First Order Methods for High Resolution Image Denoising

David Villacís

Abstract—**In this paper we are interested in comparing the performance of some of the most relevant first order non-smooth optimization methods applied to the Rudin, Osher and Fatemi (ROF) Image Denoising Model and a Primal-Dual Chambolle-Pock Image Denoising Model. Because of the properties of the resulting numerical schemes it is possible to handle these computations pixelwise, allowing implementations based on parallel paradigms which are helpful in the context of high resolution imaging.**

Index Terms—**Image Denoising, High Resolution, Parallel Computing, First Order Optimization Methods, Non-smooth optimization methods.**

I. Introduction

Image Denoising is one of the most revisited image analysis tasks, it consists of removing noise from a damaged original image. According to [1, p. 145] noise can be added in an image due to problems in the image adquisition processes such as: malfunctioning pixels in the camera sensors, faulty memory locations in hardware, analog-to-digital conversion, a noisy transmission channel, etc. Traditional techniques used to solve this problem include the use of filters [2], Wavelets [3], and lately in [4] the use of variational models have been popularized due to the elegant way of enforcing properties in the image without the need of single them out explicitly. We are interested in addressing the efficiency of some relevant variational models relying on conjugate duality and saddle point formulations and their application on parallel computing paradigms.
In this work we will consider an image as a $n_1 \times n_2$ pixel matrix, and for simplicity on the mathematical treatment we will map this matrix as a vector $x \in \mathbb{R}^n$, where $n = n_1 \cdot n_2$. Thus a pixel in location (i, j) is the element $x_{i+n_1(j-1)}$ of the vector x.
We will consider the image denoise problem as an inverse problem where the observed image is defined in (1).

$$f = T(x) + \eta, \tag{1}$$

where T is a non-linear operator and η is an additive type of noise. In order to solve this problem we can use a Tikhonov regularization of the problem in (2).

$$f = \phi(x) + \lambda R(x), \tag{2}$$

where ϕ is a data fidelity term that is selected according to the type of noise present in the image and R is called regularizer and promotes certain properties in the solutions obtained.

David Villacís is with the Research Center on Mathematical Modelling (MODEMAT), Escuela Poliécnica Nacional, Quito, Ecuador

II. Image Denoising Models

A. Total Variation Regularization (TV)

The idea of using Total Variation (TV) as a regularizer is to induce sparsity on the gradients of an image, this particular property favours piecewise constant images with sparse edges. Introducing the discrete gradient operator $\mathbb{K} : \mathbb{R}^n \to \mathbb{R}^m$ with $m = 2 \times n$ and $x \in \mathbb{R}^n$ a given image, the discrete Total Variation (TV) is defined by (3).

$$\|\mathbb{K}x\|_{p,1} = \sum_{j=1}^{n} \|(\mathbb{K}x)_j\|_p = \sum_{j=1}^{n} ((\mathbb{K}x)_j^p + (\mathbb{K}x)_{n+j}^p)^{\frac{1}{p}} \tag{3}$$

The p parameter is used to realize anisotropic ($p = 1$) or isotropic ($p = 2$), the latest is usually prefered since it does not exhibit a grind bias.

$$\|\mathbb{K}x\|_{2,1} = \sum_{j=1}^{n} \|(\mathbb{K}x)_j\|_2 = \sum_{j=1}^{n} \sqrt{(\mathbb{K}x)_j^2 + (\mathbb{K}x)_{n+j}^2} \tag{4}$$

Figure 1 shows the impact of using this regularizer on the denoising model.

B. Rudin-Osher-Fatemi (ROF) Model

The TV denoising variational model was first introduced by Rudin, Osher and Fatemi in their seminal paper [5]. They proposed a model that contains a data fidelity term for gaussian noise and a regularizer according to the following model:

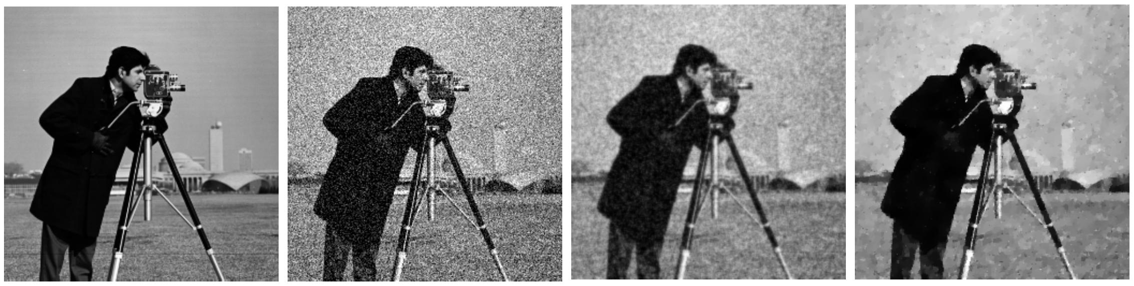
$$\min_{x \in \mathbb{R}^n} \frac{1}{2}\|x - f\|_2^2 + \lambda\|\mathbb{K}x\|_{2,1}, \tag{5}$$

where $f \in \mathbb{R}^n$ is the input corrupted image and λ is the Tichkonov regularization parameter.
In order to handle the regularization term for this model, it is a usual practice to make use of its Fenchel-Rockafellar dual version:

$$\min_{y \in \mathbb{R}^m} \|f - \mathbb{K}^\top y\|_2^2 + \delta_{B_\lambda}(y) \tag{6}$$

Where δ is the indicator function of the λ-radius ball $B_\lambda(0)$, and $y \in \mathbb{R}^m$ is the dual variable.
In order to approach this problem numerically it is helful to use a saddle point formulation of (5):

$$\min_x \max_y \frac{1}{2}\|x - f\|_2^2 + \langle \mathbb{K}x, y\rangle - \delta_{B_\lambda}(y) \tag{7}$$



(a) Original (b) Gaussian Noise (c) $\frac{1}{2}\|x-f\|_2^2+\lambda\|\mathbb{K}x\|_2^2$ (d) $\frac{1}{2}\|x-f\|_2^2+\lambda\|\mathbb{K}x\|_{2,1}$

Fig. 1: Image denoising examples using Forward-Backward method for $\|\mathbb{K}x\|_2^2$ (1c) and $\|\mathbb{K}x\|_{2,1}$ (1d) regularizers. We can see that the use of Total Variation (4) promotes sharper edges inside the image. The original image has been modified with additive gaussian noise with mean 0 and variance 0.2.

C. *TV-l_1 Model*

The denoising task is very sensitive to the noise type appearing in the image, as a matter of fact, the ROF Model presented previously works for gaussian distributed noise as noted in the work presented by Chan et al.[6]. When it comes to impulse distributed noise, seminal work by Nikolova [7] shows that using a l_1 data fidelity term gives us better results in the output image; this effect is illustrated in Figure 2. This model was described as follows:

$$\min_{x\in\mathbb{R}^n}\|x-f\|_1+\lambda\|\mathbb{K}x\|_{2,1}, \tag{8}$$

where $f\in\mathbb{R}^n$ is the input corrupted image and λ is the Tichkonov regularization parameter. It is not hard to see that this model contains two non-differentiable terms, which yields a harder problem to be solved. In order to derive numerical methods for this model, let us formulate the corresponding saddle-point problem:

$$\min_x\max_y\|x-f\|_1+\langle\mathbb{K}x,y\rangle-\delta_{B_\lambda}(y) \tag{9}$$

III. Numerical Treatment

In this section we will present the numerical methods used to solve (5) and (8) along with their corresponding algorithms. Several analytical tools such as Conjugate Duality and Proximal Operators will be used to derive formulations more suitable for a parallel computation environment.

A. *ROF Model*

1) Forward-Backward Splitting: The conjugate dual ROF Model (6) presented in section II has a particular structure that can be exploited, it is the smoothness of the data fidelity term $\|f-\mathbb{K}^\top y\|_2^2$, which in this case is differentiable. Therefore, we can use several splitting numerical mechanisms to find the solution. Let us first analyze the Forward-Backward splitting. Taking in consideration this specific model, $f(y)=\delta_{B_\lambda}(y)$ the characteristic function of the λ-ball, and ∂f the convex subdifferential of f we can describe the optimality condition as:

$$\begin{aligned}0&\in\nabla g(x)+\partial f(x),\\0&\in\mathbb{K}(f-\mathbb{K}^\top y)+\partial f(y),\\\partial f(y)&\ni-\mathbb{K}(f-\mathbb{K}^\top y),\\\tau^{-1}y+\partial f(y)&\ni\tau^{-1}y-\mathbb{K}(f-\mathbb{K}^\top y),\\y&=(I+\tau\partial f)^{-1}(y-\tau\mathbb{K}(f-\mathbb{K}^\top y)).\end{aligned}$$

We will name $(I+\tau\partial f)^{-1}$ as $prox_{\tau\partial f}$, the proximal operator for f. The proximal operator provides the unique optimizer of a Moreau-Yosida (MY) regularization of a function f:

$$f_{MY}(x)=f(x)+\frac{1}{2\tau}\|\bar{x}-x\|_2^2. \tag{10}$$

In [8] it is shown that (10) has a unique minimizer and the resulting function is proper, lower-semicontinuous and strongly convex. Moreover, its minimizer solves the following optimization problem:

$$prox_{\tau\partial f}(x)=\underset{\bar{x}}{\operatorname{argmin}}\,f(x)+\frac{1}{2\tau}\|\bar{x}-x\|_2^2 \tag{11}$$

Leading to the following numerical iteration:

$$y_{k+1}=prox_{\tau\partial f}(y_k-\tau\mathbb{K}(f-\mathbb{K}^\top y_k)). \tag{12}$$

This iteration requires the calculation of the proximal operator of $\delta_{B_\lambda}(y)$, for this application in particular this operator corresponds to the pixel-wise projection of y onto the λ-radius ball.

$$[proj_{B_\lambda}(y)]_i=\frac{y_i}{\max\{1,\lambda^{-1}\|y\|_2\}},\ \forall i=1,\ldots,n \tag{13}$$

Now, working on the dual optimization problem exclusively we can make use of Algorithm 1 to obtain the optimal dual value and retrieve its primal $x_k=f-\mathbb{K}^\top y_k$.

2) Chambolle-Pock Method: This method handles the saddle point formulation of the problem (7). We can find in this formulation that it contains a primal part and a dual part. To tackle this problem this method proposes two proximal steps, one for the primal part and one for the dual part, and an interpolation step that in the case of the ROF model yields the numerical scheme presented in Algorithm 2.

As a preliminary experiment, in Figure 3 we can see the convergence properties of the two methods presented in this section. Indeed we can see that the Chambolle-Pock method

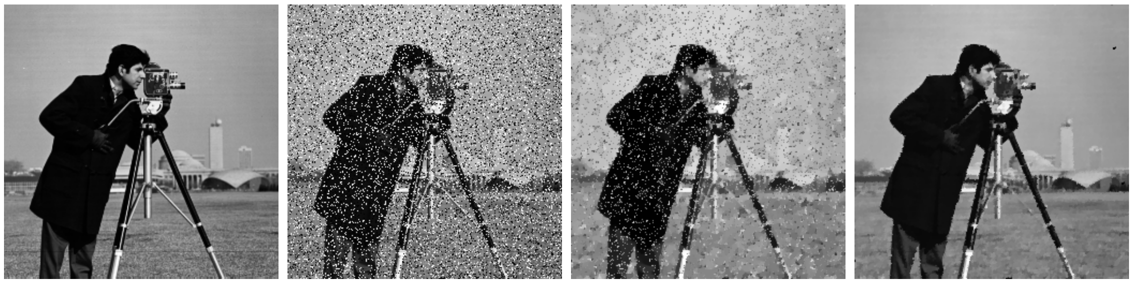


Fig. 2: Image denoising examples using Chambolle-Pock method for ROF Model (2c) and TV-l_1 model (2d) regularizers with $\lambda = 0.2$. We can see that the ROF model cannot tackle the image denoising task appropiately when it is presented with impulse noise.

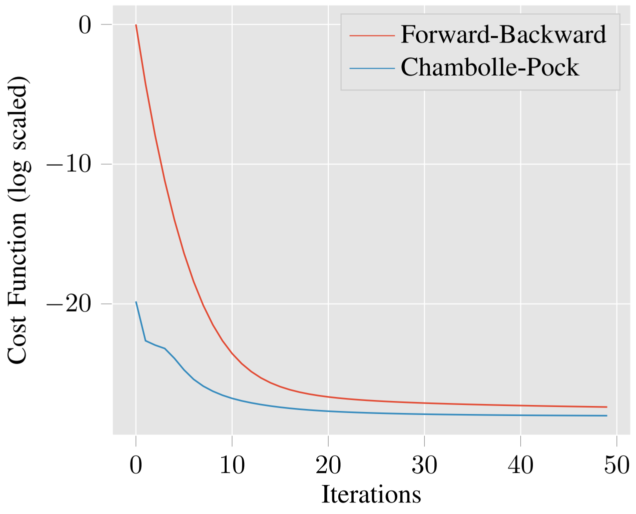


Fig. 3: ROF Cost Function evolution for Forward-Backward and Chambolle-Pock Methods.

Algorithm 1 Forward-Backward Splitting for ROF Denoising

```
1: Choose $\lambda > 0$, $\tau > 0$, $k = 0$.
2: Set number of iterations $niter$.
3: while $k < niter$ do
4:     Calculate $\nabla g(y_k) = \mathbb{K}(f - \tau \mathbb{K}^\top y_k)$
5:     Perform projection onto the $\lambda$-ball
           $y_{k+1} = proj_{B_\lambda}(y_k - \tau \nabla g(y_k))$
6:     $k = k + 1$
7: end while
8: Retrieve primal value $x_{k+1} = f - \mathbb{K}^\top y_{k+1}$.
```

Algorithm 2 Chambolle-Pock Method for ROF Denoising

```
1: Choose $\lambda > 0$, $\tau, \sigma > 0$, $k = 0$.
2: Set number of iterations $niter$.
3: while $k < niter$ do
4:     Calculate the proximal step for the primal function
           $x_{k+1} = \frac{x_k + \tau(f - \mathbb{K}^\top y_k)}{\tau + 1}$
5:     Calculate the interpolation step $\bar{x}_{k+1} = 2x_{i+1} - x_k$
6:     Perform projection onto the $\lambda$-ball
           $y_{k+1} = proj_{B_\lambda}(y_k + \sigma \mathbb{K} \bar{x}_{k+1})$
7:     $k = k + 1$
8: end while
```

presents better convergence properties thanks to the use of the extra information provided by the proximal operator.

B. *TV-l_1 Model*

1) Chambolle-Pock Method: For this model we will make use of a saddle point formulation of the TV-l_1 model (9). In order to apply the Chambolle-Pock Method we need to derive the form of the proximal operator for $g(x) = \|x - f\|_1$.

$$prox_{\tau \partial g} = \underset{\bar{x}}{\operatorname{argmin}} \|x - f\|_1 + \frac{1}{2\tau}\|\bar{x} - x\|^2. \tag{14}$$

which, using the procedure explained in Appendix A, yields:

$$\bar{x}_i = sign(x_i - f_i)\max(|x_i - f_i| - \tau, 0),\ \forall i = 1, \ldots, n. \tag{15}$$

Therefore, we can make use of the numerical scheme presented in Algorithm 3. Let us remark the fact that every operation is perfomed pixel-wise.

Algorithm 3 Chambolle-Pock Method for TV-l_1 Denoising

```
1: Choose $\lambda > 0$, $\tau, \sigma > 0$, $k = 0$.
2: Set number of iterations $niter$.
3: while $k < niter$ do
4:     Calculate the auxiliary value: $\hat{x}_{k+1} = x_k - \tau(\mathbb{K}^\top y_k)$
5:     Calculate the proximal step for the primal function
           $x_{k+1} = f + sign(\hat{x}_{k+1} - f)\max(|\hat{x}_{k+1} - f| - \tau, 0)$
6:     Calculate the interpolation step $\bar{x}_{k+1} = 2x_{k+1} - x_k$
7:     Perform projection onto the $\lambda$-ball
           $y_{k+1} = proj_{B_\lambda}(y_k + \sigma\mathbb{K}\bar{x}_{k+1})$
8:     $k = k + 1$
9: end while
```

IV. Numerical Experiments

For the numerical experiments presented in this work we will be using both a set testing images with synthetic noise and the dataset provided in [9], this dataset provides real noise generated for high resolution images of playing cards. This dataset was obtained using a PhaseOne XF medium format camera equipped with an achromatic IQ260 digital back and a PhaseOne Digita AF 120mm F4 lens. The equipment used generated 16 bit TIFF images of 8964x6716 pixels. The images were shot with an ISO of 3200 and a histogram approximately spanning a quarter of the full dynamic range, yielding an image that contains noise generated from round-off errors from the digital-analog conversor, photon counting noise and electronic noise.

Since all operations for the numerical methods described previously are applied pixel-wise, we can make use of parallel computation efectively. In this work, we used CUDA parallel programming [10] to implement the update for the numerical schemes presented for the chambolle-pock method for both ROF and TV-l_1 denoising problems.
For the CPU computations, all methods described above were implemented using python Numpy [11] numerical libraries, and all the GPU computations were coded using PyCUDA [12] with kernels written in C.

All the presented experiments where executed in the Mode-Mat HPC Cluster, this facility provided us with Xeon Phi processors for the CPU computations and NVIDIA Tesla K80 GPU coprocessors for the GPU computations.

A. ROF Model

We tested the Forward-Backward (FB) and Chambolle-Pock (CP) primal-dual methods against a set of images: Circle (106x106 px), Cameraman (256x256 px) and Lena (512x512 px) using both the CPU and GPU implementations. The results are presented in Table I, this synthetic images presented additive gaussian noise with 0 mean and 0.2 variance. In Table II the results over the real noise dataset with the following set of images: Playing Cards 1 (1280x720 px - HD Resolution), Playing Cards 2 (1920x1080 px - FullHD Resolution), Playing Cards 3 (7680x4320 px - UltraHD Resolution).
For the experiments, in the synthetic dataset we used 100 runs, the mean and standard deviation values are reported in the corresponding tables. In the case of the real noise dataset we used 10 trials with the same values reported.

TABLE I: ROF Model Processing Time

Image	FB CPU	CP CPU	CP GPU
	time (s)	time (s)	time (s)
Circle	0.0877 ± 0.008	0.0791 ± 0.001	$\mathbf{0.0261 \pm 0.004}$
Cameraman	0.4499 ± 0.008	0.4064 ± 0.001	$\mathbf{0.0269 \pm 0.001}$
Lena	1.7019 ± 0.007	1.6732 ± 0.069	$\mathbf{0.0358 \pm 0.001}$

TABLE II: ROF Model Processing Time

Image	FB CPU	CP CPU	CP GPU
	time (s)	time (s)	time (s)
Playing Cards 1	7.89 ± 0.12	7.26 ± 0.02	$\mathbf{0.067 \pm 0.01}$
Playing Cards 2	24.40 ± 0.06	21.60 ± 0.16	$\mathbf{0.118 \pm 0.01}$
Playing Cards 3	585.04 ± 0.98	511.36 ± 0.81	$\mathbf{1.717 \pm 0.01}$

TABLE III: TV-l_1 Model Processing Time - Synthetic Noise

Image	CP CPU	CP GPU
Circle	0.094254 ± 0.000323	$\mathbf{0.025320 \pm 0.000736}$
Cameraman	0.456685 ± 0.000841	$\mathbf{0.026760 \pm 0.000779}$
Lena	1.802704 ± 0.003430	$\mathbf{0.037916 \pm 0.000587}$

TABLE IV: TV-l_1 Model Processing Time - Real Noise

Image	CP CPU	CP GPU
	time (s)	time(s)
Playing Cards 1	8.23238 ± 0.030	$\mathbf{0.067131 \pm 0.001167}$
Playing Cards 2	24.16407 ± 0.051	$\mathbf{0.128384 \pm 0.007352}$
Playing Cards 3	599.70860 ± 0.754	$\mathbf{1.895479 \pm 0.003539}$

TABLE V: ROF Model Processing Time Low CPU

Image	FB CPU LOW	CP CPU LOW	CP GPU
Circle	0.180 ± 0.02	0.139 ± 0.01	$\mathbf{0.0261 \pm 0.004}$
Cameraman	1.395 ± 0.10	0.872 ± 0.05	$\mathbf{0.0269 \pm 0.001}$
Lena	5.971 ± 0.01	5.358 ± 0.1	$\mathbf{0.0358 \pm 0.001}$

TABLE VI: TV-l_1 Model Processing Time Low CPU

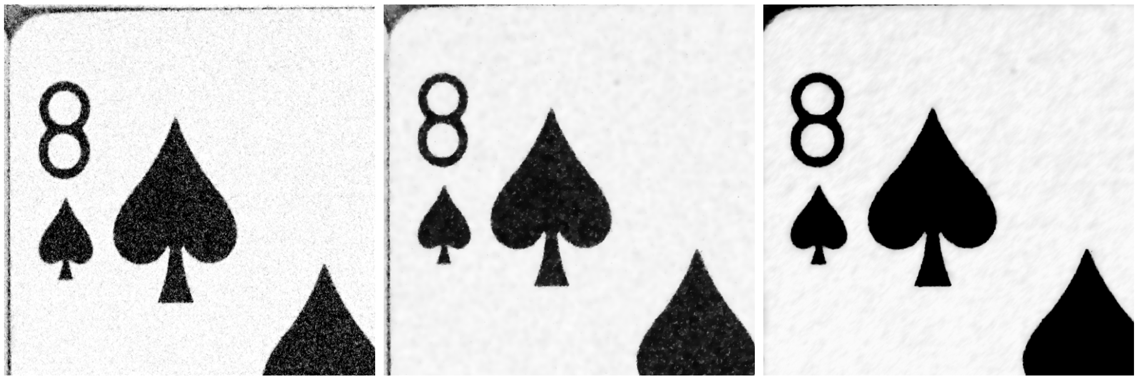
Image	CP CPU LOW	CP GPU
Circle	0.16 ± 0.01	$\mathbf{0.025 \pm 0.001}$
Cameraman	0.935 ± 0.3	$\mathbf{0.027 \pm 0.001}$
Lena	6.06 ± 0.1	$\mathbf{0.038 \pm 0.001}$

B. TV-l_1 Model

In Table III and Table IV we tested Chambolle-Pock (CP) CPU and GPU implementations against the same set of images described in the previous experiment. The synthetic images generated presented additive impulse noise.
For comparison purpouses in Tables V and Table VI the results of the paralell version of the algorithms is compared with the performance using a desktop computer using a traditional Intel Core i5 processor.

V. Conclusion

In this work we can see that the ROF and TV-l_1 image denoising models can be formulated using a saddle point formulation. This formulation allows Chambolle-Pock method



(a) Original (b) CUDA ROF Denoised (c) CUDA TV-l_1 Denoised

Fig. 4: Image denoising examples using a CUDA Implementation Chambolle-Pock method for ROF Model (4b) and TV-l_1 model (4c) regularizers with $\lambda = 0.2$. This image patch corresponds to an UltraHD Resolution Image, the execution time for the algorithm was 1.89548 seconds.

to implement an algorithm to find a solution that presents some separability properies, since the proximal operators obtained can be implemented pixelwise. Therefore, we can make use of parallel computation. In particular we used CUDA parallel computation which yields dramatic speedups for finding solutions when compared with pure serial implementations.

Appendix A $\|x-f\|_1$ Proximal Operator

We can formulate the proximal operator as:

$$\operatorname*{argmin}_{\bar{x}} \|x-f\|_1 + \frac{1}{2\tau}\|\bar{x}-x\| \tag{16}$$

We know that $\bar{x}$ is a minimizer argument of (16) if and only if it satisfies the following optimization criteria:

$$\begin{aligned} 0 &\in \partial\|\bar{x}-f\|_1 + (\bar{x}-x), \\ x &\in \tau\partial\|\bar{x}-f\|_1 + \bar{x} \end{aligned}$$

We know that the l_1-norm is separable, therefore we can analyze this equation component-wise. Let us take the case $\bar{x}_i \neq 0$ then $\partial\|\bar{x}_i - f_i\| = sign(\bar{x}_i - f_i)$, then:

$$\begin{aligned} x_i &= \tau sign(\bar{x}_i - f_i) + x_i, \\ \bar{x}_i &= x_i - \tau sign(\bar{x}_i - f_i), \\ \bar{x}_i - f_i &= x_i - f_i - \tau sign(\bar{x}_i - f_i) \end{aligned}$$

If $\bar{x}_i - f_i > 0$ then $x_i - f_i < -\tau$ and $\bar{x}_i - f_i < 0$ then $x_i - f_i > \tau$. Hence, $|x_i - f_i| > \tau$ and $sign(\bar{x}_i - f_i) = sign(x_i - f_i)$. If $\bar{x}_i - f_i = 0$ then,

$$\begin{aligned} 0 &\in f_i - x_i + \tau[-1,1], \\ x_i - f_i &\in [-\tau, \tau], \\ |x_i - f_i| &\leq \tau \end{aligned}$$

Therefore, the proximal operator can be written as

$$\begin{aligned} prox_{\tau\partial g}(x) &= \bar{x}_i - f_i, \\ &= \begin{cases} 0 & \text{if } |x_i - f_i| \leq \tau, \\ x_i - f_i - \tau sign(x_i - f_i) & \text{if } |x_i - f_i| > \tau \end{cases} \end{aligned}$$

This condition can be written in a more compact form:

$$prox_{\tau\partial g}(x) = \max\{|x_i - f_i| - \tau, 0\} sign(x_i - f_i) \tag{17}$$

References

[1] T. Chan and J. Shen, *Image Processing And Analysis: Variational, Pde, Wavelet, And Stochastic Methods*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2005.

[2] N. Wiener, *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. The MIT Press, 1964.

[3] A. Chambolle, R. A. DeVore, N. Y. Lee, and B. J. Lucier, "Nonlinear wavelet image processing: Variational problems, compression, and noise removal through wavelet shrinkage," *IEEE Transactions on Image Processing*, vol. 7, no. 3, pp. 319–335, 1998.

[4] L. I. Rudin and S. Osher, "Total variation based image restoration with free local constraints," *Image Processing, 1994. Proceedings. ICIP-94., IEEE International Conference*, vol. 60, 1994.

[5] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1-4, pp. 259–268, 1992.

[6] R. H. Chan, C.-W. Ho, and M. Nikolova, "Salt and Pepper Noise Removal by Median Type Noise Detectors and Detail-Preserving Regularization," *IEEE Transactions on Image processing*, vol. 14, no. 10, pp. 1479–1485, 2005.

[7] M. Nikolova, "A Variational Approach to Remove Outliers and Impulse Noise," in *Journal of Mathematical Imaging and Vision*, vol. 20, no. 1-2, 2004, pp. 99–120.

[8] J. Peypouquet, *Convex Optimization in Normed Spaces*. Springer, 2015.

[9] S. K. S. S. Villacis David, Helenius Teemu, "Photographic dataset: playing cards," 2017.

[10] T. M. John Cheng, Max Grossman, *Professional CUDA C Programming*, 1st ed. Wrox, 2014.

[11] S. C. C. Stfan van der Walt and G. Varoquaux, "Numpy: Open source numerical tools for Python," 2011. [Online]. Available: http://aip.scitation.org/doi/abs/10.1109/MCSE.2011.37

[12] Y. L. B. C. P. I. A. F. Andreas Klckner, Nicolas Pinto, "Pycuda and pyopencl: A scripting-based approach to gpu run-time code generation," 2009. [Online]. Available: https://arxiv.org/abs/0911.3456

David Villacís is a PhD student in Applied Mathematics from Escuela Politécnica Nacional. M.Sc. in Computer Science from University of Birmingham, UK. Currently working in variational image processing problems and optimization methods for machine learning.