numeric function g of value 1 around 0 and decreasing to 0. In the first case, that is, when the gradient is low, the diffusion is isotropic, and mostly along the level line for high gradient. Relaxing the first condition on g by taking $g(x) = 1/x$ leads to the mean curvature motion, which only regularizes the level lines without interpolation. More leeway can be obtained by weighing differently the two terms of the right hand side of (2). Figure 1 illustrates the interest of having both a reaction term and anisotropic diffusion.

This article details the implementation of a super-resolution algorithm inspired by Perona-Malik diffusion, with a reaction term ensuring the down-sampling of the image does not deviate too much from the original image. The algorithm is due to Belahmidi and Guichard [4]. We extend it with a numerical analysis of the different schemes in the literature approximating the differential operators.



(a) Isotropic diffusion (b) No reaction term (c) Reaction+anisotropic

Figure 1: With isotropic diffusion (a) or no reaction term (b), the super-resolution $\times 4$ of part of the image *zebra* is of lesser quality than with reaction term and anisotropic diffusion (c).

2 Super-Resolution with Anisotropic Diffusion

2.1 Method

The idea of the method is to apply Perona-Malik diffusion to the scaled version of the input image with a reaction term encouraging fidelity to the initial value inside each scaled pixel. Namely, given an input low-resolution image u_{lr} of size $w \times h$, modelled as an element of $\mathbb{R}^{[0,w-1]\times[0,h-1]}$, we define the upscaled image u_0 obtained by pixel duplication:

$$\forall (x,y) \in \{0,1,\dots,zw-1\} \times \{0,1,\dots,zh-1\},\ u_0(x,y) = u_{\text{lr}}(\lfloor x/z \rfloor, \lfloor y/z \rfloor), \tag{4}$$

where z denotes the scale factor, supposed to be an integer. We then define the projection of an image in the high-resolution space on the set of pixel-wise constant images by:

$$\forall (x,y) \in \{0,1,\dots,zw-1\} \times \{0,1,\dots,zh-1\},\ Pu(x,y) = \frac{1}{z^2} \sum_{i=z\lfloor x/z \rfloor}^{z\lfloor x/z \rfloor + z-1} \sum_{j=z\lfloor y/z \rfloor}^{z\lfloor y/z \rfloor + z-1} u(i,j). \tag{5}$$

In other words, each pixel of u is replaced by the average of its values inside the containing scaled pixel of size $z \times z$.

The evolution equation is written

$$\frac{\partial u}{\partial t} = D^2 u \left(\frac{Du^\top}{|Du|}, \frac{Du^\top}{|Du|} \right) + g(|Du|) D^2 u \left(\frac{Du}{|Du|}, \frac{Du}{|Du|} \right) - (Pu - u_0). \tag{6}$$

Du and D^2u represent the gradient and Hessian matrix of u, $Du^\top$ is the vector Du rotated by an angle $\pi/2$. The function g is decreasing, in order to diffuse less along a strong gradient. We choose a Cauchy function

$$g_\gamma(x) = \frac{1}{1+\gamma x^2}, \tag{7}$$

of parameter $\gamma = 10^{-1}$. Notice that the first term of the sum in (6) can be written

$$D^2 u \left(\frac{Du^\top}{|Du|}, \frac{Du^\top}{|Du|} \right) = |Du| \operatorname{div} \left(\frac{Du}{|Du|} \right), \tag{8}$$

which is the *scaled* curvature. When g is close to 1, that is $|Du|$ close to 0, the diffusion term is the Laplacian of u, $\triangle u$. Hence, when the gradient becomes 0, the diffusion term, which does not make sense, is replaced by $\triangle u$, isotropic diffusion.

The evolution equation (6) is discretized in the following manner

$$u_{n+1} = u_n + \delta t \left(\text{diffusion}(u) - (Pu - u_0)\right), \tag{9}$$

with

$$\text{diffusion}(u) = \begin{cases} \triangle u & \text{if } |Du|^2 < \tau^2, \\ D^2u\left(\frac{Du^\top}{|Du|}, \frac{Du^\top}{|Du|}\right) + g_\gamma(|Du|)D^2u\left(\frac{Du}{|Du|}, \frac{Du}{|Du|}\right) & \text{otherwise.} \end{cases} \tag{10}$$

The derivatives are obtained by finite difference schemes. For a reasonable value of τ, we want to fix it so that $g_\gamma(\tau) = 1 - 10^{-2}$ yielding approximately

$$\tau^2 = 10^{-2}/\gamma. \tag{11}$$

Our choice of $\gamma = 10^{-1}$ leads to $\tau^2 = 10^{-1}$.

2.2 Implementation

The process follows Algorithm 1. When the input is a color image, we apply the algorithm on each of the red, green and blue channels independently. The performance of the implementation and the complexity of the algorithm are discussed in Appendix C.

Algorithm 1: Super-resolution with anisotropic diffusion

Input: Discrete image u_{lr} of size $w \times h$, integer zoom factor z, constant $\gamma = 10^{-1}$, $\delta t = 0.1$, number of iterations n

Output: Super resolution image u of size $zw \times zh$

$u \leftarrow$ Zoom u_{lr} by pixel duplication // pixels of u_0: $\{0, \cdots, zw-1\} \times \{0, \cdots, zh-1\}$

$u_0 \leftarrow u$ // Need to keep initial HR image u for the reaction term

$n \leftarrow \lfloor z^2\,\delta t + 0.5 \rfloor$.

for k=1...n **do**

  $g^2 \leftarrow (\mathcal{F}_{K_{dx}}u)^2 + (\mathcal{F}_{K_{dy}}u)^2$ // (19)

  $d_1 \leftarrow \mathcal{F}_{K_{d1}}u, d_2 \leftarrow \mathcal{F}_{K_{d2}}u$ // (17)–(18) where $g^2 \geq \tau$, (20) otherwise

  Compute Pu // Projection LR (5)

  for $(i,j) \in [0, zw-1] \times [0, zh-1]$ **do**

    $r \leftarrow Pu(i,j) - u_0(i,j)$

    $e_1 \leftarrow d_1(i,j)$

    $e_2 \leftarrow d_2(i,j)/(1 + \gamma\, g^2(i,j))$

    $u(i,j) \leftarrow u(i,j) + \delta t(e_1 + e_2 - r)$

For the numerical approximations of the differential operators, we filter the image u with 3×3 kernels. The coefficients $k_{i,j} \in \mathbb{R}$ of the kernel are denoted as follows

$$K = \begin{pmatrix} k_{-1,-1} & k_{0,-1} & k_{1,-1} \\ k_{-1,0} & k_{0,0} & k_{1,0} \\ k_{-1,1} & k_{0,1} & k_{1,1} \end{pmatrix}. \tag{12}$$

The filtering operation yields an image of the same dimension defined as

$$(\mathcal{F}_K u)_{ij} = \sum_{m,n=-1,0,+1} k_{m,n}\, u_{\widetilde{i+m},\widetilde{j+n}}, \tag{13}$$

where implicit padding is applied at image boundaries:

$$\widetilde{i+m} = \max(0, \min(zw-1, i+m)) \qquad \widetilde{j+n} = \max(0, \min(zh-1, j+n)). \tag{14}$$

The difference with a classical convolution is that for second-order differential operators, the coefficients are variable depending on the position:

$$k_{m,n} : \{0, 1, \ldots, zw-1\} \times \{0, 1, \ldots, zh-1\} \to \mathbb{R}. \tag{15}$$

According to the findings of Appendix A, we compute the first order derivatives as follows

$$K_{dx} = \frac{1}{2(1+2C)} \begin{pmatrix} -C & 0 & C \\ -1 & 0 & 1 \\ -C & 0 & C \end{pmatrix}, \qquad K_{dy} = \frac{1}{2(1+2C)} \begin{pmatrix} -C & -1 & -C \\ 0 & 0 & 0 \\ C & 1 & C \end{pmatrix} \ (C = 0.3) \tag{16}$$

For the second order differential operators, we use the kernels

$$K_{d1}(i,j) = \frac{1}{g^2} \begin{pmatrix} -k_{0,0} + (g^2 - u_x u_y)/2 & 2k_{0,0} - u_y^2 & -k_{0,0} + (g^2 + u_x u_y)/2 \\ 2k_{0,0} - u_x^2 & -4k_{0,0} & 2k_{0,0} - u_x^2 \\ -k_{0,0} + (g^2 + u_x u_y)/2 & 2k_{0,0} - u_y^2 & -k_{0,0} + (g^2 - u_x u_y)/2 \end{pmatrix}, \tag{17}$$

$$K_{d2}(i,j) = \frac{1}{g^2} \begin{pmatrix} -k_{0,0} + (g^2 + u_x u_y)/2 & 2k_{0,0} - u_x^2 & -k_{0,0} + (g^2 - u_x u_y)/2 \\ 2k_{0,0} - u_y^2 & -4k_{0,0} & 2k_{0,0} - u_y^2 \\ -k_{0,0} + (g^2 - u_x u_y)/2 & 2k_{0,0} - u_x^2 & -k_{0,0} + (g^2 + u_x u_y)/2 \end{pmatrix} \text{ with} \tag{18}$$

$$g^2(i,j) = u_x(i,j)^2 + u_y(i,j)^2 \text{ and } k_{0,0} = 0.5\, g^2 - \frac{u_x^2 u_y^2}{u_x^2 + u_y^2}. \tag{19}$$

The expressions for the kernels K_{d1} and K_{d2} in (17) and (18) apply only when $g^2 \geq \tau$, where τ is the threshold defined in (11). When this condition is not satisfied, we resort to isotropic diffusion and use a kernel that approximates the Laplacian

$$K_{d1} = K_{d2} = \frac{1}{4} \begin{pmatrix} 1 & 0 & 1 \\ 0 & -4 & 0 \\ 1 & 0 & 1 \end{pmatrix}. \tag{20}$$

This unusual approximation of the Laplacian is justified in Appendix B.

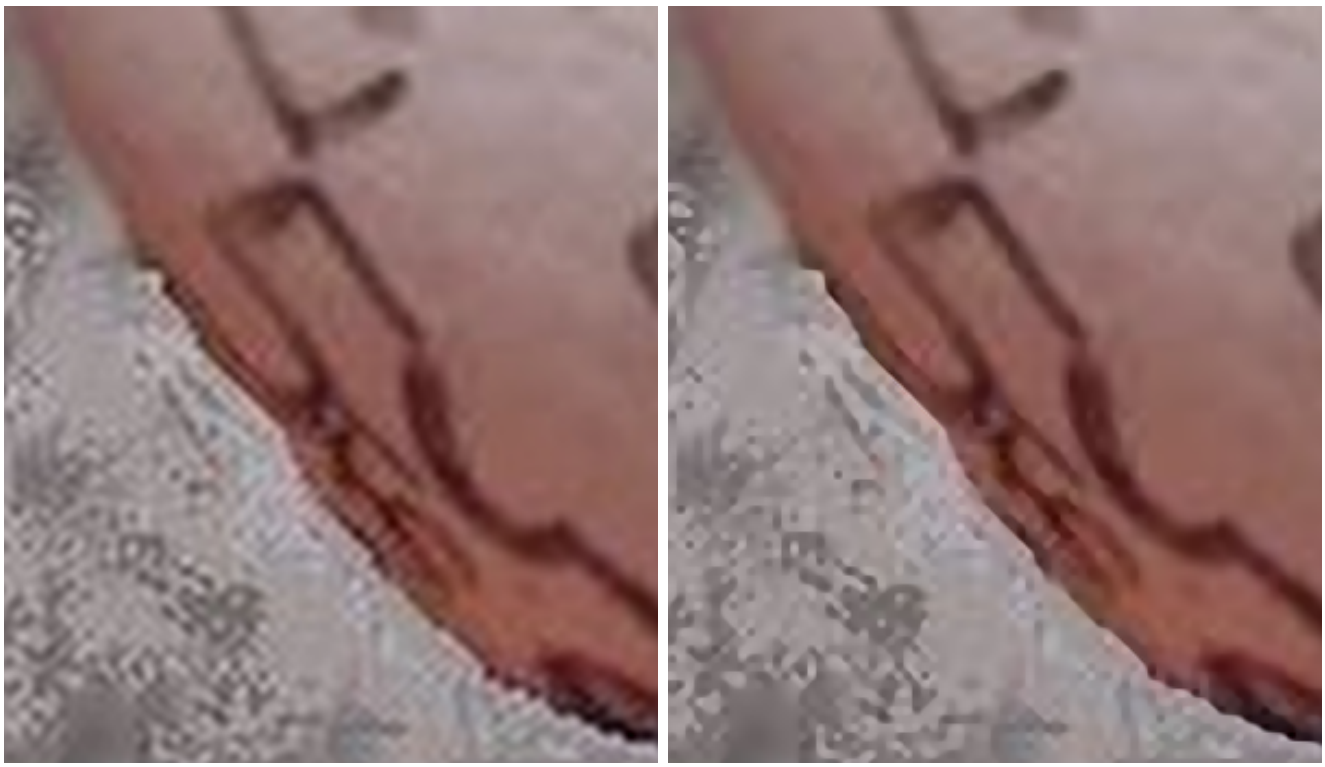
2.3 Experiments

We compare the super-resolution obtained by zero-padding in the Fourier domain with that obtained by diffusion. The former corresponds to an interpolation by cardinal sines. Actually, we approximate it by B-spline interpolation of order 11, as implemented in [5], which is indistinguishable from 0−padding. Figures 2 and 3 show the super-resolution by 0−padding (cardinal sine interpolation) and by diffusion. The latter results in sharper edges. For the Lenna image in Figure 4, the 0−padding exhibits some ringing at the fringe of the hat next to the plumes. This does not happen when diffusion is used. On the negative side, the texture on the nap of the Barbara image in Figure 5 looks fine by 0−padding, whereas some aliasing was introduced by diffusion. In Figures 4 and 5, the input was the scaled down /4 image from the dataset SET14 [13].

The super-resolution of a black and white image in Figure 6 illustrates some defects at high zoom factors. The 0−padding introduced strong ringing due to the high frequencies. The diffusion does not have such problem but exhibits some undulations at the slanted side of the triangle, beginning at ×4. This artifact is quite strong at ×8. It happens that even increasing significantly the number

zoom $\times 4$ by $0-$padding zoom $\times 4$ by diffusion

Figure 2: Super-resolution by $0-$padding and by anisotropic diffusion.



zoom $\times 4$ by $0-$padding zoom $\times 4$ by diffusion

Figure 3: Super-resolution by $0-$padding and by anisotropic diffusion.

zoom $\times 4$ by $0-$padding zoom $\times 4$ by diffusion

Figure 4: Super-resolution by $0-$padding and by anisotropic diffusion.. Notice the ringing at the fringe of the hat, attenuated by the diffusion.

zoom $\times 4$ by $0-$padding zoom $\times 4$ by diffusion

Figure 5: Super-resolution by $0-$padding and by anisotropic diffusion. The diffusion introduced some aliasing that is not prominent in the $0-$padding.

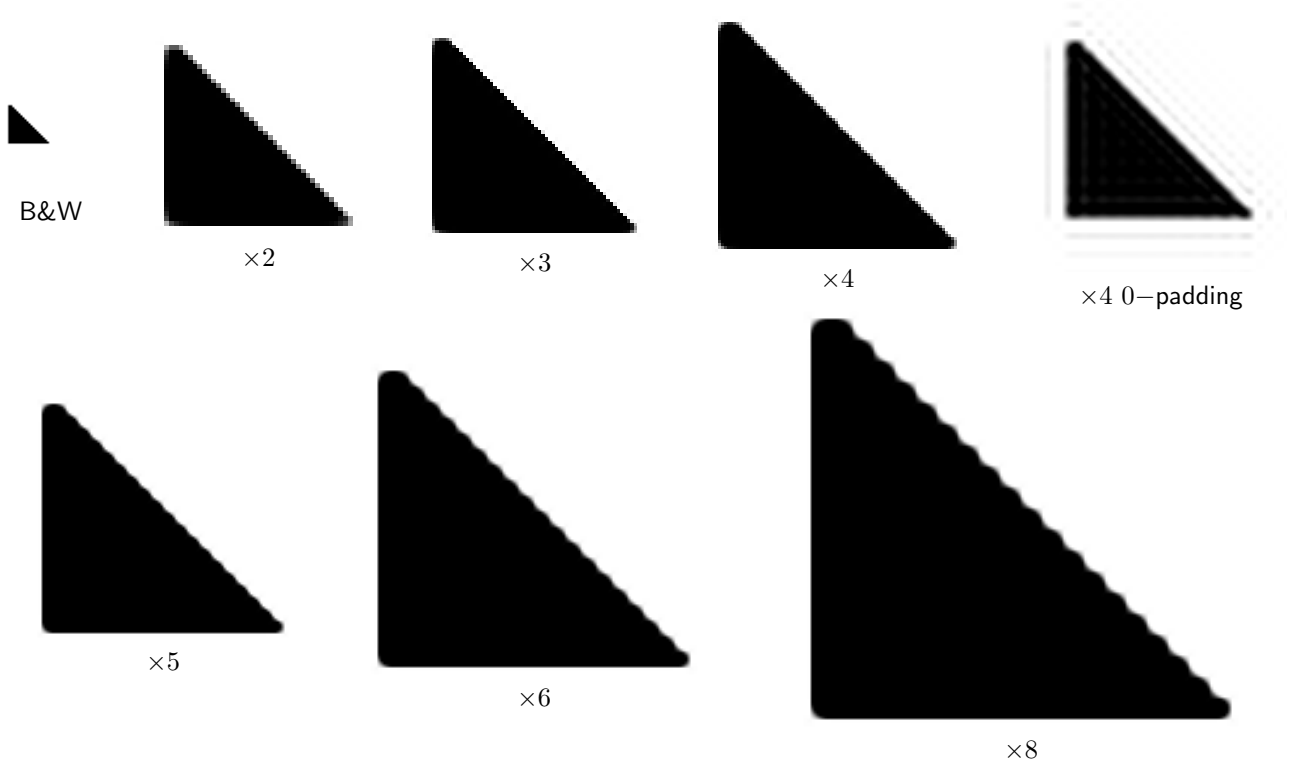


Figure 6: Super-resolution of a black and white image. Notice that $0-$padding introduces conspicuous ringing. The rescaled images by diffusion do not exhibit such artifact, but introduce undulations at the slanted side, noticeable beginning at $\times 4$.

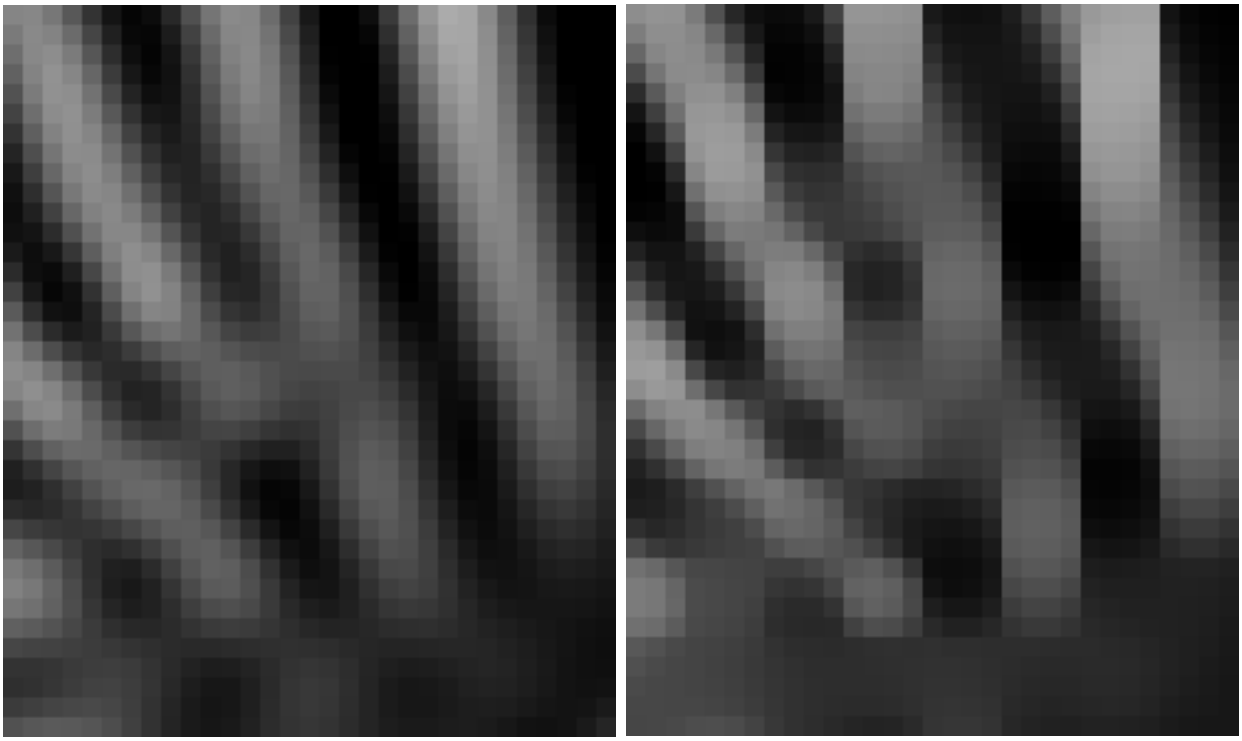
of iterations does not fix the artifact: the diffusion alone would have removed the ripples, but the reaction term prevents such effects by promoting the mean value on blocks of $z \times z$ pixels to be either black or white. For these reasons, the scheme of Alvarez-Morel [3] was selected for all experiments.

The different schemes for the computation of second order derivatives, as summarized in Table 1, are compared in Figure 7. A part of the *zebra* image from SET14 [13] was scaled up $\times 4$ by diffusion and furthermore $\times 8$ by pixel duplication. As shown, the scheme of Alvarez [1] has some pixelation artifacts, whereas Sapiro-Tannenbaum [12] is slightly more blurry. All other schemes yield results almost identical to the scheme of Alvarez-Morel [3].

Figure 8 shows a comparison with other later algorithms. The first one relies on geometric contour stencils [7], which estimate the local contour shapes. The second one is closest to the presented one, diffusing along a direction depending on the structure tensor [8]. On the *ramp* image, both other methods exhibit a conspicuous ringing artifact. However, on a real photograph, stencils yield a slightly better preserved geometry but still noticeable diffusion. The method based on structure tensor has much sharper edges and looks the best.

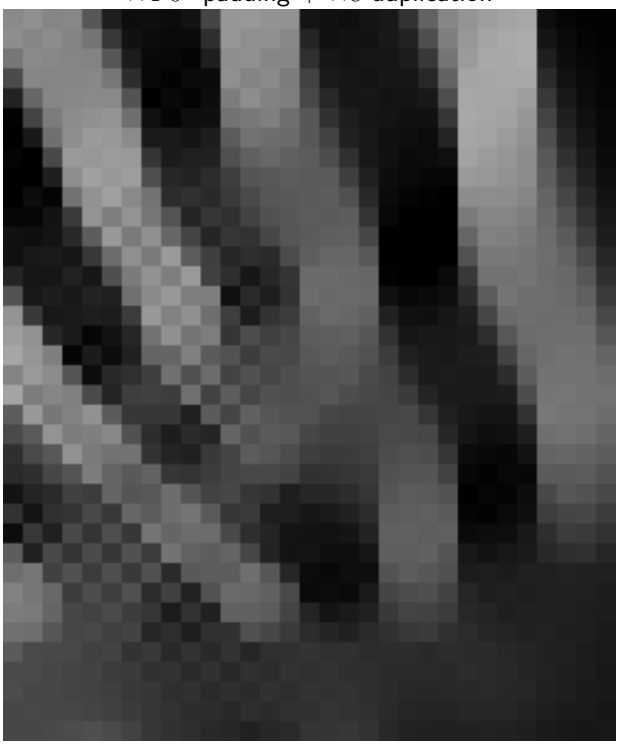
3 Conclusion

Among the unsupervised methods for single image super-resolution, PDE-based methods constitute one category, the other one relies on exemplars taken from the image itself, exploiting the patch redundancy. The latter methods are more costly. Naturally, all have been superseded by supervised methods, which however face a difficulty, the lack of pairs of images of low and high resolution. Such pairs in the training set are synthesized by scaling down the images by some process involving downsampling, blurring, and possibly noise addition. However, the neural network then learns to invert the synthetic procedure, which may be quite different from the image acquisition pipeline, and leads to a difficulty to generalize.

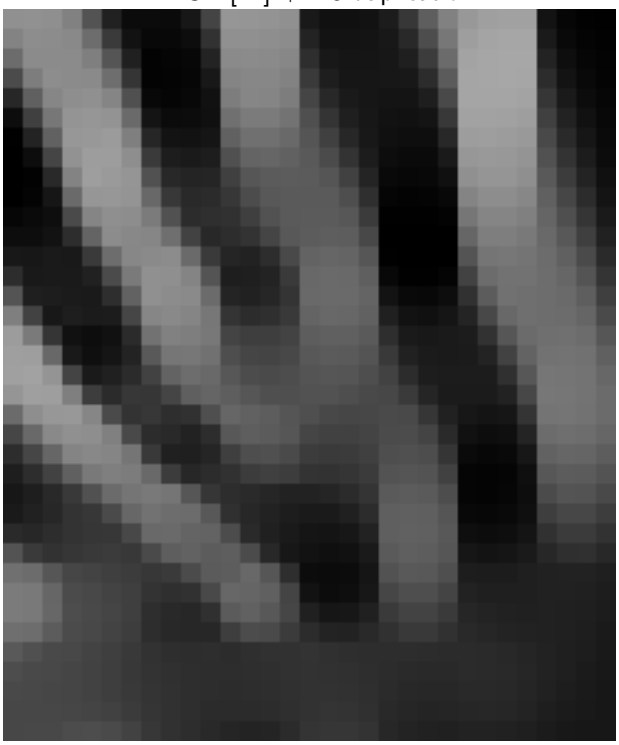


$\times 4$ $0-$padding + $\times 8$ duplication

$\times 4$ ST [12] + $\times 8$ duplication



$\times 4$ Alvarez [1] + $\times 8$ duplication



$\times 4$ AM [3] + $\times 8$ duplication

Figure 7: Comparison of some schemes from second order derivatives in Table 1. An extract for the $\times 4$ *zebra* image is compared to $0-$padding. Each extract is then zoomed in $\times 8$ by pixel duplication for better examination. ST stands for Sapiro-Tannenbaum and AM for Alvarez-Morel. Notice the pixelation artifacts of [1] and the slight additional blur of ST. All other propositions from Table 1, as well as (42), (not shown) look the same as AM [3].

(a) Stencils [7] (b) Structure tensor [8] (c) Ours

Figure 8: Comparison with other methods on $\times 4$ super-resolution, with images *ramp* and *zebra*.

A Numerical Schemes for Differential Operators

To obtain numerical schemes for the derivatives, as is common in image processing, we use convolutions. This choice is natural, since we seek linear and translation covariant operators. We use kernels with minimal support, namely 3×3. For derivatives of any order, a constant signal must yield 0. Hence, using notations from (12)

$$\sum_{ij} k_{ij} = 0. \tag{21}$$

A.1 First Order Derivatives

We concentrate on the x-derivative, the y-derivative can be deduced directly

$$g(x,y) = f(y,x) \quad \Rightarrow \quad \frac{\partial g}{\partial y}(x,y) = \frac{\partial f}{\partial x}(y,x), \tag{22}$$

from which we conclude that the kernels are transposes of each other

$$K_{dy} = K_{dx}^{\top}. \tag{23}$$

As a first constraint, let us notice that

$$f(x,y) = g(y) \quad \Rightarrow \quad \frac{\partial f}{\partial x}(x,y) = 0. \tag{24}$$

We deduce that the sum of coefficients of each row of the kernel vanishes

$$\forall j \in \{-1,0,1\},\ k_{-1,j} + k_{0,j} + k_{1,j} = 0. \tag{25}$$

As a second constraint, the horizontal mirror symmetry should be satisfied for any function f

$$g(x,y) = f(-x,y) \quad \Rightarrow \quad \frac{\partial g}{\partial x}(x,y) = -\frac{\partial f}{\partial x}(-x,y). \tag{26}$$

Obviously, we get

$$\forall j \in \{-1,0,1\},\ k_{-1,j} + k_{1,j} = 0, \tag{27}$$

which together with (25) yields $k_{0,j} = 0$.

The vertical mirror symmetry, written

$$g(x,y) = f(x,-y) \quad \Rightarrow \quad \frac{\partial g}{\partial x}(x,y) = \frac{\partial f}{\partial x}(x,-y), \tag{28}$$

implies $k_{1,-1} = k_{1,1}$.

Now, the derivative of the linear function with vertical level lines

$$f(x,y) = x \quad \Rightarrow \quad \frac{\partial f}{\partial x}(x,y) = 1, \tag{29}$$

permits deducing $4k_{1,1} + 2k_{1,0} = 1$.

There remains a single degree of freedom in the choice of coefficients. Introducing a ratio C, we can choose $k_{1,1} = C\,k_{1,0}$ to get

$$K_{dx} = \frac{1}{2(1+2C)} \begin{pmatrix} -C & 0 & C \\ -1 & 0 & 1 \\ -C & 0 & C \end{pmatrix}, \qquad K_{dy} = \frac{1}{2(1+2C)} \begin{pmatrix} -C & -1 & -C \\ 0 & 0 & 0 \\ C & 1 & C \end{pmatrix}. \tag{30}$$

Taking inspiration from [9], Alvarez et al. [2] chose $k_{1,1} = C\,k_{1,0}$ with

$$C = \frac{\sqrt{2}-1}{2-\sqrt{2}} = \frac{\sqrt{2}-1}{\sqrt{2}(\sqrt{2}-1)} = \frac{1}{\sqrt{2}}. \tag{31}$$

Curiously, the first equality is taken literally in [2], without notice of the simplified form. The reasoning of such choice of C should appear in [9] (full text seemingly absent from internet), but it was not reported in the journal version [10]. Prior work [3] used the value $C = 1/2$, leading to the Sobel filter. The choice $C = 1$ leads to the Prewitt filter.

A.2 Second Order Derivatives

Where $|Du| \neq 0$, we define functions f_0 and f_1 such that

$$|Du|\ \mathrm{div}\left(\frac{1}{|Du|}Du\right) = \frac{1}{|Du|^2}(u_y^2 u_{xx} - 2u_x u_y u_{xy} + u_x^2 u_{yy}) = \frac{1}{|Du|^2} f_0(Du, D^2u), \tag{32}$$

$$\frac{1}{|Du|^2} D^2u(Du,Du) = \frac{1}{|Du|^2}(u_x^2 u_{xx} + 2u_x u_y u_{xy} + u_y^2 u_{yy}) = \frac{1}{|Du|^2} f_1(Du, D^2u). \tag{33}$$

We can write, ignoring terms of degree 3 or higher in h and l

$$u(s_1 h, s_2 l) = u(0,0) + s_1 h u_x + s_2 l u_y + \frac{1}{2}\left(h^2 u_{xx} + 2s_1 s_2 h l u_{xy} + l^2 u_{yy}\right), \tag{34}$$

with $s_1, s_2 \in \{-1,0,1\}$, all derivatives being evaluated at $(0,0)$. We take $h = l = 1$ to get

$$u(s_1,s_2) = u(0,0) + s_1 u_x + s_2 u_y + \frac{1}{2}\left(s_1^2 u_{xx} + 2s_1 s_2 u_{xy} + s_2^2 u_{yy}\right). \tag{35}$$

If $v(x,y) = u(-x,-y)$, we have $f_i(Dv(x,y), D^2v(x,y)) = f_i(Du(-x,-y), D^2u(-x,-y))$ for $i \in \{0,1\}$. Due to these symmetries we can parameterize

$$K_f = \begin{pmatrix} k_{1,1} & k_{0,1} & k_{1,-1} \\ k_{1,0} & -4k_{0,0} & k_{1,0} \\ k_{1,-1} & k_{0,1} & k_{1,1} \end{pmatrix}. \tag{36}$$

Using the approximation (35), we get

$$K_f \cdot u = \sum_{i,j} k_{i,j} u_{i,j} = u_{0,0}(-4k_{0,0} + 2k_{1,0} + 2k_{0,1} + 2k_{1,1} + 2k_{1,-1}) + u_{xx}(k_{1,-1} + k_{0,1} + k_{1,1}) + u_{yy}(k_{1,-1} + k_{1,0} + k_{1,1}) + 2u_{xy}(k_{1,1} - k_{1,-1}). \quad (37)$$

Identification term by term with (32) yields a linear system of four equations with five unknowns

$$\begin{cases} -4k_{0,0} + 2k_{1,0} + 2k_{0,1} + 2k_{1,1} + 2k_{1,1} = 0 \\ k_{1,-1} + k_{0,1} + k_{1,1} = u_y^2 \\ k_{1,-1} + k_{1,0} + k_{1,1} = u_x^2 \\ k_{1,1} - k_{1,-1} = -u_x u_y \end{cases} \Rightarrow \begin{cases} k_{1,0} = 2k_{0,0} - u_y^2 \\ k_{0,1} = 2k_{0,0} - u_x^2 \\ k_{1,-1} = -k_{0,0} + (u_x^2 + u_y^2 + u_x u_y)/2 \\ k_{1,1} = -k_{0,0} + (u_x^2 + u_y^2 - u_x u_y)/2 \end{cases} \quad (\text{for } f_0). \quad (38)$$

The same identification with (33) yields

$$\begin{cases} -4k_{0,0} + 2k_{1,0} + 2k_{0,1} + 2k_{1,1} + 2k_{1,1} = 0 \\ k_{1,-1} + k_{0,1} + k_{1,1} = u_x^2 \\ k_{1,-1} + k_{1,0} + k_{1,1} = u_y^2 \\ k_{1,1} - k_{1,-1} = u_x u_y \end{cases} \Rightarrow \begin{cases} k_{1,0} = 2k_{0,0} - u_x^2 \\ k_{0,1} = 2k_{0,0} - u_y^2 \\ k_{1,-1} = -k_{0,0} + (u_x^2 + u_y^2 - u_x u_y)/2 \\ k_{1,1} = -k_{0,0} + (u_x^2 + u_y^2 + u_x u_y)/2 \end{cases} \quad (\text{for } f_1). \quad (39)$$

It is easy to check that unless $u_x = \pm u_y$, the four parameters cannot be simultaneously non-negative, a necessary condition for the stability of the numerical scheme. Several choices of $k_{0,0}$ were proposed in the literature to minimize, with different criteria, the incompatibility of these constraints.

Notice that if $v(x,y) = u(y,x)$, we have $f_i(Dv(x,y), D^2v(x,y)) = f_i(Du(y,x), D^2u(y,x))$, which indicates that K_f must be symmetric, hence $k_{0,1} = k_{1,0}$. Again, this is not possible in general. The choice of Alvarez [1], $k_{0,0} = 0.25\,(u_x^2 + u_y^2)$, instead, sets $k_{0,1} = -k_{1,0}$. Various suggestions appear in the literature, see Table 1. Notice that [12] has no direct scheme for f_0 and f_1 but uses rather estimates of second order derivatives, which amounts to take $k_{0,0} = 0.5\,(u_x^2 + u_y^2)$. Indeed, in this case

$$K_{f0} = \begin{pmatrix} -0.5\,u_x u_y & u_x^2 & 0.5\,u_x u_y \\ u_y^2 & -2(u_x^2 + u_y^2) & u_y^2 \\ 0.5\,u_x u_y & u_x^2 & -0.5\,u_x u_y \end{pmatrix} = u_y^2 \begin{pmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{pmatrix} - 2u_x u_y \begin{pmatrix} -0.5 \\ 0 \\ 0.5 \end{pmatrix} \begin{pmatrix} -0.5 & 0 & 0.5 \end{pmatrix} + u_x^2 \begin{pmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{pmatrix}. \quad (40)$$

We identify in the three matrices standard difference schemes for u_{xx}, u_{xy} and u_{yy}.

Article	Sapiro-Tannenbaum [12]	Alvarez-Morel [3]	Cohignac et al. [6]	Alvarez [1]
$k_{0,0}$	$0.5\,(u_x^2 + u_y^2)$	$0.5\,(u_x^2 + u_y^2) - \frac{u_x^2 u_y^2}{u_x^2 + u_y^2}$	$\frac{u_x^2 + u_y^2 - \lvert u_x u_y \rvert + \max(u_x^2, u_y^2)}{4}$	$0.25\,(u_x^2 + u_y^2)$
$m \pm \sigma(\times 10^{-2})$	-8.60 ± 6.39	-8.67 ± 6.77	-8.54 ± 6.09	-8.55 ± 6.12

Table 1: Different choices for the free parameter $k_{0,0}$ found in the literature and the statistics (average $\pm$ standard deviation) of scaled mean curvature (32), for $C = 0.3$ in the gradient computation.

A.3 Numerical Simulations

To test the different choices for the constant C as in (30), we generate an image along the following protocol:

1. In a floating point image of size $zw \times zw$, put at value 0 the pixels inside the disk of radius zr centered at (zw, zw), at value 1 all other pixels. z is a zoom factor.

2. Convolve with Gaussian signal of standard deviation $\sigma = 0.8z$ (truncated at $\pm 3\sigma$) in x then y directions.

3. Subsample by the factor z to get an image of size $w \times w$.

This represents a sampled disk of radius r placed in the middle of the image. In practice, we take $r = 10$ and $w = 2r + 10 = 30$. For all angles $\theta_i = 2i\pi/l$, with perimeter $l = \lceil 2\pi r \rceil$, $i = 0, \dots l-1$, take the 3×3 patch centered at $([w/2 + r\cos\theta_i], [w/2 + r\sin\theta_i])$, with operator $[x]$ the closest integer to value x, and filter the image using kernels for x and y derivatives, giving a vector $g_i = (gx_i, gy_i)$. We compare this vector with the ground truth proportional to $v_i = (\cos\theta_i, \sin\theta_i)$ yielding an angular error

$$\alpha_i = \cos^{-1}\left(\frac{1}{\|g_i\|} g_i \cdot v_i\right). \tag{41}$$

We then compute the mean m and standard deviation σ of the α_i. The ground truth modulus of g_i is not available but we can compute the standard deviation of the $\|g_i\|$.

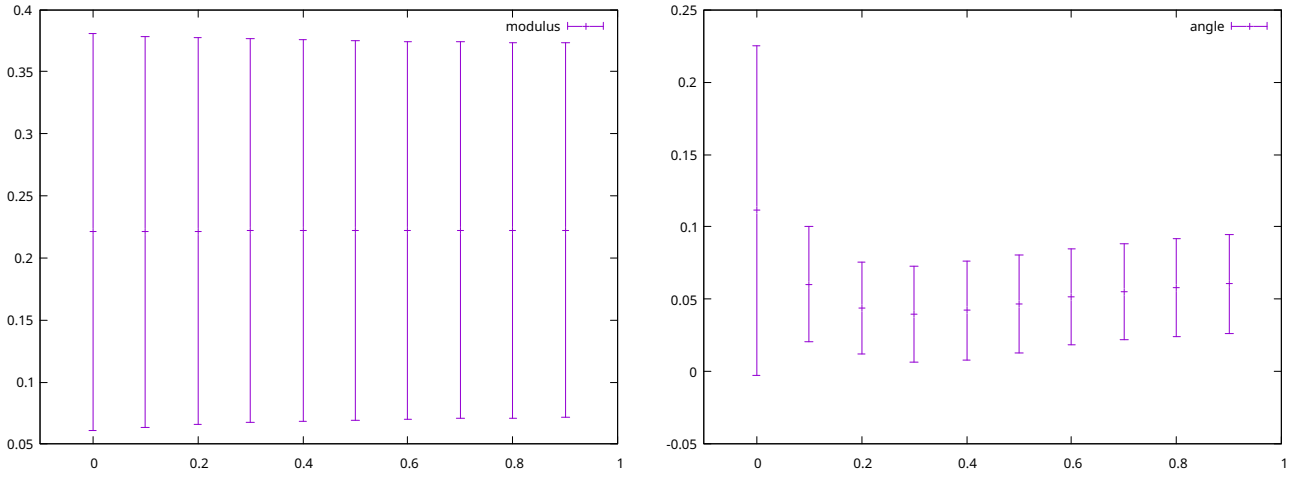


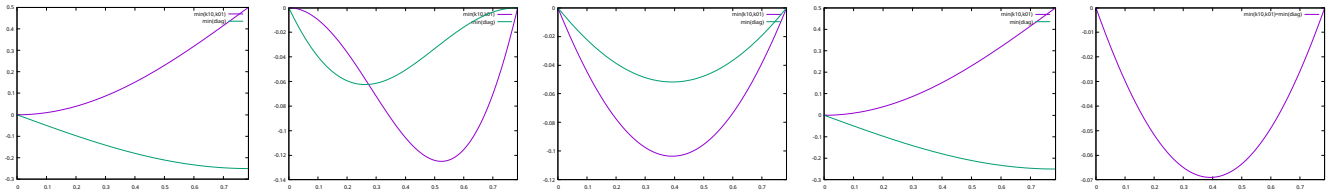
Figure 9: Errors (average $\pm$ standard deviation) for modulus and angle in computing the gradient along a circle, using a 3×3 kernel, as a function of the factor C such that $k_{1,1} = C\,k_{1,0}$. The sub-sampling factor is $z = 2$.

Figure 9 show the error plots $m \pm \sigma$ depending on the value of C in the range $[0, 0.9]$. The modulus is almost insensitive to the value of C, whereas the minimum error of 2.3° for the angle is at $C = 0.3$. The commonly used value (31) of $C = 1/\sqrt{2} \sim 0.7$ has a worse angular error of 3.2° with sensibly identical standard deviation. This has been obtained with a sub-sampling factor $z = 2$. At this level, the sub-sampled image is still strongly aliased. The experience remains interesting because it shows how the estimation is sensitive to aliasing. Increasing the factor, for instance $z = 20$, yields a well-sampled image. Although errors decrease for all values of C, the same tendency is observed: values of C around 0.3 keep getting lower error than others.

Table 1 includes the statistics of the scaled curvature (32) along the circle. It is apparent that all formulas for $k_{0,0}$ have a standard deviation as high as the average, revealing a strong lack of isotropy of the numerical scheme.

We can formulate another proposition for the choice of $k_{0,0}$. In (38) and (39), we have

$$\min(k_{1,0}, k_{0,1}) = 2k_{0,0} - \max(u_x^2, u_y^2) \qquad \min(k_{1,-1}, k_{1,1}) = -k_{0,0} + (u_x^2 + u_y^2 - |u_x u_y|)/2. \tag{42}$$



$k_{0,0} = 0.5$ [12] Alvarez-Morel [3] Cohignac et al. [6] Alvarez [1] Ours (43)

Figure 10: Minimum coefficients in scaled curvature for the different choices of $k_{0,0}$: $min(k_{1,0}, k_{0,1})$ and minimum of diagonal coefficients, see (42). They are plotted as a function of $\theta \in [0, \pi/4]$, writing $u_x = \cos\theta$, $u_y = \sin\theta$ for unit norm gradient.

The first term is an increasing function of $k_{0,0}$, whereas the second is decreasing. To maximize the minimum of all four values, we put the last two equal, leading to

$$k_{0,0} = (u_x^2 + u_y^2 - |u_x u_y| + 2\max(u_x^2, u_y^2))/6. \tag{43}$$

In Figure 10, the symmetric role played by u_x and u_y lets us restrict the study to $u_x \geq u_y \geq 0$ and we set $u_x^2 + u_y^2 = 1$. Numeric tests with (43) yield a scaled mean curvature along the circle of $(-8.56 \pm 6.11) \times 10^{-2}$, which is on par with other measures in Table 1, even though the coefficients reach the least negative value among all choices of $k_{0,0}$.

It appears that the scheme of Alvarez [1] has the advantage of simplicity with a second best (lowest) standard deviation. However, as shown in Figure 7 it provokes some pixelation artifacts. The scheme of Sapiro-Tannenbaum [12] is a bit more blurry than other ones. All other schemes look the same as its close variant Alvarez-Morel [3].

B Approximation for the Laplacian operator

A usual kernel for the Laplacian operator is the following

$$K_{d1} = K_{d2} = \frac{1}{4(1+\frac{1}{\sqrt{2}})} \begin{pmatrix} \frac{1}{\sqrt{2}} & 1 & \frac{1}{\sqrt{2}} \\ 1 & -4(1+\frac{1}{\sqrt{2}}) & 1 \\ \frac{1}{\sqrt{2}} & 1 & \frac{1}{\sqrt{2}} \end{pmatrix}. \tag{44}$$

Terms in $\sqrt{2}$ are related to the distance of diagonal neighbors. In our case, we use

$$\frac{1}{4} \begin{pmatrix} 1 & 0 & 1 \\ 0 & -4 & 0 \\ 1 & 0 & 1 \end{pmatrix}. \tag{45}$$

The reason is that when the (squared) gradient norm goes to 0, the diffusion should be Laplacian, and we set $d1$ and $d2$ to half the Laplacian. Summing the expressions (17) and (18) leads to

$$K_{d1} + K_{d2} = \frac{1}{g^2} \begin{pmatrix} g^2 - 2k_{0,0} & 4k_{0,0} - g^2 & g^2 - 2k_{0,0} \\ 4k_{0,0} - g^2 & -8k_{0,0} & 4k_{0,0} - g^2 \\ g^2 - 2k_{0,0} & 4k_{0,0} - g^2 & g^2 - 2k_{0,0} \end{pmatrix}. \tag{46}$$

All depends on the limit of $k_{0,0}/g^2$ when g^2 goes to zero. Such limit does not exist for all choices of $k_{0,0}$. However, if we impose $|u_x| = |u_y|$, $k_{0,0}/g^2$ is a constant, equal to 0.5 for Sapiro-Tannenbaum [12] and 0.25 for all other choices. We commit to the latter and divide by 2 the terms of (46), leading to (45). In practice, we noticed no visual difference in the results with both formulas, so we use (45), which requires fewer floating point operations due to the four 0 coefficients.

C Performance of Implementation and Complexity

The algorithm implementation closely follows Algorithm 1. Each filter has complexity $O(z^2N)$ with N the number of pixels and z the zoom factor. This is the complexity of one iteration and, since by default the number of iterations is proportional to z^2, the final complexity is $O(z^4N)$. The number of iterations was determined empirically. Notice that the term z^2 is the size of a "zoomed" pixel.

The algorithm relying mainly on repeated filtering of the image, it is embarrassingly parallel. It is implemented with minimal effort with OpenMP: all image row filters are distributed to a group of threads. By default, OpenMP creates as many threads as there are CPU cores on the machine. However, this may not be always optimal. For instance, Intel processors for laptop have P-cores and E-cores. The former are for performance and the latter for energy efficiency. When part of the workload is attributed to E-cores, the P-cores finish their workload before and remain idle until E-cores finish before the program can proceed. This is illustrated in Figure 11: on the laptop computer, performance deteriorates at 7 cores (it has 6 P-cores), before it starts improving again because each E-core has less workload. On the desktop machine, having 10 CPU and 20 cores, the effect does not occur.

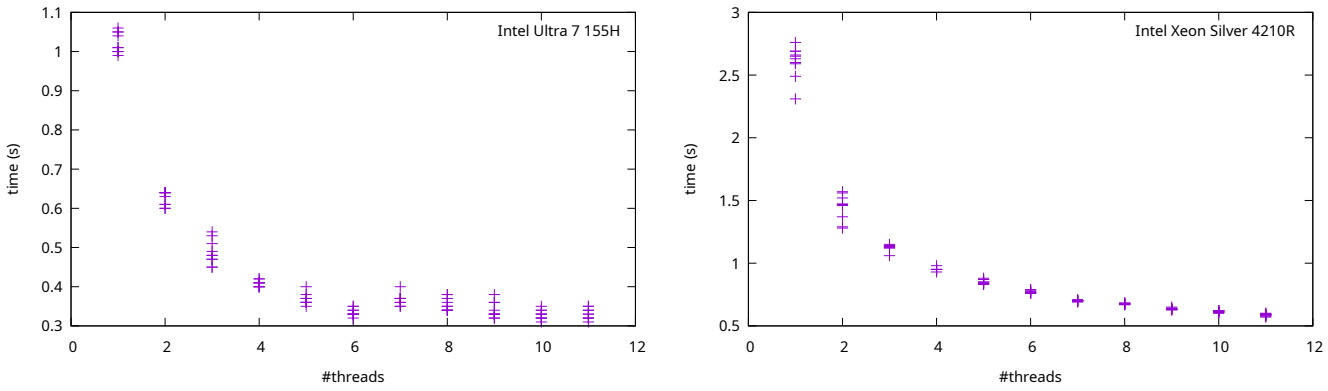


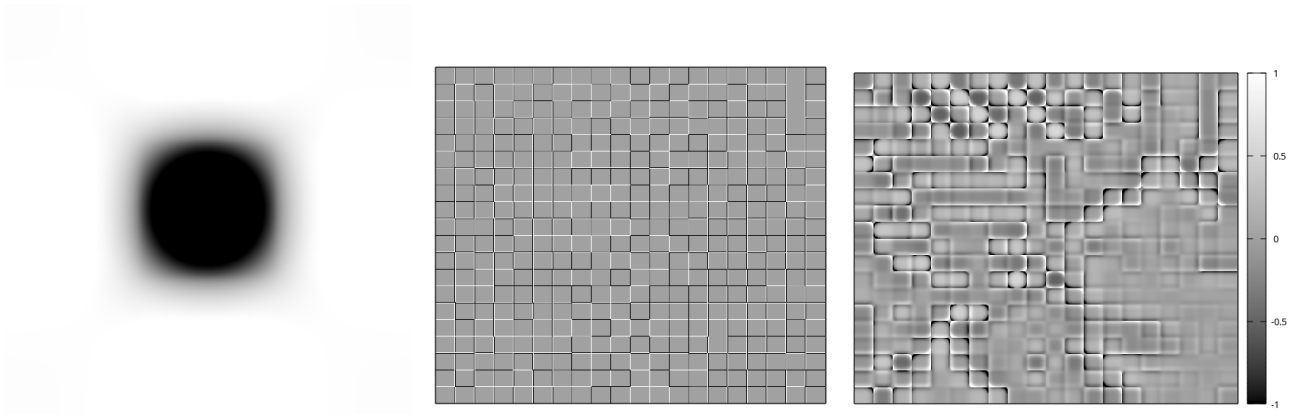
Figure 11: Elapsed time for a zoom $\times 4$ of the three channels of image *zebra*. Left: on a laptop. Right: on a desktop machine.

D Infinitesimal Generator

The evolution equation is written

$$\frac{\partial u}{\partial t} = f(Du, D^2u). \tag{47}$$

The impulse response of the evolution can be simulated easily: take a 3×3 white image with a black dot at the middle, and apply the super-resolution by a factor $z = 128$ (Figure 12). The result is between a rounded square and a disk. The infinitesimal generator is the function f driving the evolution. At early iterations, diffusion happens only at zoomed pixel boundaries, while it spreads inside these later.



(a) Impulse response (b) Inf. generator (it. 1) (c) Inf. generator (it. 100)

Figure 12: (a) "Impulse response": zoom $\times 128$ of a black dot on white background. (b) Infinitesimal generator at iteration 1 of part of image *zebra* with factor $\times 32$. (c) At iteration 100.

Acknowledgments

The author would like to thank Abdelmounim Belahmidi for initial research on the subject and providing his source code for inspiration. Jean-Michel Morel and Luis Alvarez provided useful feedback on the first version of the paper. Finally, the reviewers' reports were instrumental in motivating Appendix C and D and helped improve the code and the present manuscript.

Image Credits

 F. Mira, CC-BY-SA-2.0[2]

 Kalan, CC-BY-SA-2.5[3]

 Classical *Lenna* image from SET14 [13].

 Classical *Barbara* image from SET14 [13].

 *ramp*, P. Monasse, CC-BY-SA-4.0

 *zebra* from SET14 [13].

The SET14 dataset can be downloaded from https://github.com/jbhuang0604/SelfExSR/tree/master/data.

References

[1] L. Alvarez, *Images and PDE's*, in International Conference on Analysis and Optimization of Systems Images, Wavelets and PDEs (ICAOS), Springer, 1996, pp. 3–14. https://doi.org/10.1007/3-540-76076-8_112.

[2] https://commons.wikimedia.org/wiki/File:Green_(3028716998).jpg
[3] https://commons.wikimedia.org/wiki/File:WikiEasterEgg-2.jpg

[2] L. Alvarez, P.-L. Lions, and J.-M. Morel, *Image selective smoothing and edge detection by nonlinear diffusion. II*, SIAM Journal on Numerical Analysis, 29 (1992), pp. 845–866. https://doi.org/10.1137/0729052.

[3] L. Alvarez and J. Morel, *Formalization and computational aspects of image analysis*, Acta Numerica, 3 (1994), pp. 1–59. https://doi.org/10.1017/S0962492900002415.

[4] A. Belahmidi and F. Guichard, *A partial differential equation approach to image zoom*, in International Conference on Image Processing (ICIP), vol. 1, IEEE, 2004, pp. 649–652. https://doi.org/10.1109/ICIP.2004.1418838.

[5] T. Briand and P. Monasse, *Theory and Practice of Image B-Spline Interpolation*, Image Processing On Line, 8 (2018), pp. 99–141. https://doi.org/10.5201/ipol.2018.221.

[6] T. Cohignac, F. Eve, F. Guichard, C. Lopez, and J. Morel, *Affine morphological scale space: Numerical analysis of its fundamental equation*, tech. report, Ceremade, Université Paris Dauphine, 1993.

[7] P. Getreuer, *Image interpolation with geometric contour stencils*, Image Processing On Line, 1 (2011), pp. 98–116. https://doi.org/10.5201/ipol.2011.g_igcs.

[8] ——, *Roussos-Maragos tensor-driven diffusion for image interpolation*, Image Processing On Line, 1 (2011), pp. 178–186. https://doi.org/10.5201/ipol.2011.g_rmdi.

[9] M. Nitzberg and T. Shiota, *Nonlinear image smoothing with edge and corner enhancement*, tech. report, Harvard University, Division of Applied Sciences, 1990. https://doi.org/10.1109/34.149593.

[10] ——, *Nonlinear image filtering with edge and corner enhancement*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 14 (1992), pp. 826–833. https://doi.org/10.1109/34.149593.

[11] P. Perona and J. Malik, *Scale-space and edge detection using anisotropic diffusion*, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 12 (1990), pp. 629–639. https://doi.org/10.1109/34.56205.

[12] G. Sapiro and A. Tannenbaum, *Affine invariant scale-space*, International Journal of Computer Vision, 11 (1993), pp. 25–44. https://doi.org/10.1007/BF01420591.

[13] R. Zeyde, M. Elad, and M. Protter, *On single image scale-up using sparse-representations*, in Curves and Surfaces, Springer Berlin Heidelberg, 2012, pp. 711–730. https://doi.org/10.1007/978-3-642-27413-8_47.