Fast Edge-Aware Processing via First Order Proximal Approximation

Hicham Badri, Hussein Yahia, Driss Aboutajdine

Fast Edge-Aware Processing via First Order Proximal Approximation

Hicham Badri, *Student Member*, Hussein Yahia and Driss Aboutajdine, *Senior Member*

Abstract—We present a new framework for fast edge-aware processing of images and videos. The proposed smoothing method is based on an optimization formulation with a non-convex sparse regularization for a better smoothing behavior near strong edges. We develop mathematical tools based on first order approximation of proximal operators to accelerate the proposed method while maintaining high-quality smoothing. The first order approximation is used to estimate a solution of the proximal form in a half-quadratic solver, and also to derive a warm-start solution that can be calculated quickly when the image is loaded by the user. We extend the method to large-scale processing by estimating the smoothing operation with independent 1D convolution operations. This approach linearly scales to the size of the image and can fully take advantage of parallel processing. The method supports full color filtering and turns out to be temporally coherent for fast video processing. We demonstrate the performance of the proposed method on various applications including image smoothing, detail manipulation, HDR tone-mapping, fast edge simplification and video edge-aware processing.

Index Terms—Fast image smoothing, sparsity, edge-aware processing.

1 Introduction

During the past few years, there has been a significant amount of work on edge-aware filtering. Unlike regular Gaussian smoothing, edge-aware filters blur the image while preserving sharp edges. Probably the most popular edge-aware filter is the bilateral filter [1] that performs a weighted averaging of the pixel values in a window based on both space and range distances. The bilateral filter can be seen as a high-dimensional filter working in a 5D space when performed on 2D RGB images [2]. A naive implementation of this filter is computationally demanding as it operates in a high-dimension space. Many researchers tried to boost this filter or at least simulate bilateral-like results by either using linear interpolation [3], optimized data structures such as the bilateral grid [4], reformulation and downsampling [5], [6], constant time spatial filters decomposition [7], Gaussian KD-trees [8], Supports Vector Machines regression [9], recursive implementation [10], dimensionality reduction [11], adaptive manifolds [12], among others. The bilateral filter is the building block of a wide range of applications such as HDR tone-mapping [3], non-photorealistic rendering [13], upsampling [14] and non-blind image deconvolution [15]. Unfortunately, bilateral filtering suffers from several issues. For instance, it tends to produce several halo artifacts in detail manipulation applications as pointed out by Farbman *et al.* [16]. It also tends to blur some edges and refuses to wash-out some large-scale details as can be seen in Figure 1. The main problem of the bilateral filter comes from its smoothing behavior that is controlled by two parameters σ_s and σ_r, called respectively the scale and range parameters. As σ_s increases, the bilateral filter acts like a range filter and as σ_r increases the bilateral filter becomes a Gaussian filter. Increasing σ_s tends to preserve sharp edges but fails at smoothing small-scale details. On the other hand, increasing σ_r tends to smooth small scale details but over-smooths sharp edges. As a result, the bilateral filter may not be suitable for some edge-aware manipulation applications due to its smoothing behavior. New local filtering methods perform filtering in a pyramid to prevent halo artifacts. The local laplacian filter method [17], [18] uses a Laplacian pyramid while the mixed-domain method [19] uses a Gaussian pyramid. While these methods produce high-quality smoothing results compared to previous local filtering methods, they require a relatively large processing time.

Gradient-domain methods have emerged as another set of methods for edge-aware manipulation. These methods are based on optimization, contrary to local filtering methods such as the bilateral filter. The first method introduced in this category is the Total Variation (TV) method [20]. This approach consists in minimizing an energy regularized by a convex gradient function (l_1-norm). The method was mainly used for denoising and its use for computer graphics applications was limited. The method of Farbman *et*

- *H. Badris PhD is funded by an INRIA (Direction of Research) CORDI-S grant. He is making a PhD in co-supervision with INRIA Bordeaux Sud-Ouest and Mohammed V-Agdal University - LRIT, Associated Unit to CNRST (URAC 29) - Rabat, Morocco. E-mail: hicham.badri@inria.fr.*
- *H. Yahia is with INRIA Bordeaux Sud-Ouest. E-mail: hussein.yahia@inria.fr.*
- *D. Aboutajdine is with Mohammed V-Agdal University - LRIT, Associated Unit to CNRST (URAC 29). E-mail: aboutaj@ieee.org.*

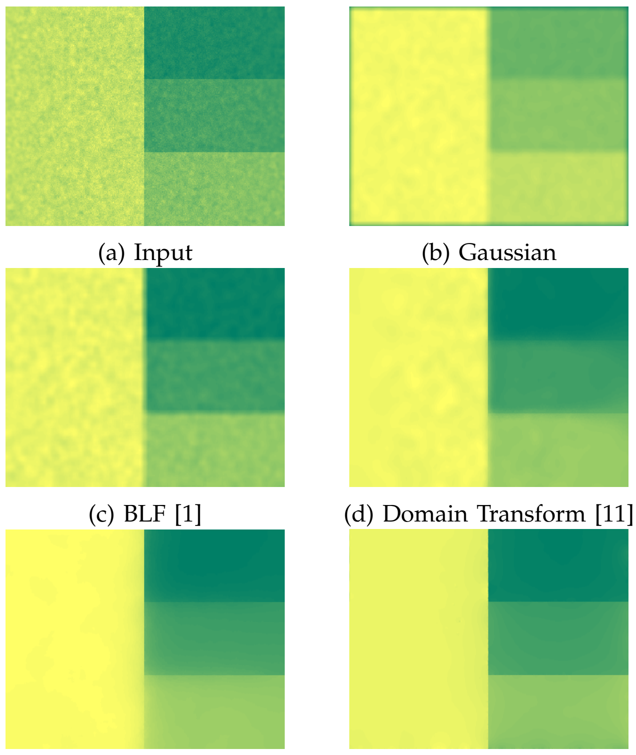


Fig. 1: Image smoothing of the noisy input (a) with various edge-aware filters (image from [16]). Local filtering methods such as the bilateral filter and the domain transform are not able to correctly smooth large-scale details and tend to blur sharp edges. In contrast, our method smooths both small and large details while being computationally efficient.

al. [16] is also based on a convex optimization formulation but instead of the l_1-norm, the method makes use of a weighted least squares formulation (WLS). The approach has shown to produce improved smoothing results and was successfully used in detail manipulation applications. Xu *et al.* make use of the l_0-minimization framework [21] to progressively suppress details. The l_0 quasi-norm is a non-convex and non-differentiable function that highly promotes sparsity. The method produces sharp and piecewise-like smoothing that is suitable for applications such as abstraction and non-photorealistic rendering. These optimization-based methods produce in general a high-quality smoothing result and do not introduce halo artifacts in detail manipulation applications. However, they suffer from two issues. The first one is the limited smoothing behavior that is directly related to the choice of the regularizer. For instance, using the l_0 quasi-norm produces only sharp and piecewise-like smoothing, which may be unsuitable for some smoothing applications. The second issue is the computational cost. In order to preserve sharp edges, the regularization term should promote sparsity in the gradient domain. Minimization results in solving large inhomogeneous linear systems in the case of the WLS method or a large iteration number of gradient shrinkage/reconstruction such as the case of the method in [21]. We provide in Figure 1 some smoothing results on synthetic data to compare local filtering methods such as the bilateral filter and optimization methods such as WLS and the proposed method. The input image (a) contains noise on multiple-scales. The bilateral filter (c) is not able to smooth large-scale details and introduces strong edge blurring. The domain transform method [11], which is a fast method based on local filtering, is also not able either to correctly smooth the image. Similar to the bilateral filter, the result (d) contains large-scale noise and blurred edges. The optimization methods WLS and the proposed method produce a much better smoothing result. However, the proposed method does not require solving a large inhomogeneous system and has a flexible smoothing behavior.

In this paper, we present a fast and flexible framework for image smoothing based on non-convex optimization and various approximations to make the method computationally efficient. In the first part of the paper, we show how to efficiently estimate a solution to optimization problems with differentiable non-convex functions. We use a half-quadratic solver with a first order approximation to estimate the solution of the non-convex proximal operator. In the second part of the paper, we show how to accelerate this solver by introducing a warm-start solution and forcing a low number of iterations. We also propose two flexible regularization functions derived from Cauchy and Welsch functions that produce suitable photographic smoothing behavior. In the third part of the paper, we discuss numerical solutions for fast processing. We show how to estimate the proposed filter with few independent convolutions that can fully take advantage of parallel processing. Not only this parallel filtering approach enables fast processing, it also permits the method to be applied to large-scale images. Finally, we present various edge-aware applications produced with our method and compare with state-of-the-art methods.

2 Problem Formulation

Given an input image g, we seek a smooth image u that is close to g under a sparse gradient assumption. The problem is formulated as follows :

$$\underset{u}{\operatorname{argmin}} \frac{\lambda}{2}||u-g||_2^2 + \psi(\nabla u), \tag{1}$$

where $\psi(.)$ is a function and λ is a positive regularization term. Producing a smooth image requires forcing the output u to have sparse gradients. Thus,

$\psi(.)$ should be a sparsity-inducing function[1]. Problem (1) is not easy to solve as the function $\psi(.)$ can be a non-smooth or not even convex. One popular method to tackle optimization problems of this form is by introducing an additional variable v to obtain a half-quadratic form [22]:

$$\underset{u,v}{\text{argmin}} \frac{\lambda}{2}||u-g||_2^2 + \psi(v) + \frac{\beta}{2}||\nabla u - v||_2^2, \tag{2}$$

where β is a new regularization term. The problem then can be solved by alternate minimization :

$$\begin{aligned}(p_1) \ : \ v^{(k+1)} &\leftarrow \underset{v}{\text{argmin}}\, \psi(v) + \frac{\beta}{2}||\nabla u^{(k)} - v||_2^2 \\ (p_2) \ : \ u^{(k+1)} &\leftarrow \underset{u}{\text{argmin}}\, \lambda||u-g||_2^2 + \beta||\nabla u - v^{(k+1)}||_2^2,\end{aligned} \tag{3}$$

where k is the current iteration number. If $\psi(.)$ is convex, then, following [23], it can be shown that (3) converges to u as $k \to \infty$. Otherwise, (3) converges to a local minima (please refer to the Appendix for more details). Problem (p_2) is a least-squares problem and is relatively easy to solve. However, problem (p_1) is hard to solve due to the presence of the non-quadratic function $\psi(.)$. In a nutshell, the function $\psi(.)$ determines the distribution of the gradient of the smooth output image u. In the MAP estimation framework, problem (1) is derived from the Bayes rule $P(u|g) \propto P(g|u)P(u)$. Searching for the smooth image comes to solve the following problem :

$$\underset{u}{\text{argmin}} - \{\log(P(g|u)) + \log(P(u))\}. \tag{4}$$

For instance, in the case of the Laplacian distribution, we get $e^{-\tau|z|}$, which comes to take $\psi(.)$ as the l_1-norm on the gradient. The smoothing behavior of the filter is totally determined by the choice of the function $\psi(.)$. The main issue with this choice is that, in order to correctly smooth strong edges, the distribution of the gradients should be highly kurtotic to promote sparsity in the gradient field, which naturally leads to non-convex functions. We will see through this paper how to perform edge-aware smoothing with non-convex but differentiable functions $\psi(.)$ using a first order proximal estimation.

2.1 Non-Convex Proximal Operators

We discuss here our approach to solve problem (p_1). Note that this problem takes the form of the *proximal operator* [24] :

$$\mathbf{prox}_{th}(x) = \underset{y}{\text{argmin}} \left\{ h(y) + \frac{1}{2t}||y-x||_2^2 \right\}, \tag{5}$$

where t is a positive regularization term. This operator is important in splitting algorithms and has many useful interpretations. For a differentiable function h, the solution can be found by direct minimization of the energy, which leads to the following equation :

$$y + t\nabla h(y) = x. \tag{6}$$

Therefore, the solution of problem (5) is given by the following inverse function :

$$\mathbf{prox}_{th}(x) = (I + t\nabla h)^{-1}(x). \tag{7}$$

Inverse functions are sometimes very hard to evaluate, especially for non-convex functions. For this reason, we use a first order Taylor expansion $h(y) \approx h(x) + \nabla h(x)^T(y-x)$, which results in a more tractable estimation :

$$\mathbf{prox}_{th}(x) \approx x - t\nabla h(x). \tag{8}$$

By replacing x by $\nabla u^{(k)}$, h by ψ and t by $1/\beta$ in problem (p_1), we get the following estimate :

$$v^{(k+1)} \leftarrow \mathbf{prox}_{\frac{1}{\beta}\psi}(\nabla u^{(k)}) \approx \nabla u^{(k)} - \frac{1}{\beta}\nabla\psi\left(\nabla u^{(k)}\right), \tag{9}$$

which simplifies to pixelwise operations :

$$v_p^{(k+1)} \leftarrow \nabla u_p^{(k)}\left(1 - \frac{1}{\beta} w_\psi\left(\nabla u_p^{(k)}\right)\right), \tag{10}$$

where $w_\psi(x) = \frac{\psi(x)'}{x}$ is the weight function of ψ and p is the pixel location. For more details about the derivations and convergence, please refer to the Appendix.

2.2 Photographic Smoothing Behavior

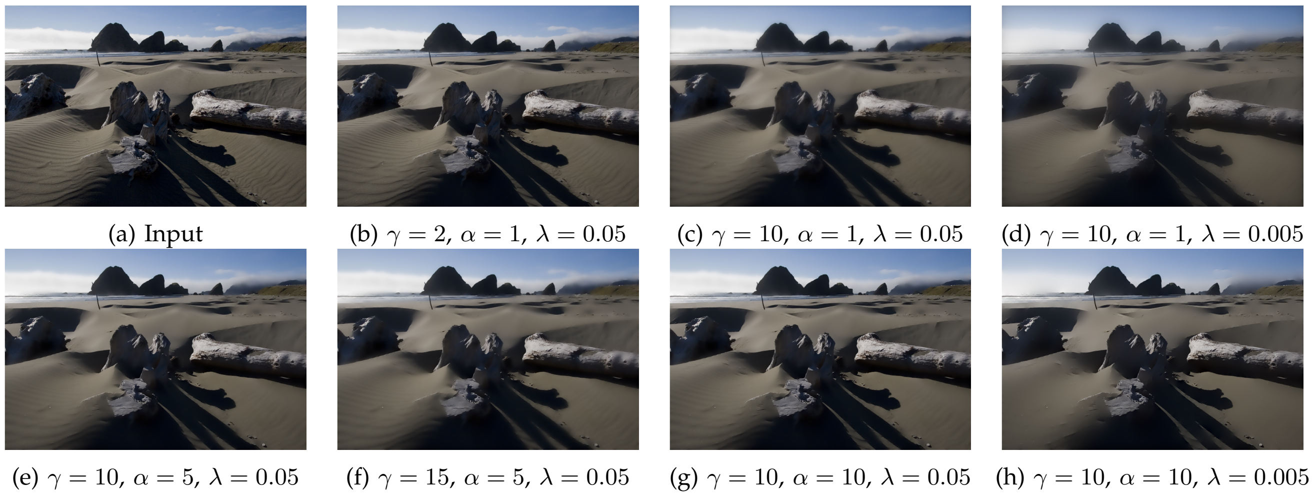
Now that we know how to estimate a solution to the proximal operator, we need to choose an appropriate function $\psi(.)$. As discussed before, one important parameter in edge-aware processing is defining the smoothing behavior. This is directly related to the derivative distribution prior adopted in the method. Studies have shown that real-world images' gradients distribution has a heavier tail than a Laplacian distribution [25], which suggests using a non-convex regularization. In order to be able to generate different smoothing results, we propose two flexible models derived from Cauchy and Welsch functions, two models widely used in M-estimation. The weight functions $w_{\psi_i}(x)$ of Cauchy and Welsh functions are given as follows (please refer to the Appendix) :

$$\begin{aligned}\text{Cauchy :} \quad & w_{\psi_1}(x) = \tfrac{1}{1+(x/\gamma)^2} \\ \text{Welsch :} \quad & w_{\psi_2}(x) = e^{-\left((x/\gamma)^2\right)}\end{aligned} \tag{11}$$

These functions act as thresholding operators as they have an inverted sigmoid-like shape. We introduce a new parameter α that controls the nature of the thresholding :

$$w_1(x) = \frac{1}{1+(x/\gamma)^\alpha} \ , \ w_2(x) = e^{-((x/\gamma)^\alpha)}. \tag{12}$$

1. Typically a function that models a heavy-tailed distribution.



(a) Input (b) $\gamma = 2$, $\alpha = 1$, $\lambda = 0.05$ (c) $\gamma = 10$, $\alpha = 1$, $\lambda = 0.05$ (d) $\gamma = 10$, $\alpha = 1$, $\lambda = 0.005$ (e) $\gamma = 10$, $\alpha = 5$, $\lambda = 0.05$ (f) $\gamma = 15$, $\alpha = 5$, $\lambda = 0.05$ (g) $\gamma = 10$, $\alpha = 10$, $\lambda = 0.05$ (h) $\gamma = 10$, $\alpha = 10$, $\lambda = 0.005$

Fig. 2: Results produced with the proposed method for various parameter settings (image from [17]). The parameter γ controls the main smoothing behavior of the filter, α controls the blur, while λ controls the balance between the original image and the smoothing.

As $\alpha \to \infty$, the functions act more as hard-thresholding operators and approach the l_0 case. As a result, decreasing α produces more blurry results and increasing γ smooths more details. The parameter λ, on the other hand, controls the balance between the original image and the smoothing. Visual results for various parameters using function w_2 can be found in Figure 2.

For color images, we define the following gradient function for full color filtering :

$$T(\nabla u) = \sqrt{\left(\sum_{k=1}^{ch} |\frac{\partial u_k}{\partial x}|\right)^2 + \left(\sum_{k=1}^{ch} |\frac{\partial u_k}{\partial y}|\right)^2}, \quad (13)$$

where ch is the number of channels. The solution of problem (p_1) becomes :

$$v_p^{(k+1)} \leftarrow \nabla u_p^{(k)} \left(1 - \frac{1}{\beta} w_i \left(T(\nabla u^{(k)})_p\right)\right). \quad (14)$$

2.3 Efficient Warm-Start

Solving problem (3) corresponds to an iterative process. Thus, the initial solution $u^{(0)}$ plays an important role in terms of speed of the algorithm. To accelerate the method, we derive a warm-start solution $u^{(0)}$ from (p_2). The warm-start solution corresponds to a rough estimation that can be calculated quickly. Using Euler-Lagrange equation, we rewrite the solution of (p_2) in the matrix form :

$$\underbrace{(\beta L + \lambda I)}_{A}\, \underbrace{u^{(0)}}_{x} = \underbrace{\left(-\beta \mathrm{div}(v^{(0)}) + \lambda g\right)}_{b}, \quad (15)$$

where $Lu^{(0)} \equiv -\mathrm{div}(\nabla u^{(0)})$. We consider the quadratic form $f(x) = \frac{1}{2}x^T Ax - b^T x$. Solving for $f(x) = 0$ is equivalent to solving the system (15). To estimate a rough solution of f, we use a quadratic regularization, which takes the form of the proximal form on the point $x^{(0)}$:

$$\mathbf{prox}_{tf}(x^{(0)}) = \underset{x}{\mathrm{argmin}} \left\{ f(x) + \frac{1}{2t}||x - x^{(0)}||_2^2 \right\}, \quad (16)$$

where $x^{(0)}$ is close to x. Applying the first order approximation of equation (8), we get the linearized form :

$$x \approx x^{(0)} - t\left\{Ax^{(0)} - b\right\}, \quad (17)$$

Now replacing $x^{(0)}$ with the input image g and applying equation (14) on $v^{(0)}$, we get the first order estimation :

$$u^{(0)} \approx g + \xi \mathrm{div}\left(\nabla g - \nabla g \circ (1 - \frac{1}{\beta} w_i(T(\nabla g)))\right), \ \xi > 0, \quad (18)$$

where $\circ$ is a pointwise multiplication operator and $\xi = \beta t$. As we want a relatively small number of iterations, we rather fix $\beta = 1$ in the rest of the paper. This choice has two motivations. First, as the functions w_1 and w_2 have values between 0 and 1, the term $(1 - \frac{1}{\beta} w_i)$ will also have values between 0 and 1 for $\beta = 1$, and thus one can use only the parameters γ and α to control smoothing. The second reason is that, with the same β at each iteration, one does not need to update the preconditioner if equation (p_2) is solved using the preconditioned conjugate gradient method, or update the filters in the separable filters approach that will be introduced in the next section. In a theoretical point of view, β should slightly increase at each iteration. As we force a low iteration number for faster processing, fixing β has little impact on the truncated solution and offers a more efficient implementation. For the warm-start parameters, ξ is set between 0.01 and 0.25, α is fixed to 2 and γ is set to around 400. Figure 3 shows the importance of the warm-start solution when filtering at low iterations.

Only 3 iterations were used to produce the smoothing result. The input image is the same as in Figure 1.

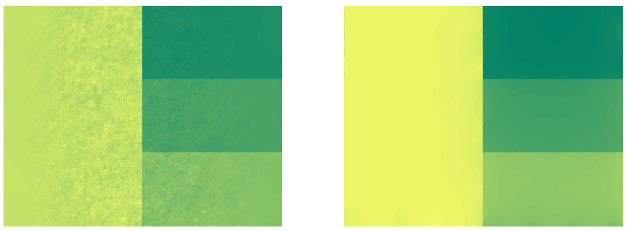

(a) No warm-start (b) With warm-start

Fig. 3: Smoothing example showing the importance of the warm-start solution when filtering at low iterations (3 iterations). We took : $\gamma = 4.1$, $\alpha = 12$ and $\lambda = 5 \times 10^{-4}$.

2.4 Fast Numerical Solution

Here we discuss fast methods to compute problem (p_2). The problem is quadratic and the solution is given by Euler-Lagrange equation :

$$\lambda u^{(k+1)} - \nabla^2 u^{(k+1)} = \lambda g - \text{div}(v^{(k+1)}), \tag{19}$$

One approach to solve equation (19) consists in using the Fourier transform. As the divergence and Laplacian operators can be expressed using convolutions, introducing the Fourier transform permits to split the differential operators from the variable $u^{(k+1)}$ and the solution is given as follows (for $\beta = 1$) :

$$u^{(k+1)} \leftarrow \mathcal{F}^{-1}\left(\frac{\mathcal{F}\left(\lambda g - \text{div}(v^{(k+1)})\right)}{\lambda - \text{lap}}\right), \tag{20}$$

where $\mathcal{F}$ is the Fourier transform, div is the discrete divergence operator [2] and lap is the OTF (optical transfer function) of the discrete Laplacian filter. Calculating the filter lap depends only on the size of the image and can be calculated only once when for example the image is loaded by the user, or even pre-stored for multiple image sizes. Another approach to solve equation (19) is using sparse linear solvers. Differential operations can expressed as linear operations using discrete differential operators D_x and D_y, the solution corresponds to the following linear system :

$$\underbrace{(L + \lambda I)}_{A}\underbrace{u^{(k+1)}}_{x} = \underbrace{\left(-\text{div}(v^{(k+1)}) + \lambda g\right)}_{b}. \tag{21}$$

The matrix A is symmetric, positive-semidefinite and very sparse due to the Laplacian matrix L. The good news about this system is that it corresponds to a *homogeneous* system [26]. Contrary to *inhomogeneous* systems as the one that corresponds to the WLS solution [16], they are much easier to solve. Using the latest Laplacian matrix pre-conditioner by Krishnan *et al.* [26], we have noticed that solving equation (20) takes in general only one iteration, and two iterations for very small λ values. This is very fast processing : for two iterations of the proposed method, around $2 \times 3 = 6$ pre-conditioned conjugate gradient iterations in total are required to perform edge-aware smoothing of a full-color image. Solving equation (p_1) corresponds to simple pointwise image processing. Note also that, as β is fixed, the method does not need to update the preconditioner. Hence, the Laplacian matrix and the preconditioner for various discrete λ values and image sizes can be pre-computed and stored for faster processing.

Note however that the use of these two approaches can be problematic for large-scale images due to large memory requirement. We propose a simple method to extend the proposed fast edge-processing method to large-scale processing and can fully take advantage of parallel processing. The idea consists in transforming the deconvolution operation in equation (20) first to a convolution operation, then estimating the convolution kernel with separable filters. Equation (20) can be written as the following convolution :

$$u^{(k+1)} \leftarrow \underbrace{\left(\lambda g - \text{div}\left(v^{(k+1)}\right)\right)}_{g_{conv}} \star \underbrace{\mathcal{F}^{-1}\left(\frac{1}{\lambda - lap}\right)}_{G_\lambda}. \tag{22}$$

Unfortunately, the filter G_λ has a large support and using a straightforward convolution is costly. However, note that : 1) the size of the filter depends only on λ, 2) as λ becomes larger, the filter G_λ tends to Dirac's delta function, a 1×1 filter. This means that for a given $\lambda > 0$ value, G_λ can be estimated with a smaller kernel of size $h \times h$, given an error tolerance. Figure 4 shows how the support of the filter becomes smaller when λ becomes larger. The following table shows the MSE (Mean Square Error) between the ground truth filter G_λ estimated in a grid of size 1201×1201 and its truncated version, for various values of λ and various sizes without any rescaling :

TABLE 1: MSE of the truncated filter for different kernel sizes.

	Kernel size	MSE	Kernel size	MSE
$\lambda = 1$	201x201	$5.22\,10^{-20}$	71x71	$5.3\,10^{-20}$
$\lambda = 1$	29x29	$2.16\,10^{-12}$	7x7	$8\,10^{-8}$
$\lambda = 0.1$	201x201	$1.3\,10^{-18}$	71x71	$3.8\,10^{-10}$
$\lambda = 0.1$	29x29	$2.75\,10^{-7}$	7x7	$7\,10^{-6}$
$\lambda = 0.03$	201x201	$2.6\,10^{-12}$	71x71	$1.95\,10^{-7}$
$\lambda = 0.03$	29x29	$6.7\,10^{-6}$	7x7	$3.43\,10^{-5}$

The table above simply shows that one can indeed estimate the filter G_λ with high accuracy using a much smaller truncated filter. As a result, instead of using two Fourier transforms, equation (21) can

2. The discrete Laplacian operator Δ corresponds to $\Delta := -(\nabla_x^T \nabla_x + \nabla_y^T \nabla_y) := L$, where L is the discrete Laplacian matrix.

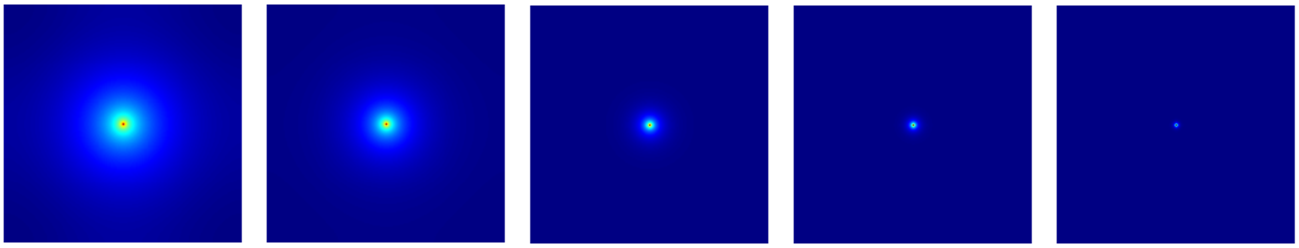

(a) $\lambda = 10^{-9}$ (b) $\lambda = 10^{-3}$ (c) $\lambda = 0.01$ (d) $\lambda = 0.06$ (e) $\lambda = 0.5$

Fig. 4: As λ becomes larger, the G_λ filter tends to Dirac's delta function as shown in these experiments.

be calculated using a convolution with a medium-size kernel. In the next paragraph, we show that the cropped version of the filter can be estimated with a small set of separable filters which offers significant speed-up compared to a straightforward convolution.

Efficient Estimation via Separable Filters

A convolution with a large kernel is still costly. It turns out that the kernel in our case can be written as the sum of few separable filters making the process much faster. It is well known that a matrix of rank 1 can be written as the product of two vectors. More generally, if a kernel is of rank r, it can be written as the sum of r two successive convolutions with $1D$ kernels. This operation is highly parallelizable and can make our filter faster for large-scale processing. Mathematically speaking, the filters are given by SVD decomposition

$$G_\lambda = D\Sigma S^T = \sum_{i=1}^{r} \sigma_i d_i s_i^T, \tag{23}$$

where d_i and s_i denote respectively the i-th column of the matrices D and S, and σ_i denotes the singular value at position i. It turns out that the filter G_λ is real symmetric and positive-semidefinite. Hence, calculating $u^{(k+1)}$ becomes :

$$u^{(k+1)} \leftarrow \sum_{i=1}^{r} (g_{conv} \star \sqrt{\sigma_i} d_i) \star \sqrt{\sigma_i} d_i^T \tag{24}$$

In order to perform less filtering operations, we take the sum to $r_t < r$ instead of the full rank of the matrix. This comes to using a *truncated* SVD and corresponds to the best r_t-rank approximation in the sense of the squared Frobenius norm. We found that, for $\lambda \gtrsim 0.04$, $r_t = 2$ or 3 is generally enough to guarantee high-quality visual results as shown in Figure 5.

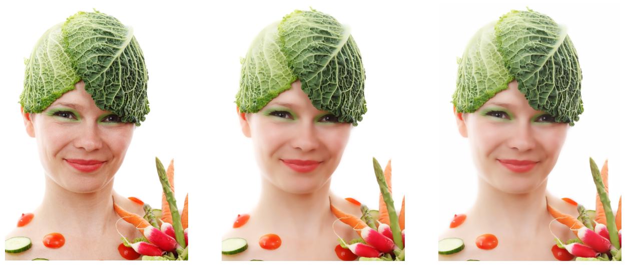

(a) Input (b) FFT result (c) Separable filters

Fig. 5: Smoothing example with the proposed method (image from publicdomainpictures.net). (b) smoothing performed with the Fourier transform, (c) smoothing performed using the separable filters approach using 2 separable filters of size $h = 31$. The parameters were set as follows : $\lambda = 0.07$, $\gamma = 9$, $\alpha = 2$ using function w_2.

2.5 Analysis

The proposed method consists mainly in two steps. The first step consists in calculating the gradient, the weights, the divergence and the summation. These are pixelwise operations that can be calculated quickly and take advantage of parallel processing. In fact, they can be coded more efficiently by combining for example the gradient and the weights calculations in the same loop. They can also take advantage of SIMD instructions for even more efficient processing. The second step consists in the image reconstruction. Here again, the processing time reported in this paper is for serial filtering. A more efficient implementation would perform filtering in parallel using SIMD instructions for instance. For the separable filtering (SP) approach, the reconstruction cost is $r_t \times 2 \times h$ operation per pixel, where r_t operations can be performed in parallel. Concerning the warm-start solution, it consists in pixelwise operations. For large images, the weights can be calculated on a downsampled image for faster processing. We use the following setup : Intel Xeon CPU E5-2609 2.24 GHz and Nvidia Tesla C2075 GPU on Matlab 2013a and Linux 64bits, Intel i7-2670QM 2.20Ghz on Visual Studio 2008 and Windows 7. Processing time for various methods can be found in Table 2. NaN means that the method encountred out of memory error. As can be seen, the proposed method offers attractive efficiency with a non-optimized Matlab implementation. We expect to get faster processing with an optimized and parallelized C/C++ implementation.

3 Applications

We present in this section various applications produced with the proposed method as well as a comparison with various state-of-the-art methods.

	CPU/C++			CPU/Matlab	
Resolution	BLF [10]	BLF [7]	MD [19]	Fast-LLF [18]	L_0 [21]
256×256	0.004	0.028	1.55	0.5	0.17
512×512	0.017	0.11	5.78	0.9	0.77
1024×768	0.05	0.34	14.82	1.9	2.42
1920×1080	0.13	0.90	35.84	4.3	6.76
2048×1536	0.20	1.37	69.90	6.3	11.26
4096×3072	0.84	NaN	NaN	21	45.33
6400×4800	NaN	NaN	NaN	48	NaN

CPU/Matlab			GPU/Matlab		
warm-start	FFT	SF	warm-start	FFT	SF
0.011	0.014	0.027	0.0037	0.006	0.009
0.035	0.052	0.075	0.0047	0.011	0.013
0.04	0.15	0.20	0.011	0.035	0.03
0.09	0.38	0.45	0.027	0.097	0.07
0.14	0.66	0.68	0.04	0.138	0.11
0.55	2.7	2.6	0.17	0.54	0.40
1.26	NaN	6.2	0.44	NaN	NaN

TABLE 2: Processing time in secondes for various methods (left) and our method (right).

3.1 Image Smoothing

Image smoothing is a very popular method to filter out small noise and produce abstracted versions of a natural image. It is the building-block for a wide range of applications such as detail manipulation, HDR tone-mapping, fast edge simplification and video edge-aware processing.

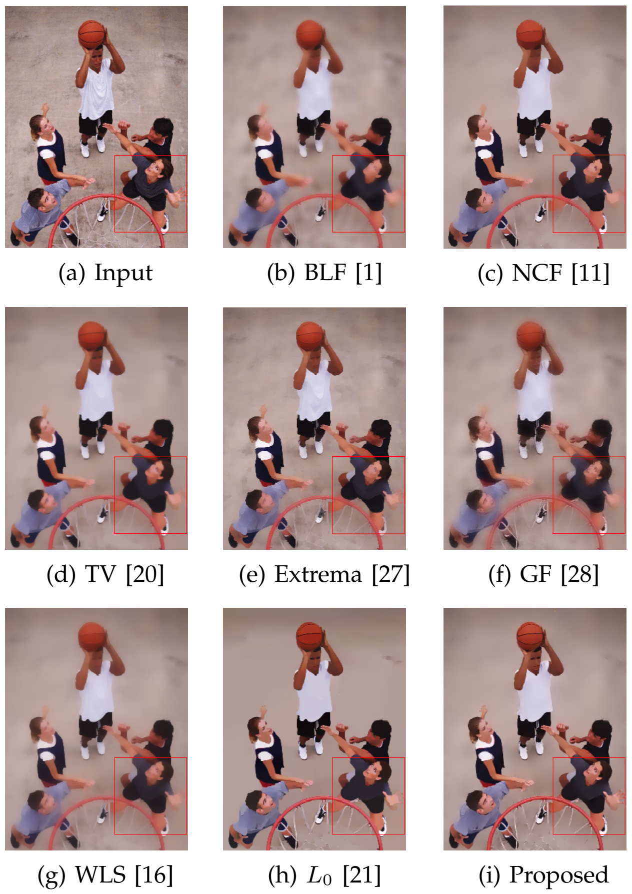


Fig. 6: Smoothing comparison with various state-of-the-art methods (image from [21]). The proposed method produces high-quality smoothing while being flexible and computationally efficient. A close-up is given in Figure 7.

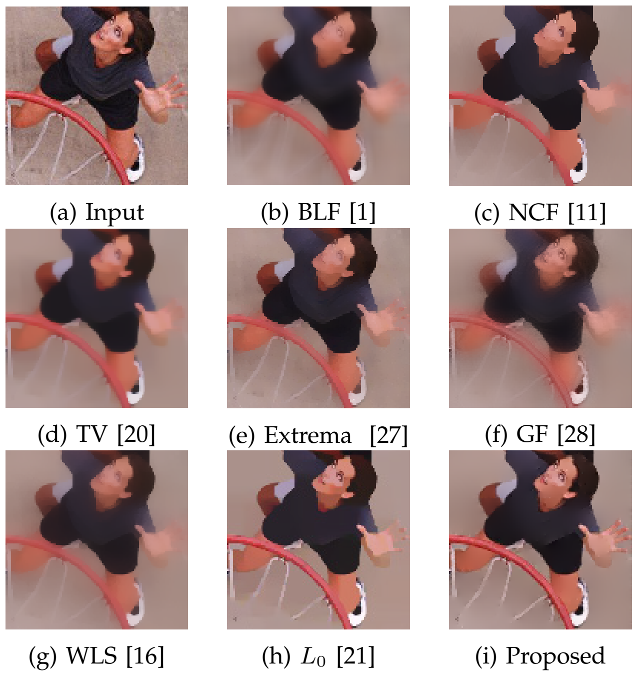
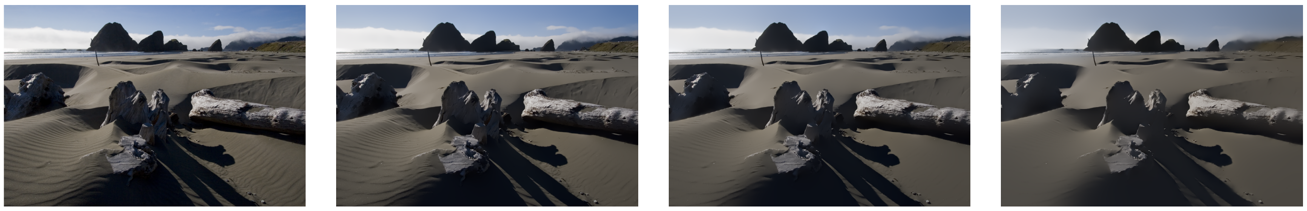


Fig. 7: Close-up on Figure 6. Note how the proposed method preserved the features on the face and the hand.

Smoothing Quality Comparison

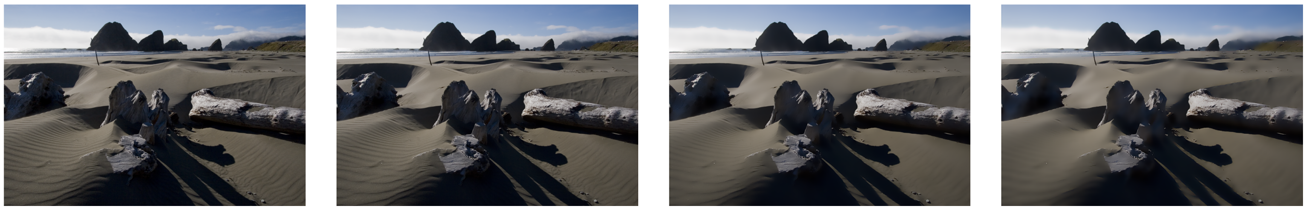
A natural image smoothing example is given in Figure 6 as well as a comparison with 7 state-of-the-art methods. The bilateral filter result (b) ($\sigma_s = 40$, $\sigma_r = 0.25$) produces a globally blurred result. The most important salient structures such as the face and the hand of the lady were completely washed-out. This result was produced with a brute-force bilateral filter implementation, which offers the best BLF quality possible as some fast BLF implementations do not produce exactly the same quality as the brute-force implementation. The second result (c) was produced with the fast Domain Transform method using the NC filter ($\sigma_s = 50$, $\sigma_r = 0.47$). The global blurring produced by the BLF is much less present, but the method is not able to preserve some important salient structures such as the face and the hand of the lady. The third example (d) is produced with our implementation of TV regularization using a half-quadratic solver. The use of a half-quadratic solver

Local Laplacian Filters [17]



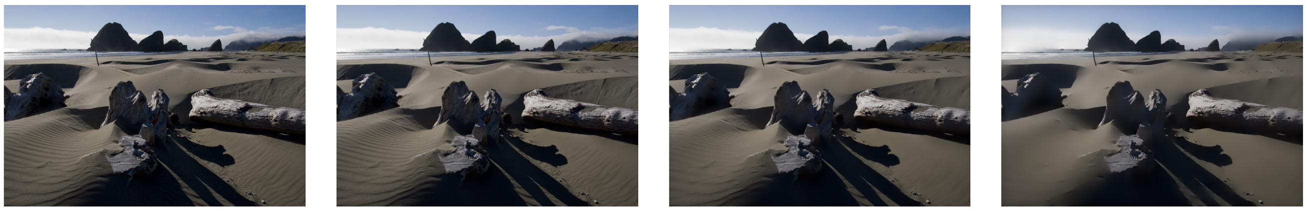
(a) Input (b) $\sigma_r = 0.1$, $\alpha = 4$, $\beta = 1$ (c) $\sigma_r = 0.2$, $\alpha = 4$, $\beta = 1$ (d) $\sigma_r = 0.4$, $\alpha = 4$, $\beta = 1$

Mixed-Domain Method [19]



(e) Input (f) $\alpha = 0.8$, $\beta = 1$, $\sigma = 0.18$ (g) $\alpha_r = 0.4$, $\beta = 1$, $\sigma = 0.18$ (h) $\alpha_r = 0.08$, $\beta = 1$, $\sigma = 0.18$

Proposed



(i) Input (j) $\gamma = 2$, $\alpha = 20$, $\lambda = 0.05$ (k) $\gamma = 5$, $\alpha = 20$, $\lambda = 0.05$ (l) $\gamma = 9.5$, $\alpha = 20$, $\lambda = 0.005$

Fig. 8: Comparison with multi-scale local filters. Our method produces comparable smoothing quality while offering computational efficiency in addition.

here is important so we can compare it with the L_0-minimization method and the proposed method. With parameters $\lambda = 0.1$ and $\kappa = 2$, the method took around 23 iterations, which comes to the cost of using 23×3 FFTs. The result contains a global blurring, and some salient structures were also not preserved. The local extrema method that consists in extracting the extrema envelopes and produce a smooth signal by calculating the mean of these envelopes has shown to be effective in textured areas. The method in this case (e) was not able to correctly smooth the background and also was not able to correctly smooth important salient structures. The forth result is produced by the Guided Filtering method ($radius = 5$, $\epsilon = 0.15^2$). The GF method is fast but produces severe global blur in the smoothing result. The fifth example is produced using the Weighted Least Squares method of Farbman *et. al* ($\lambda = 0.25$, $\alpha = 1.2$). The method seems to produce a better result than TV regularization (d) but the most important salient structures were blurred. Note also that the method requires solving a large inhomogenous linear system that becomes harder to solve as λ increases. The seventh result (h) is produced with the L_0 gradient minimization method ($\lambda = 0.015$, $\kappa = 2$). The approach produces a better result than the other 6 results. The faces are better preserved and the method does not suffer from the global blur produced by most of the other methods. However, the method required 23×3 FFTs. Finally, the proposed method (i) ($\gamma = 12$, $\alpha = 20$, $\lambda = 0.04$, function w_1) produces a high-quality smoothing result at a low computational cost. Note how the salient structures such as the faces and the hand of the lady on the left are preserved. The method required 2 iterations. This is more than 12 times faster than L_0-minimization when using the FFT approach. Using the sparse linear system approach with Krishnan *et al.* preconditioner [26], the method required a total of $2 \times 1 \times 3 = 6$ preconditioned conjugate gradient iterations, where the 3 iterations in each iteration can be performed in parallel. Note also that the preconditioner does not to be updated and can be fixed in advance according to the value λ.

We compare with multi-scale local filtering methods such as Local Laplacian Filters [17] and the mixed-domain method [19]. These methods produce high-quality results but they are relatively computationally demanding. Results are given in Figure 8. As can be seen, the proposed smoothing method produces comparable smoothing quality at lower computational cost. Note that the results presented here are produced

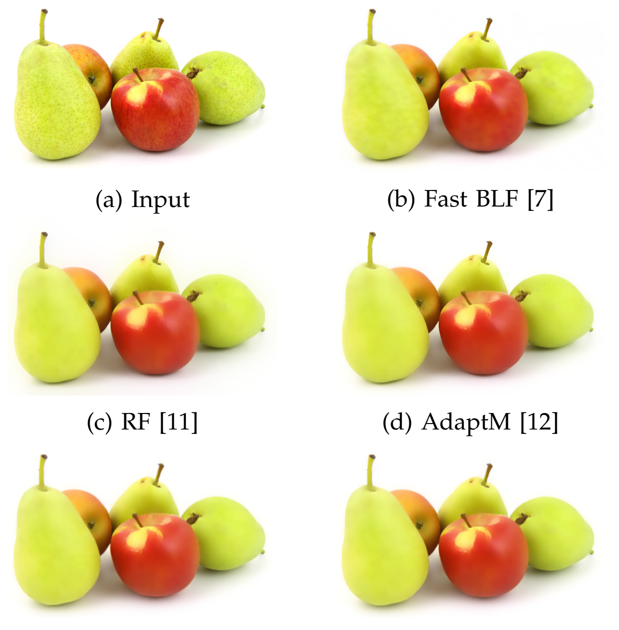

(a) Input (b) Fast BLF [7]
(c) RF [11] (d) AdaptM [12]
(e) Proposed (1 iteration) (f) Proposed (2 iterations)

Fig. 9: Smoothing comparison with various fast state-of-the-art methods (image from publicdomainpictures.net). The proposed method produces a high-quality smoothing.

with the original Local Laplacian Filter implementation that is computationally very demanding. The Fast Local Laplacian Filters method [18] that aims at making the Local Laplacian Filter faster is only an approximation and does not produce comparable smoothing quality.

Fast Smoothing Comparison

Another smoothing example is presented in Figure 9. We evaluate the quality and speed of 3 fast edge-aware filters : the fast implementation of the bilateral filter in [7], the domain transform method (RF filter) in [11] and the adaptive manifolds method in [12]. We use the following setup : Intel Xeon CPU E5-2609 2.24 GHz, Matlab 2013a on Linux 64bits and an Intel i7-2670QM 2.20Ghz on Windows 7. The result (b) was produced with the author's C++ code (Visual Studio). It took around 0.09 secs to produce this result with values ($\sigma_s = 0.01$, $\sigma_r = 0.1$). The adaptive manifolds method took around 0.4 seconds (Matlab implementation) to produce the result (d) ($\sigma_s = 10, \sigma_r = 0.2$). The proposed method produces a high-quality result even using one single iteration (e). The result is further improved with a second iteration. The pears are correctly smoothed and the image does not contain background artifacts. With a Matlab implementation and using the FFT method, the proposed method with 1 iteration took around 0.045 seconds, and around 0.09 seconds with 2 iterations with the following parameters : $\gamma = 12$, $\alpha = 25$, $\lambda = 0.03$ using the function w_1. The processing is around 5 times faster using Matlab's GPU computing toolbox.

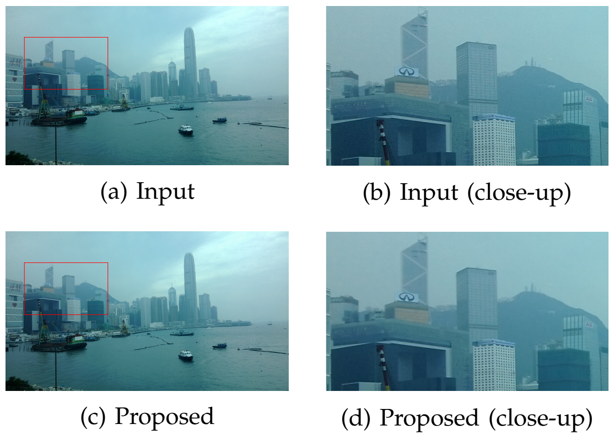

(a) Input (b) Input (close-up)
(c) Proposed (d) Proposed (close-up)

Fig. 10: Large-scale image smoothing of a real-world smartphone 4MP image.

Large-Scale Image Smoothing

We present here an example of large-scale processing with the proposed method on a real photo taken with a smartphone (Nokia Lumia 620) in Figure 10. Mid-range smartphones can produce high-resolution images but the result is relatively noisy. Using the FFT approach in this case can cause memory issues. We rather use the proposed filter with the separable filters approach to perform noise reduction. We took $r_t = 2$ (two separable filters) and $h = 15$ for parameters $\gamma = 12$, $\alpha = 8$, $\lambda = 0.7$, function w_2 for only 1 iteration. The total cost of the smoothing here comes to two successive convolutions with a 1×15 filter two times, which is computationally efficient. It took around 0.82 seconds to filter the image on Matlab without parallel processing.

3.2 Multi-Scale Detail Enhancement

Edge-aware filtering can be used to decompose one image into several layers according to its degrees of details. One can thereafter manipulate each layer and recombine them to boost details on multiple scales [16]. Let B_0,...,B_k be different smoothed versions of the input image g. As k becomes larger, B_k becomes coarser, with $g = B_0$. These layers are called *base layers*. Detail layers D_l are extracted by subtracting the base layer B_{l+1} from its richer version B_l as $D_l = B_l - B_{l+1}$. Each detail layer is then multiplied by a parameter and summed to form the output image. Figure 11 (b) presents an example of multi-scale detail enhancement applied to the flower (a). As can be seen, the result generated with the proposed method is visually similar to the one produced with the WLS filter (c). However, generating the two layers took only a total of 0.048 seconds with our approach. The WLS method [16] took 2.7 seconds

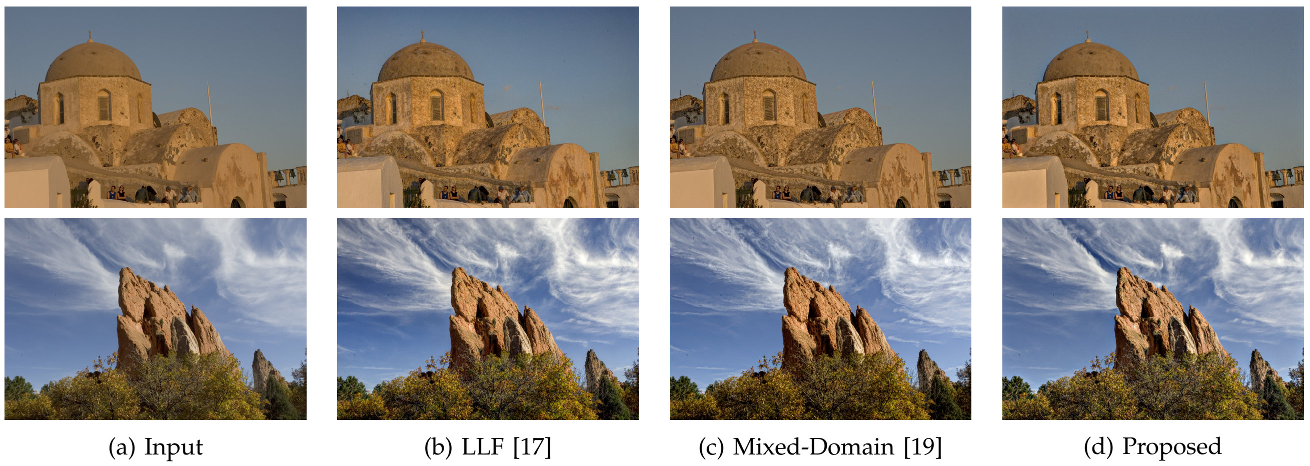


Fig. 12: Detail enhancement examples (images from [17]). The proposed method produces a high-quality detail manipulation with reduced processing time.

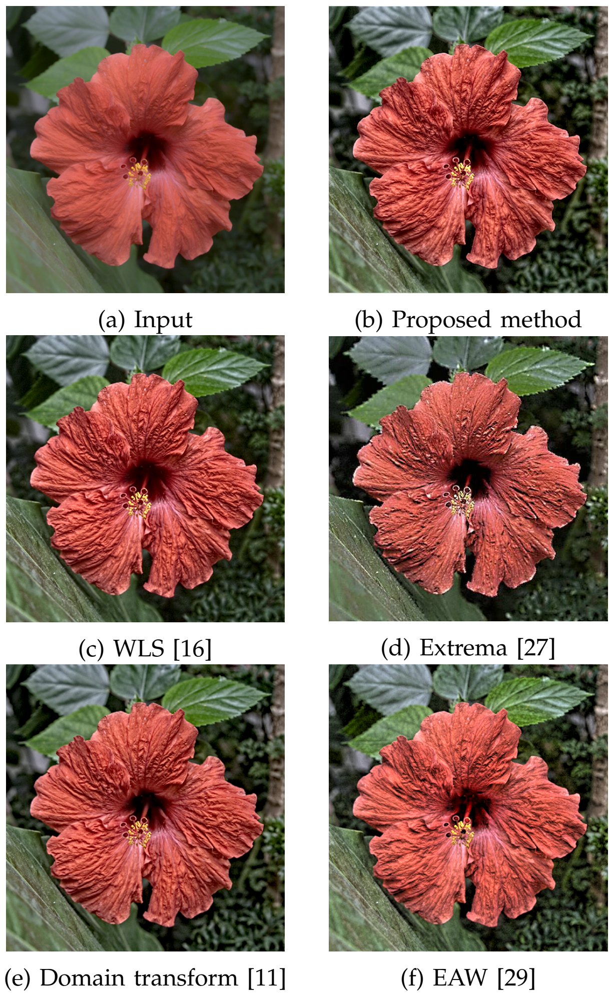


Fig. 11: Fine Detail Enhancement. (a) Input image. (b) Proposed method using w_1. (c) Weighted-least squares approach [16]. (d) Local extrema [27]. (e) Domain transform [11]. (f) Edge-avoiding wavelets from [29].

with the direct solver and around 1.5 seconds with the PCG method [30] using an incomplete Cholesky factorization preconditioner (the method took more processing time with the preconditioner in [26] due to the preconditioner update processing time. The incomplete Cholesky factorization preconditioner produces in this case a high-quality result while being fast to generate). Figure 12 shows examples of detail enhancement compared with the Local Laplacian Filters [17] and the mixed-domain [19] methods. As can be seen, the proposed method offers high-quality results with reduced processing time.

3.3 HDR Tone Mapping

Edge-aware filtering can be used for tone-mapping high dynamic range images by performing a multi-scale decomposition of the log-luminance channel similar to the one discussed in the previous paragraph. We present in Figure 14 an example of HDR tone mapping with our approach using one detail layer. Our result (d) is artifact-free and visually similar to (c). The proposed solution took only 0.025 seconds on Matlab to extract the base layer. Another example of HDR tone mapping is presented in Figure 13 with a 3 layers decomposition. The method produces a suitable photographic look with reduced processing time.

3.4 Edge Simplification

Edge extraction is an important application in computer vision and graphics. The goal is to extract the perceptually most important structures from a natural image. Natural signals are very complex with unpredictable perturbations everywhere that makes accurate edge extraction extremely difficult. Instead of designing a highly sophisticated and adapted edge detector, which is very hard to realize, edge-aware filters can be used to reduce unecessary details and better extract edges using regular edge detectors. We propose to use the filter presented herein for edge

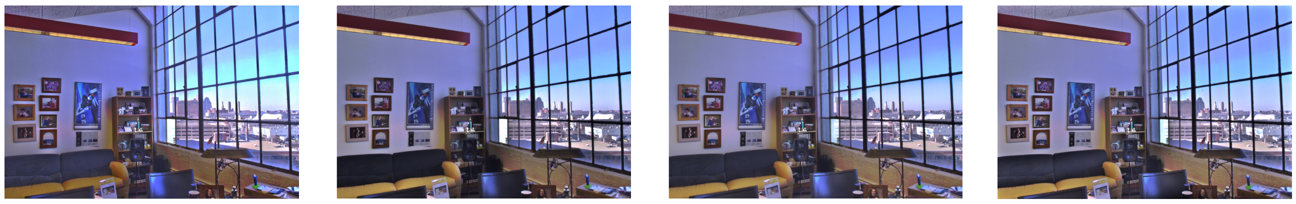

(a) BLF [3] (b) RF [11] (c) WLS [16] (d) Proposed method

Fig. 14: HDR tone mapping example. (a) Result with the bilateral filter. (b) Result using the domain transform approach. (c) Result using WLS [16]. (d) Our result for 1 iteration with $\lambda = 0.006$, $\gamma = 40$ and $\alpha = 2$ using the function w_2.

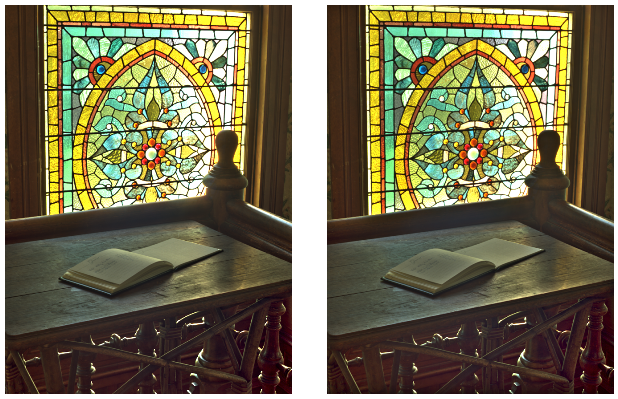

(a) WLS [16] (b) Proposed method

Fig. 13: Multi-layer HDR Tone Mapping. (a) Result with WLS. (b) Proposed method. *(HDR image © Industrial Light & Magic. All rights reserved.)*

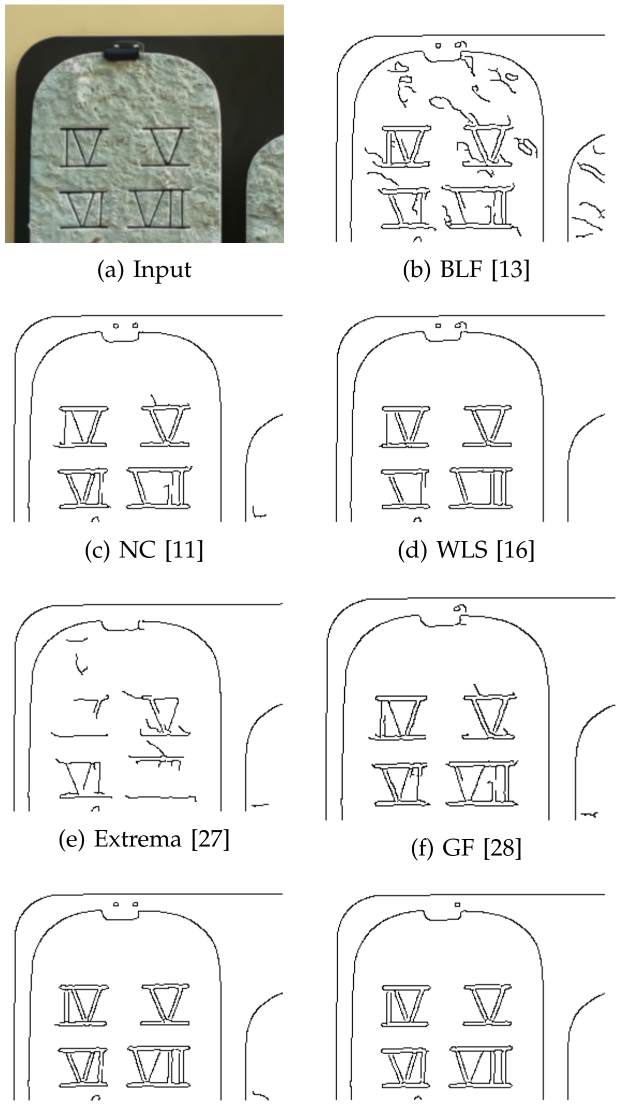

(a) Input (b) BLF [13] (c) NC [11] (d) WLS [16] (e) Extrema [27] (f) GF [28] (g) L_0 [21] (h) Proposed method

Fig. 15: Edge simplification example (picture from [21]). The proposed method permits to extract relevant edge structures while being computationally efficient.

extraction and compare it with various state-of-the art filters. To evaluate the performance of each method, we used the same edge detector [31] for all the filters. Smoothing is applied only on the luminance channel. Results are presented in Figure 15. Note how the proposed filter (h) is able to get rid of the unecessary details and correctly captures important structures. Methods such as bilateral filter (b) and extrema method (e) are much less adapted to this task.

3.5 Fast Video Processing

One important point about an edge-aware filter is its ability to be extended for video processing. Temporal coherence must be hold in order to prevent flickering effects. It turns out that the proposed method is temporally coherent even when performing frame-per-frame filtering. Theoretically, temporal coherence can be easily added in our method by simply adding the temporal gradients and adjusting it to the temporal gradients of the original video. The method in this case would require a 3D FFT. We found that simple frame-per-frame filtering works already quite well and does not introduce any flickering effect. We can thus take advantage of the separable filters approach for fast large-scale video smoothing. GPU processing can be used instead of the CPU for real-time processing. An example is given in Figure 16 for a video frame. The smoothing was performed in Matlab using a Tesla C2075 GPU device in real-time.

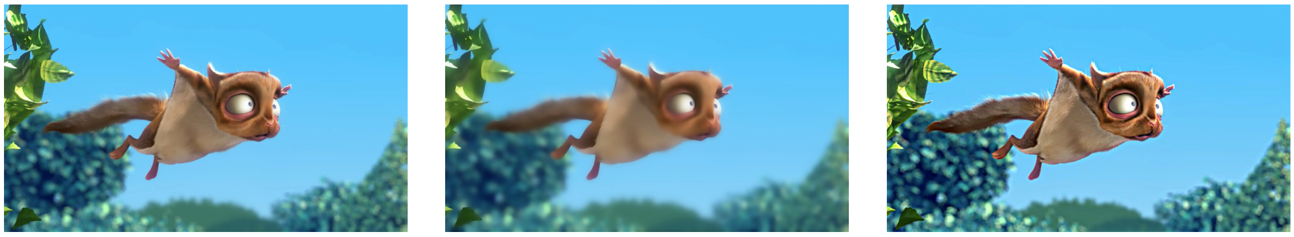

(a) Input frame (b) Smoothing result (c) Detail enhancement

Fig. 16: Fast video filtering example (©2008, Blender Foundation / www.bigbuckbunny.org)

4 Conclusion

We present a fast solution for high-quality edge-aware processing. The proposed approach is based on non-convex optimization and makes use of various mathematical tools to perform efficient processing. First, we show how to estimate non-convex differentiable proximal operators using a first order approximation. Secondly, we use a first order proximal estimation to derive a warm-start solution that permits to generate high-quality smoothing at low iterations. The third contribution of the paper consists in proposing numerical estimations for even faster processing. We show that the proposed filter can be reduced to few convolutions with separable filters such that the size of the filters is independent of the size of the image. The separable filters approach permits fast large-scale image processing at low memory cost. We demonstrate the effectiveness of the proposed method on various applications such as image smoothing, detail manipulation, HDR tone-mapping and edge simplification and compare with various state-of-the-art methods.

References

[1] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proceedings of the Sixth International Conference on Computer Vision*, ser. ICCV '98. Washington, DC, USA: IEEE Computer Society, 1998, pp. 839–. [Online]. Available: http://dl.acm.org/citation.cfm?id=938978.939190

[2] D. Barash, "A fundamental relationship between bilateral filtering, adaptive smoothing, and the nonlinear diffusion equation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 6, pp. 844–847, Jun. 2002. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2002.1008390

[3] F. Durand and J. Dorsey, "Fast bilateral filtering for the display of high-dynamic-range images," in *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '02. New York, NY, USA: ACM, 2002, pp. 257–266. [Online]. Available: http://doi.acm.org/10.1145/566570.566574

[4] J. Chen, S. Paris, and F. Durand, "Real-time edge-aware image processing with the bilateral grid," *ACM Trans. Graph.*, vol. 26, no. 3, Jul. 2007. [Online]. Available: http://doi.acm.org/10.1145/1276377.1276506

[5] S. Paris and F. Durand, "A fast approximation of the bilateral filter using a signal processing approach," *Int. J. Comput. Vision*, vol. 81, no. 1, pp. 24–52, Jan. 2009. [Online]. Available: http://dx.doi.org/10.1007/s11263-007-0110-8

[6] F. Banterle, M. Corsini, P. Cignoni, and R. Scopigno, "A low-memory, straightforward and fast bilateral filter through subsampling in spatial domain." *Comput. Graph. Forum*, vol. 31, no. 1, pp. 19–32, 2012.

[7] Q. Yang, K.-H. Tan, and N. Ahuja, "Real-time o(1) bilateral filtering," in *CVPR*, 2009, pp. 557–564.

[8] A. Adams, N. Gelfand, J. Dolson, and M. Levoy, "Gaussian kd-trees for fast high-dimensional filtering," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 21:1–21:12, Jul. 2009. [Online]. Available: http://doi.acm.org/10.1145/1531326.1531327

[9] Q. Yang, S. Wang, and N. Ahuja, "Svm for edge-preserving filtering," in *CVPR*, 2010, pp. 1775–1782. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2010.5539847

[10] Q. Yang, "Recursive bilateral filtering," in *ECCV*, 2012, pp. 399–413. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-33718-5_29

[11] E. S. L. Gastal and M. M. Oliveira, "Domain transform for edge-aware image and video processing," *ACM Trans. Graph.*, vol. 30, no. 4, pp. 69:1–69:12, Jul. 2011. [Online]. Available: http://doi.acm.org/10.1145/2010324.1964964

[12] ——, "Adaptive manifolds for real-time high-dimensional filtering," vol. 31, no. 4. New York, NY, USA: ACM, Jul. 2012, pp. 33:1–33:13. [Online]. Available: http://doi.acm.org/10.1145/2185520.2185529

[13] H. Winnemöller, S. C. Olsen, and B. Gooch, "Real-time video abstraction," in *ACM SIGGRAPH 2006 Papers*, ser. SIGGRAPH '06. New York, NY, USA: ACM, 2006, pp. 1221–1226. [Online]. Available: http://doi.acm.org/10.1145/1179352.1142018

[14] J. Kopf, M. F. Cohen, D. Lischinski, and M. Uyttendaele, "Joint bilateral upsampling," *ACM Trans. Graph.*, vol. 26, no. 3, Jul. 2007. [Online]. Available: http://doi.acm.org/10.1145/1276377.1276497

[15] L. Yuan, J. Sun, L. Quan, and H.-Y. Shum, "Progressive inter-scale and intra-scale non-blind image deconvolution," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 74:1–74:10, Aug. 2008. [Online]. Available: http://doi.acm.org/10.1145/1360612.1360673

[16] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski, "Edge-preserving decompositions for multi-scale tone and detail manipulation," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 67:1–67:10, Aug. 2008. [Online]. Available: http://doi.acm.org/10.1145/1360612.1360666

[17] S. Paris, S. W. Hasinoff, and J. Kautz, "Local laplacian filters: Edge-aware image processing with a laplacian pyramid," in *ACM SIGGRAPH 2011 Papers*, ser. SIGGRAPH '11. New York, NY, USA: ACM, 2011, pp. 68:1–68:12. [Online]. Available: http://doi.acm.org/10.1145/1964921.1964963

[18] S. H. J. K. M. Aubry, S. Paris and F. Durand, "Fast local laplacian filters: Theory and applications," *ACM Transactions on Graphics*, 2014.

[19] X.-Y. Li, Y. Gu, S.-M. Hu, and R. R. Martin, "Mixed-domain edge-aware image manipulation," *IEEE Transactions on Image Processing*, vol. 22, no. 5, pp. 1915–1925, 2013.

[20] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Phys. D*, vol. 60, no. 1-4, pp. 259–268, Nov. 1992. [Online]. Available: http://dx.doi.org/10.1016/0167-2789(92)90242-F

[21] L. Xu, C. Lu, Y. Xu, and J. Jia, "Image smoothing via l0 gradient minimization," in *Proceedings of the 2011 SIGGRAPH Asia Conference*, ser. SA '11. New York, NY, USA: ACM, 2011, pp. 174:1–174:12. [Online]. Available: http://doi.acm.org/10.1145/2024156.2024208

[22] D. Geman and C. Yang, "Nonlinear image recovery with half-quadratic regularization," 1993.

[23] J. Idier, "Convex half-quadratic criteria and interacting auxiliary variables for image restoration," *IEEE Transactions on Image Processing*, vol. 10, no. 7, pp. 1001–1009, 2001.

[24] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends in Optimization, 2013*.

[25] D. Krishnan and R. Fergus, "Fast image deconvolution using hyper-laplacian priors," in *Advances in Neural Information Processing Systems 22*, 2009, pp. 1033–1041.

[26] D. Krishnan, R. Fattal, and R. Szeliski, "Efficient preconditioning of laplacian matrices for computer graphics," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 142:1–142:15, Jul. 2013. [Online]. Available: http://doi.acm.org/10.1145/2461912.2461992

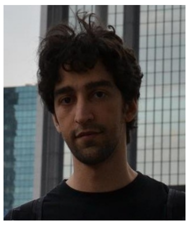
[27] K. Subr, C. Soler, and F. Durand, "Edge-preserving multiscale image decomposition based on local extrema," in *ACM SIGGRAPH Asia 2009 papers*, ser. SIGGRAPH Asia '09. New York, NY, USA: ACM, 2009, pp. 147:1–147:9. [Online]. Available: http://doi.acm.org/10.1145/1661412.1618493

[28] K. He, J. Sun, and X. Tang, "Guided image filtering," in *Proceedings of the 11th European Conference on Computer Vision: Part I*, ser. ECCV'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 1–14. [Online]. Available: http://dl.acm.org/citation.cfm?id=1886063.1886065

[29] R. Fattal, "Edge-avoiding wavelets and their applications," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 1–10, 2009.

[30] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2003.

[31] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679–698, Jun. 1986. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.1986.4767851



Hicham Badri received the Master degree in computer sicence and telecomunications from Mohammed V University Agdal - Faculty of Science in 2012, Rabat - Morocco. He is currently pursuing his Ph.D. in co-supervision with INRIA Bordeaux Sud-Ouest, France and Mohammed V-Agdal University - LRIT, Associated Unit to CNRST (URAC 29). His reserach interests include image processing, sparse methods and multiscale methods.

Hussein Yahia is the head of INRIA research team GEOSTAT (Geometry and statistics in acquisition data). Dr. H. Yahia received the Doctorat ode 3eme cycle from University Paris 11 (Orsay) and the HDR (Habilitation Diriger des Recherches) form University Paris 13. He is specialized in non-linear signal processing and the analysis of complex signals and systems using advanced nonlinear physics. Dr. H. Yahia has also made substantial contributions in Computer Graphics and Image Processing. Dr. H. Yahia is developing strong collaboration with the LEGOS Laboratory in Toulouse (UMR CNRS 55 66) as well ICM-CSIC in Barcelona, and also IIT Roorkee in India (Associated team OPTIC with Prof. Dharmendra Singh). He is also involved in many contract with the French spatial agency CNES and the European Spatial Agency (ESA). Dr. H. Yahia is the author or co-author of about 80 publications in international peer-reviewed journal and conferences including top-ranked conferences such as ACM SIGGRAPH, CVPR and ECCV. He is a member of the editorial board of journals in signal processing or complex systems and has been supervising more than 10 PhD students.

Driss Aboutajdine received the Doctorat de 3me Cycle and the Doctorat d'Etat-es-Sciences degrees in signal processing from the Mohammed V-Agdal University, Rabat, Morocco, in 1980 and 1985, respectively. He joined Mohammed V-Agdal University, Rabat, Morocco, in 1978, first as an assistant professor, then as an associate professor in 1985, and full Professor since 1990. Over 35 years, he developed teaching and research activities covering various topics of signal and image processing, wireless communication and pattern recognition which allow him to advise more than 50 Ph.D. theses and publish over 200 journal papers and conference communications. He succeeded to found and manage since 1993 and 2001 respectively the LRIT Laboratory and the Centre of Excellence of Information & Communication Technology (pole de competences STIC) which gathers more than 30 research laboratories from all Moroccan universities and including industrial partners as well. Prof. Aboutajdine has organized and chaired several international conferences and workshops. He was elected member of the Moroccan Hassan II Academy of Science and technology on 2006 and fellow of the TWAS academy of sciences on 2007. As an IEEE Senior Member, he co-founded the IEEE Morocco Section in 2005 and he is chairing the Signal Processing chapter he founded in December 2010. He is currently the director of the CNRST research center in Morocco, Rabat.

Appendix

Study of Convergence

We study here the convergence properties of the solver when a first-order approximation is introduced on the sparse prior. Suppose the following general minimization problem :

$$\text{minimize}\, f(x) + g(x), \tag{25}$$

where f represents the data fitting term and g is the sparse gradient prior. We want to verify the optimality condition :

$$0 \in \partial f(x) + \partial g(x). \tag{26}$$

To do so, we first start by introducing the intermediate variable v for half-quadratic splitting. For fixed x and v, we have :

$$\begin{aligned} v &= \underset{v}{\operatorname{argmin}}\, g(v) + \tfrac{\beta}{2}||x-v||_2^2 \\ x &= \underset{x}{\operatorname{argmin}}\, f(x) + \tfrac{\beta}{2}||x-v||_2^2, \end{aligned} \tag{27}$$

which results in the following intermediate solutions

$$v = (I + \frac{1}{\beta}\partial g)^{-1}x \quad , \quad x = (I + \frac{1}{\beta}\partial f)^{-1}v. \tag{28}$$

By introducing the first order approximation on g (eq 8), we get :

$$v = (I - \frac{1}{\beta}\partial g)x \quad , \quad x = (I + \frac{1}{\beta}\partial f)^{-1}v. \tag{29}$$

Now by adding the two equations together, it is easy to see that the optimality condition (eq (26)) is verified. This means that convergence is guaranteed if the proximal operator associated to the sparse prior is linearized. Now the difference is that, in the convex case (l_1-norm), the linearization corresponds to the exact solution of the proximal operator. That is, we have :

$$(I + \frac{1}{\beta}\partial g)^{-1}x = (I - \frac{1}{\beta}\partial g)x. \tag{30}$$

As a result, the minimum reached is a global optimum of the main problem (25). However, in the non-convex case, the linearized proximal operator does not correspond exactly to that solution. As a result, the minimum reached by the solver is local.

First Order Derivation

The first order solution of the proximal operator is derived as follows : instead of minimizing the exact proximal operator that is defined as :

$$\mathbf{prox}_{th}(x) = \underset{y}{\text{argmin}} \left\{ h(y) + \frac{1}{2t}||y - x||_2^2 \right\}, \tag{31}$$

we rather minimize the linearized version that is tractable. It consists in replacing $h(y)$ by its first-order approximation $h(y) \approx h(x) + \nabla h(x)^T(y - x)$ in the energy formulation :

$$\underset{y}{\text{argmin}} \left\{ h(x) + \nabla h(x)^T(y - x) + \frac{1}{2t}||y - x||_2^2 \right\}. \tag{32}$$

This problem is equivalent to :

$$\underset{y}{\text{argmin}} \left\{ \underbrace{\nabla h(x)^T y + \frac{1}{2t}||y - x||_2^2}_{E} \right\}. \tag{33}$$

Now applying Euler-Lagrange equation we get :

$$\nabla E = \nabla h(x) + \frac{1}{2t}(2y - 2x) = 0. \tag{34}$$

Finally, we get the first order approximation closed-form of equation (8) :

$$\mathbf{prox}_{th}(x) \approx x - t\nabla h(x). \tag{35}$$

First Order Shrinkage Derivation

We discuss the derivation process from the sparse prior formulation to the shrinkage formula. For the sake of simplicity, we consider the Cauchy case with $\alpha = 2$, the same demonstration holds for other functions. The sparsity-inducing function is defined as follows :

$$\psi(x) = \frac{\gamma^2}{2}\log\left(1 + (\frac{x}{\gamma})^2\right). \tag{36}$$

The *influence function* is defined as follows :

$$\psi(x)' = \frac{\partial\psi(x)}{\partial x} = \frac{x}{1 + (x/\gamma)^2}, \tag{37}$$

and the *weight function* is defined as follows :

$$w(x) = \frac{\psi(x)'}{x} = \frac{1}{1 + (x/\gamma)^2}, \tag{38}$$

which is the same defined in eq (11). Now, replacing x by the gradient at pixel p and iteration k defined as $\nabla u_p^{(k)}$ and ∇h by the influence function $\psi(x)'$ in the first order approximation eq (8), we get the pixelwise shrinkage operation defined in eq (10).