

Combining Multiple Directions in the Total Variation Function: Application to Image Inpainting Using the L-BFGS Algorithm

Bruno Schelk

Systems Engineering and Computer Science Program
Federal University of Rio de Janeiro
bschelk@cos.ufrj.br

Michel Tcheou

Department of Electronics and Telecommunications Engineering
Rio de Janeiro State University
mtcheou@uerj.br

Lisandro Lovisolo

Department of Electronics and Telecommunications Engineering
Rio de Janeiro State University
lisandro@uerj.br

Laura Bahiense

Systems Engineering and Computer Science Program
Federal University of Rio de Janeiro
laura@cos.ufrj.br

Luidi Simonetti

Systems Engineering and Computer Science Program
Federal University of Rio de Janeiro
luidi@cos.ufrj.br

ABSTRACT

In image processing, several tasks require the replacing of missing or deteriorated pixels in digital images through algorithms capable of estimating plausible values for the damaged regions. Total Variation (TV)-based formulations have been widely used for this purpose, generally relying on horizontal and vertical differences between adjacent pixels. In this work, we propose a new formulation that combines four TV directions into a single objective function, enabling a more expressive representation of local structures to improve the quality of restored images. The resulting unconstrained differentiable nonlinear optimization problem is solved using the limited-memory quasi-Newton (L-BFGS) algorithm. Numerical experiments on diverse image sets and noise types show that our approach outperforms the classical TV formulation in terms of standard image quality metrics such as Peak Signal-to-Noise Ratio (PSNR) and the Structural Similarity Index Metric (SSIM), without increasing computational cost.

KEYWORDS. Unconstrained Differentiable Nonlinear Optimization, Convex Optimization, L-BFGS, Total Variation, Image Inpainting.

Paper topics: ONL – Nonlinear Optimization, OC – Continuous Optimization.

1. Introduction

Image processing is a fundamental area within computer vision and digital signal processing, involving tasks such as the analysis, enhancement, restoration, and transformation of digital images [Minaee et al., 2021]. Image processing techniques are employed to extract meaningful information or to improve image quality for visualization or subsequent analysis [Liu et al., 2022]. These techniques can range from basic operations such as filtering, contrast adjustment, and noise reduction, to advanced methods such as segmentation, edge detection, and object recognition [Chudasama et al., 2014]. As digital imaging technologies evolve, the development of efficient and accurate image processing algorithms becomes increasingly important, particularly those that can handle degraded, incomplete, or noisy visual data [Buades et al., 2005; Chan and Shen, 2005]. These algorithms are essential in several applications, including medical imaging, remote sensing, industrial inspection, autonomous vehicles, and multimedia systems [Jähne, 2005].

In this regard, the image inpainting problem involves the process of reconstructing missing, corrupted, or degraded regions in digital images, with the objective of generating visually consistent content that maintains structural coherence with the surrounding areas [Elharrouss et al., 2020; Guillemot and Le Meur, 2013]. This problem can be applied for various purposes, such as the restoration of old photographs, the removal of unwanted objects or text, and the recovery of lost information due to compression or transmission errors [Bertalmio et al., 2000; Chan and Shen, 2001]. To obtain high-quality results, specialized algorithms are designed to estimate the values of the damaged regions based on information from their adjacent pixels. These algorithms often rely on strong mathematical models and optimization techniques, but they can be computationally demanding. In general, faster inpainting methods tend to yield less accurate reconstructions, whereas more sophisticated models achieve higher fidelity at the expense of increased computational complexity [Elharrouss et al., 2020; Guillemot and Le Meur, 2013; Dahl et al., 2010].

Among the various techniques explored in the literature, methods based on Total Variation (TV) regularization have demonstrated strong performance in image restoration tasks [Chambolle et al., 2010]. Initially introduced by Rudin et al. [1992] for image denoising, the TV framework exploits the assumption that natural images exhibit smoothness between adjacent pixels and estimates missing values by minimizing local intensity differences. Subsequently, TV-based models were successfully adapted to the image inpainting problem [Chan and Shen, 2001; Afonso et al., 2010], and further improvements were proposed by incorporating low-rank approximations and other regularization strategies [Han et al., 2014; Dong et al., 2012a].

In traditional TV-based inpainting formulations, the objective is to minimize the sum of finite differences between adjacent pixels, typically considering only one horizontal and vertical directions. Although effective, this model may have limited capacity to capture the full spatial structure of complex image regions. To address this limitation, in this paper we introduce a novel inpainting formulation that combines TV terms computed over distinct pixel directions. While the classical TV formulation computes finite differences using (below, right) adjacent pixels, we extend this approach by introducing three additional ones: (right, above), (above, left), and (left, below). This broader neighborhood structure allows the model to better exploit directional features and textures that are often present in natural images.

Our mathematical formulation leads to a differentiable unconstrained nonlinear optimization problem, which we solve using the limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) quasi-Newton algorithm [Bonnans et al., 2006]. This algorithm is particularly effective for large-scale problems in which the number of variables is high, as is typically the case in image processing tasks. We applied the (L-BFGS) algorithm to solve the inpainting problem using the different functions presented in this work, considering various types of noise and sets of images, and evaluated the results using standard image quality metrics.

The paper is organized as follows. Section 2 introduces the mathematical formulation, including the definition of the optimization problem. Section 3 details the algorithm used to solve the problem. In Section 4, we present and analyze the numerical results obtained from several experiments. Finally, Section 5 presents the conclusion of the paper.

2. Mathematical Formulation

A digital image can be represented as a grid of $n_1 \times n_2$ pixels, where n_1 and n_2 denote the number of rows and columns, respectively, and each pixel is denoted by $x_{i,j}$. Let $R \subseteq X$ be a region (continuous or not) of the image to be restored. As stated in the introduction (Section 1), the goal of the image inpainting problem is to estimate the values of the pixels in R in such a way that the resulting image retains acceptable quality and visual consistency. To guide this reconstruction, one commonly used regularization approach is based on Total Variation (TV), which quantifies local intensity variations by measuring the differences between adjacent pixels. Formally, the total variation of an image X over a region R is defined as [Rodríguez, 2013]:

$$\text{TV}(X, R) = \sum_{(i,j) \in \Lambda_R} \sqrt{\|x_{i+1,j} - x_{i,j}\|_2^2 + \|x_{i,j+1} - x_{i,j}\|_2^2}, \quad (1)$$

which calculates the differences between each pixel $x_{i,j}$ and its below and right adjacent pixels, for all pixels in the region R . The set Λ_R contains the indices (i, j) of the pixels inside the region R .

The TV function in equation (1) is nonlinear, convex, and non-differentiable at the origin [Chambolle et al., 2010]. Non-differentiability generally makes it difficult to apply fast algorithms to the optimization process, i.e., to find its minimum. To address this, we adopt a smoothed version of the TV function that not only approximates (1) but also enables the use of derivative-based optimization methods. This smoothing procedure is known as the Linear Total Variation Approximation (LTVA) [Han et al., 2014]:

$$\text{LTVA}(X, R) = \sum_{(i,j) \in \Lambda_R} \|x_{i+1,j} - x_{i,j}\|_2^2 + \|x_{i,j+1} - x_{i,j}\|_2^2. \quad (2)$$

Pixels inside a given image structure tend to be similar [Gonzalez, 1984], resulting in minimal differences between adjacent pixels. Since the total variation (TV) is computed as the sum of these differences, its value is expected to be small. This assumption underlies the use of TV for image inpainting, in which pixels in region R are iteratively restored by solving the following optimization problem:

$$\min_{X \in \mathbb{R}^{n_1 \times n_2}} f(X, R), \quad f(X, R) := \text{a variation measure of } X \text{ in } R. \quad (3)$$

In the inpainting problem described in (3), the TV and LTVA functions can be used, as defined in (1) and (2), respectively, as variation measures in the image X over region R . In both functions, differences are computed between each pixel and its immediately adjacent pixels to the right and below. In contrast, this work generalizes the notion of pixel differences by considering additional neighboring relationships, referred to here as directional differences or simply directions. Based on this, we define the function

$$\text{LTVA}_l(X, R) := \sum_{(i,j) \in \Lambda_R} \|\nabla_l x_{i,j}\|_2^2, \quad (4)$$

where $\nabla_l x_{i,j}$ is the two-dimensional vector with the finite difference approximations of the first-order horizontal and vertical partial derivatives for the pixel of coordinates (i, j) at direction l .

For $\nabla_l x_{i,j}$, we set four directions ($l = 1, 2, 3, 4$) as

$$\begin{aligned} \nabla_1 x_{i,j} &= \begin{bmatrix} x_{i+1,j} - x_{i,j} \\ x_{i,j+1} - x_{i,j} \end{bmatrix}, & \nabla_2 x_{i,j} &= \begin{bmatrix} x_{i,j+1} - x_{i,j} \\ x_{i-1,j} - x_{i,j} \end{bmatrix}, \\ \nabla_3 x_{i,j} &= \begin{bmatrix} x_{i-1,j} - x_{i,j} \\ x_{i,j-1} - x_{i,j} \end{bmatrix}, & \nabla_4 x_{i,j} &= \begin{bmatrix} x_{i,j-1} - x_{i,j} \\ x_{i+1,j} - x_{i,j} \end{bmatrix}, \end{aligned} \quad (5)$$

in which $\nabla_1 x_{i,j}$ computes the difference between adjacent pixels (below, right); $\nabla_2 x_{i,j}$ (right, above); $\nabla_3 x_{i,j}$ (above, left); and $\nabla_4 x_{i,j}$ (left, below). We can also interpret (5) as the same directional operation applied to the image X after rotating it by 90 degrees. In practice, in (4) we incorporate each TV direction from (5) separately into function (2).

We also choose to simultaneously consider the different TV directions defined in (5) within the problem in (3). Thus, we combine them into a single function, called the Sum of Linear Total Variation Approximation (SLTVA), which is defined and simplified as follows:

$$\begin{aligned} \text{SLTVA}(X, R) &= \frac{1}{4} \sum_{l=1}^4 \text{LTVA}_l(X, R) \\ &= \frac{1}{2} \sum_{(i,j) \in \Lambda_R} \|x_{i+1,j} - x_{i,j}\|_2^2 + \|x_{i,j+1} - x_{i,j}\|_2^2 + \|x_{i-1,j} - x_{i,j}\|_2^2 + \|x_{i,j-1} - x_{i,j}\|_2^2. \end{aligned} \quad (6)$$

If function (6) outperforms function (4) in terms of image quality for some image processing problem, the likely rationale is that it leverages a greater amount of information from the image compared to (4). Therefore, we also examine a function that extends the pixel neighborhood to include all eight adjacent pixels, herein referred to as the Alternative Linear Total Variation Approximation (ALTVA), defined as:

$$\begin{aligned} \text{ALTVA}(X, R) &:= \frac{1}{2} \sum_{(i,j) \in \Lambda_R} \|x_{i+1,j} - x_{i,j}\|_2^2 + \|x_{i,j+1} - x_{i,j}\|_2^2 + \|x_{i-1,j} - x_{i,j}\|_2^2 + \|x_{i,j-1} - x_{i,j}\|_2^2 \\ &\quad + \|x_{i+1,j+1} - x_{i,j}\|_2^2 + \|x_{i-1,j+1} - x_{i,j}\|_2^2 + \|x_{i-1,j-1} - x_{i,j}\|_2^2 + \|x_{i+1,j-1} - x_{i,j}\|_2^2. \end{aligned} \quad (7)$$

An important observation is that the image X can be composed of three planes corresponding to the RGB color channels. In this case, each coordinate x_{ij} is a three-dimensional vector. In practice, all computations are carried out simultaneously across the three color planes, but each plane computes only within itself.

3. L-BFGS Algorithm

A solution to problem (3) can be obtained using an unconstrained optimization algorithm based on an iterative scheme in the form

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} d^{(k)}, \text{ with} \quad (8)$$

$$d^{(k)} = -Q^{(k)} \nabla f(x^{(k)}), \quad (9)$$

where $\alpha^{(k)} > 0$ is a step size, $\nabla f(x^{(k)})$ is the gradient vector of function f and $d^{(k)}$ is the descent direction.

In the scheme described in (8)–(9), when $Q^{(k)}$ corresponds to the inverse of the Hessian matrix, the method is known as Newton's method [Polyak, 2007]. However, computing the exact inverse of the Hessian is often computationally expensive. To address this, Quasi-Newton methods have been developed, which iteratively build approximations of the inverse of the Hessian matrix. Two of the most well-known approaches in this class are Davidon–Fletcher–Powell (DFP) [Davidon, 1959; Fletcher and Powell, 1963] and Broyden–Fletcher–Goldfarb–Shanno (BFGS) [Shanno, 1970; Goldfarb, 1970; Fletcher, 1970; BROYDEN, 1970].

Even for small-sized images, problem (3) may involve millions of variables, and although an approximate inverse of the Hessian matrix can be computed, its storage becomes impractical in large-scale settings due to memory limitations. Therefore, we adopt the Limited-Memory Quasi-Newton method (L-BFGS) [Liu and Nocedal, 1989] to solve (3), which provides an efficient trade-off between accuracy and memory usage.

The L-BFGS is an optimization algorithm commonly used to minimize smooth nonlinear functions, particularly in large-scale problems. It is a limited-memory variant of the quasi-Newton BFGS method, designed to reduce memory usage by storing only a few vectors from previous iterations. L-BFGS belongs to the class of iterative optimization methods. It attempts to find the minimum point of a function $f(x)$ using only evaluations of the function and its gradient $\nabla f(x)$, without requiring the full Hessian matrix. This makes it especially efficient for problems involving a large number of variables.

The scheme defined by equations (8) and (9), applied to the inpainting problem, leads to Algorithm 1. This algorithm calls Algorithm 2 to determine the descent direction $d^{(k)}$ [Bonnans et al., 2006; Liu and Nocedal, 1989] and Algorithm 3 to compute the step size $\alpha^{(k)}$ calculated through the Wolfe rule [Wolfe, 1969, 1971].

Algorithm 1 Image inpainting using L-BFGS.

Input: $x^{(0)}$ (corrupted image), integer $m > 0$ and Λ_R (mask for the corrupted region)

```

1:  $k \leftarrow 0$ 
2: repeat
3:   Call Algorithm 2 to compute  $d^{(k)}$ 
4:   Call Algorithm 3 to compute  $\alpha^{(k)}$ 
5:    $x^{(k+1)} \leftarrow x^{(k)} + \alpha^{(k)} d^{(k)}$ 
6:   if  $k > m$  then
7:     Discard the vector pair  $\{s^{(k-m)}, y^{(k-m)}\}$  from storage
8:   end if
9:   Compute and save  $s^{(k)} \leftarrow x^{(k+1)} - x^{(k)}$ ,  $y^{(k)} \leftarrow \nabla f(x^{(k+1)}) - \nabla f(x^{(k)})$ 
10:   $k \leftarrow k + 1$ 
11: until  $\|\nabla f(x^{(k)})\| < \epsilon$ 
Output:  $\hat{X} = x^{(k+1)}$  (the restored image)

```

Algorithm 2 Computation of the descent direction.

Input: Positive definite symmetric matrix $Q^{(0)}$, k (iterations executed in Algorithm 1), gradient vector $\nabla f(x^{(k)})$, and $s^{(i)}, y^{(i)}, i = 1, \dots, m$.

```

1:  $r^{(m)} := \nabla f(x^{(k)})$ 
2: for  $i = m, \dots, 1$  do
3:    $\gamma^{(i)} \leftarrow \langle r^{(i)}, s^{(i)} \rangle / \langle y^{(i)}, s^{(i)} \rangle$ 
4:    $r^{(i-1)} \leftarrow r^{(i)} - \gamma^{(i)} y^{(i)}$ 
5: end for
6:  $h^{(0)} := Q^{(0)} r^{(0)}$ 
7: for  $i = 1, \dots, m$  do
8:    $\beta^{(i)} \leftarrow \langle y^{(i)}, h^{(i-1)} \rangle / \langle y^{(i)}, s^{(i)} \rangle$ 
9:    $h^{(i)} \leftarrow h^{(i-1)} + (\gamma^{(i)} - \beta^{(i)}) s^{(i)}$ 
10: end for
Output:  $d^{(k+1)} = -h^{(m)}$  (descent direction)

```

Algorithm 3 Computation of the step size via Wolfe Rules.

Input: $\alpha > 0$, parameters $0 < \sigma_1 < \sigma_2 < 1$

```

1:  $\underline{\alpha} \leftarrow 0$  and  $\bar{\alpha} \leftarrow 0$ 
2: while  $f(x^{(k)} + \alpha d^{(k)}) > [f(x^{(k)}) + \sigma_1 \alpha \langle \nabla f(x^{(k)}), d^{(k)} \rangle]$  and  $\langle \nabla f(x^{(k)} + \alpha d^{(k)}), d^{(k)} \rangle < \sigma_2 \langle \nabla f(x^{(k)}), d^{(k)} \rangle$  do
3:   if  $f(x^{(k)} + \alpha d^{(k)}) > [f(x^{(k)}) + \sigma_1 \alpha \langle \nabla f(x^{(k)}), d^{(k)} \rangle]$  then
4:      $\bar{\alpha} \leftarrow \alpha$  and go to 12
5:   else if  $\langle \nabla f(x^{(k)} + \alpha d^{(k)}), d^{(k)} \rangle < \sigma_2 \langle \nabla f(x^{(k)}), d^{(k)} \rangle$  then
6:      $\underline{\alpha} \leftarrow \alpha$ 
7:   end if
8:   if  $\bar{\alpha} = 0$  then
9:      $\alpha \leftarrow$  any value  $> \underline{\alpha}$  and go to 3
10:  end if
11:   $\alpha \leftarrow$  any value  $\in (\underline{\alpha}, \bar{\alpha})$  and go to 3
12: end while
Output:  $\alpha^{(k)} = \alpha$  (the step size)

```

The use of the L-BFGS method to compute the descent direction $d^{(k)}$ (as described in Algorithm 2) avoids the need to store the Hessian matrix, whether in its exact or approximated form, consequently saving memory. Given a symmetric positive definite matrix $Q^{(k)}$, the descent direction $d^{(k)} = -Q^{(k)} \nabla f(x^{(k)})$ can be calculated using the last pair of vectors m , where m is a predefined parameter. These pair of vectors are given by $s^{(k)} = x^{(k)} - x^{(k-1)}$ and $y^{(k)} = \nabla f(x^{(k)}) - \nabla f(x^{(k-1)})$.

It should be noted that the approximation matrix $Q^{(k)}$ in L-BFGS requires storing only $2mn$ elements, instead of the $n(n+1)/2$ needed by quasi-Newton methods such as DFP or BFGS, where n is the number of variables in the problem. However, as the number of iterations k increases, additional storage would be required to keep all $s^{(k)}$ and $y^{(k)}$ vectors. To address this, L-BFGS retains only the most recent m vector pairs in a last-in-first-out (LIFO) structure, ensuring that memory usage remains within fixed bounds while maintaining computational efficiency throughout the optimization process.

In optimization algorithms based on quasi-Newton methods, the choice of the step size α_k along the descent direction d_k is crucial to ensure both the efficiency and stability of the method. Although classical conditions such as those proposed by Armijo [Armijo, 1966] or Goldstein [Goldstein, 1965] can be employed in line search strategies, in practice, the Wolfe conditions [Wolfe, 1969, 1971] are preferred in these methods.

The motivation for adopting the Wolfe conditions is that quasi-Newton methods construct an approximation of the Hessian matrix (or its inverse) so that it remains positive definite, which requires $s_k^\top y_k > 0$. This condition is not always guaranteed when using the Armijo rule alone, since it focuses exclusively on sufficient decrease in the objective function. As a result, the Armijo rule may accept excessively small values of α_k , which reduces the magnitude of the vector s_k , making it nearly zero and consequently compromising the Hessian update. Other alternatives, such as the Goldstein conditions, in addition to the Armijo rule, prevent the acceptance of very small step lengths, but still do not guarantee that $s_k^\top y_k > 0$.

On the other hand, as seen in Algorithm 3, the Wolfe conditions include, in addition to the Armijo condition in line 3, that ensures sufficient decrease in the objective function, a curvature condition that controls the directional derivative (see line 5). This second condition prevents the step size from being excessively small and also guarantees that $s_k^\top y_k > 0$, thereby validating the quasi-Newton update. In general, Wolfe conditions represent an effective compromise between progress along the descent direction and stability of the Hessian approximation, and are therefore the standard choice in quasi-Newton algorithms.

4. Numerical Experiments

We present numerical experiments that involve images with various types of content, as shown in column (a) of Figure 1, specifically bike (first line), butterfly (second line), flower (third line), girl (fourth line), hat (fifth line), leaves (sixth line), parrot (seventh line), and plants (eighth line). These images are 256×256 pixels in size and are represented in the RGB color space [Dong et al., 2012b].

To test our approach, we introduce various types of noise to the images. More precisely, the images in column (b) of Figure 1 are corrupted as follows: different missing ratios of 20% (first line), 40% (second line), 60% (third line), and 80% (fourth line), representing the fraction of pixels that are erased and need to be restored; image blocks are erased, with 100 blocks of 5×5 pixels (fifth line) and 50 blocks of 10×10 pixels (sixth line); and, finally, to simulate more structured noises often found in real-world scenarios, we introduce text (seventh line) and scratch (eighth line) noise, which are common in digitized images.

The quality of the restored images is compared by applying Algorithm 1 to solve the optimization problem presented in (3), using all the functions described in this work as variation measures. Columns (c)–(f) of Figure 1 show the restored images generated with the four TV variants (LTVA $_l$, for $l = 1, 2, 3, 4$) based on the directions outlined in (5), while columns (g) and (h) display the results obtained with SLTVA (6) and ALTVA (7), respectively.

To assess the quality of the restored images, we used two well-known evaluation metrics: the global Peak Signal-to-Noise Ratio (PSNR) [Chanda and Majumder, 2011] and the Structural Similarity Index Metric (SSIM) [Wang et al., 2004]. Both metrics are monotonic, increasing as the restored image becomes more similar to the original. All algorithms were implemented in Matlab, and the experiments were conducted on a PC equipped with an Intel Core i7 CPU running at 2.79 GHz, 16 GB of RAM, and running the Windows 10 operating system. The stopping tolerance criterion for Algorithm 1 was set to $\epsilon = 10^{-3}$.

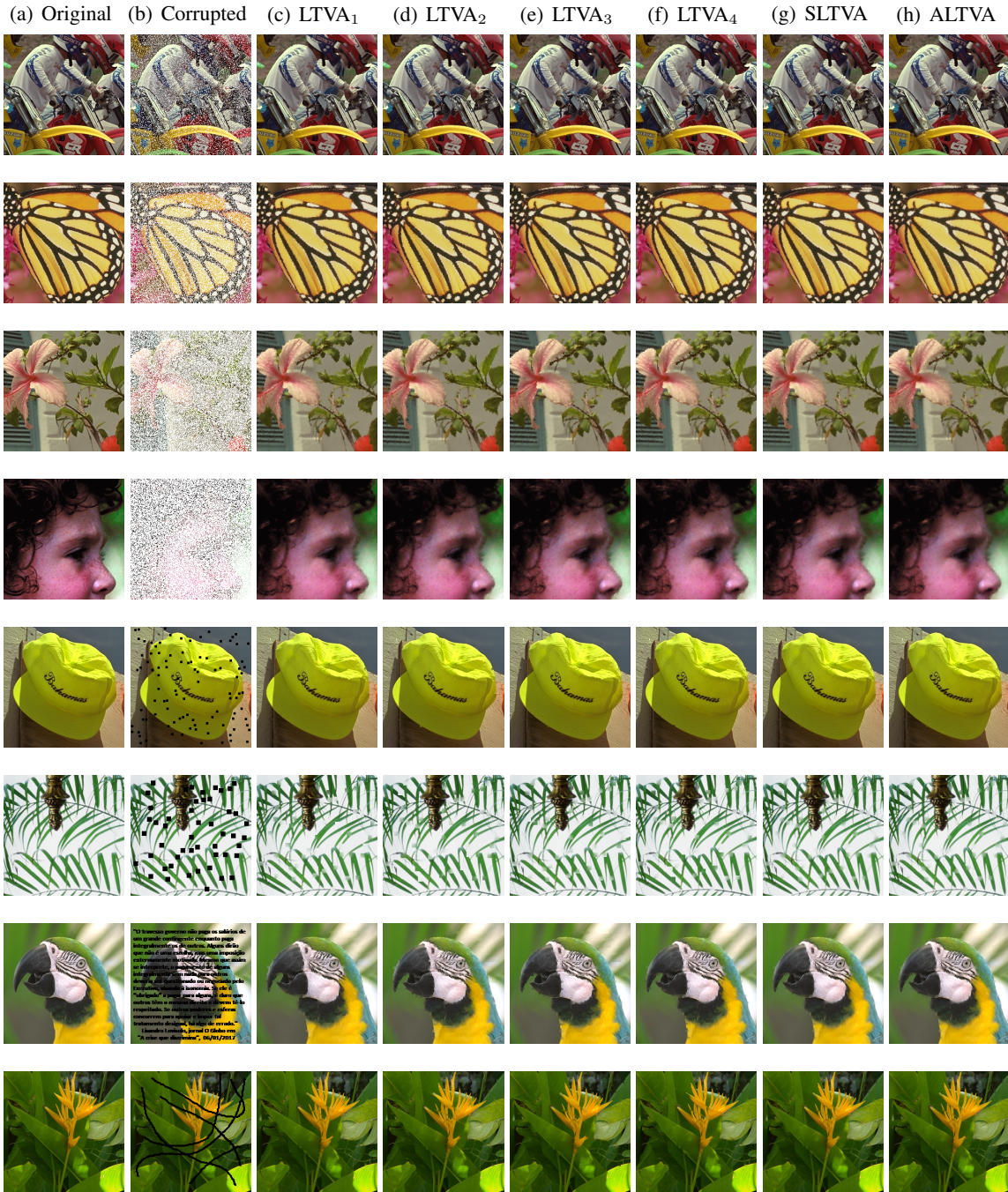


Figure 1: Performance of different TV-based image inpainting functions. Column (a) shows the original images; (b) presents the corrupted versions; (c)–(f) display the inpainted images obtained using the four TV variants ($LTVA_l$, for $l = 1, 2, 3, 4$); (g) presents the restored images produced by SLTVA, and (h) those from ALTVA. From top to bottom, the second column brings examples of the corruption obtained by the missing ratios of 20%, 40%, 60% and 80%, by erasing 100 blocks of 5×5 and 50 blocks of 10×10 pixels, as well as corruption by text and scratch

Table 1 presents the PSNR, SSIM, and execution time (these values are averages for the 8 test images) using the TV variants (LTVA₁, LTVA₂, LTVA₃ and LTVA₄), the SLTVA and ALTVA, for different types of pixel corruption. The first important observation is that the optimum solution obtained through SLTVA provides better-quality inpainted images, w.r.t. the PSNR and SSIM, than any TV direction variant alone and also the ALTVA, irrespective of the corruption type.

In terms of execution time, the superior image quality achieved by SLTVA does not come at the cost of increased computational effort. Although higher reconstruction quality is expected to come at the cost of increased computational complexity, SLTVA not only achieved the best visual outcomes among all evaluated functions but also converged significantly faster than the classical TV-based variants. This combination of efficiency and accuracy highlights the effectiveness of the proposed approach, especially in scenarios where both image fidelity and low runtime are crucial.

Given that the proposed SLTVA function, which considers four adjacent pixels, outperformed all the LTVA functions (which uses only two), the natural next step was to explore function ALTVA that extends the pixel neighborhood to incorporate all eight adjacent pixels. The idea was to examine whether further expanding the local context would enhance image restoration quality. As expected, ALTVA also outperformed all the LTVA functions, both in terms of PSNR and SSIM, although in some cases (20% missing and block5) it required longer convergence times. To our surprise, when comparing ALTVA with SLTVA, however, SLTVA always achieved better image quality, as measured by both PSNR and SSIM. This suggests that simply increasing the number of neighboring pixels does not necessarily result in better restoration — at least for the functions explored in this study. Regarding computational time, ALTVA was faster than SLTVA in the 80% missing, text, and scratch cases, but slower for all other noise types.

Table 1: Quality assessment and execution time for the four TV variants (LTVA_{*l*}, for $l = 1, 2, 3, 4$), the proposed SLTVA and also ALTVA.

Noise type		LTVA ₁	LTVA ₂	LTVA ₃	LTVA ₄	SLTVA	ALTVA
20% missing ratio	PSNR(dB)	32.14	31.73	32.08	31.72	34.66	33.53
	SSIM	0.9815	0.9803	0.9814	0.9803	0.9889	0.9860
	Time(s)	4.72	5.35	5.31	4.89	4.08	7.30
40% missing ratio	PSNR(dB)	28.41	28.08	28.37	28.14	30.42	29.56
	SSIM	0.9566	0.9544	0.9567	0.9546	0.9708	0.9645
	Time(s)	9.33	9.24	9.33	9.35	7.44	8.26
60% missing ratio	PSNR(dB)	25.90	25.70	25.91	25.72	27.36	26.66
	SSIM	0.9206	0.9191	0.9213	0.9184	0.9393	0.9288
	Time(s)	14.79	14.22	13.32	12.83	10.04	10.57
80% missing ratio	PSNR(dB)	23.33	23.26	23.31	23.26	24.08	23.59
	SSIM	0.8567	0.8564	0.8568	0.8551	0.8742	0.8597
	Time(s)	33.80	31.65	35.45	35.06	26.42	24.39
block 5	PSNR(dB)	34.91	34.68	35.08	34.81	37.12	37.02
	SSIM	0.9911	0.9909	0.9910	0.9907	0.9940	0.9939
	Time(s)	3.87	3.51	3.11	3.63	2.94	4.83
block 10	PSNR(dB)	29.91	29.45	29.65	29.57	31.90	31.88
	SSIM	0.9762	0.9754	0.9759	0.9755	0.9829	0.9827
	Time(s)	4.34	4.44	4.14	4.36	3.95	4.13
text	PSNR(dB)	27.25	27.02	27.22	27.28	28.71	28.49
	SSIM	0.9472	0.9462	0.9479	0.9478	0.9598	0.9577
	Time(s)	14.02	15.79	15.64	14.60	9.28	8.79
scratch	PSNR(dB)	30.69	30.18	30.68	30.17	32.02	31.77
	SSIM	0.9767	0.9753	0.9766	0.9747	0.9816	0.9805
	Time(s)	5.50	6.68	7.11	6.32	3.77	2.95

Employing the SLTVA function and solving problem (3) through the iterative procedure described in Algorithm 1 yields, at each iteration, a progressively updated version of the image being restored. In general, such iterative schemes are designed to gradually improve the visual quality of the image as the optimization advances. Ideally, one expects a monotonic improvement in the image quality, meaning that both metrics and perceptual quality should increase with each iteration.

Figure 2 illustrates this behavior by displaying the evolution of the PSNR (Figure 2(a)) and SSIM (Figure 2(b)), along with the functional value $f(x^{(k)})$ during the optimization process. These plots correspond to the “girl” image with 80% missing data – specifically, the fourth image in column (b) of Figure 1. It can be seen from the figure that as the value of the functional $f(x^{(k)})$ decreases, both PSNR and SSIM exhibit a corresponding increase, which indicates an improvement in image quality. This behavior confirms that the algorithm effectively directs the solution toward higher-quality image restorations. Finally, the best restored image is achieved when the algorithm converges to the minimum (the optimal value) of the functional, demonstrating the effectiveness of the proposed approach.

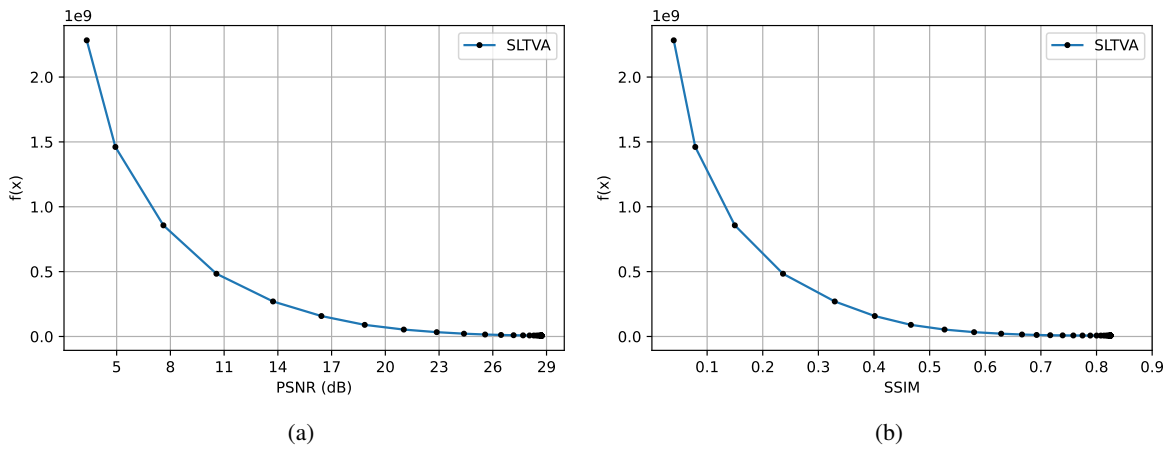


Figure 2: The functional value with respect to PSNR (a) and SSIM (b) obtained using SLTVA function along the iterations of the optimization process.

5. Conclusion

The image inpainting problem involves the process of reconstructing missing, corrupted, or degraded regions in digital images, aiming to generate visually consistent content that maintains structural coherence with the surrounding areas. In this work, we propose a new mathematical formulation that combines four total variation (TV) directions into a single objective function, enabling a more expressive representation of local structures to improve the quality of restored images. The resulting unconstrained differentiable nonlinear optimization problem is solved using the limited-memory quasi-Newton (L-BFGS) algorithm.

We compared our combined TV directions mathematical optimization function (SLTVA) against the four variants of the smoothed TV function (LTVA) and a function that extends the pixel neighborhood to include all eight adjacent pixels (ALTVA), using the well-known image quality metrics Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Metric (SSIM). Numerical experiments on diverse image sets and noise types show that our approach SLTVA outperforms not only the four variants of the classical TV functions LTVA but also the extended function ALTVA, both in terms of standard image quality metrics PSNR and SSIM, without increasing computational cost. Future research may explore incorporating our proposed function into existing image processing methods by replacing the classical TV term originally used in their formulations.

Acknowledgements

This work was partially supported by CNPq (308792/2022-0, 304680/2023-0) and CAPES (Finance Code 001). We also thank the referees for their valuable comments.

References

- Afonso, M. V., Bioucas-Dias, J. M., and Figueiredo, M. A. (2010). Fast image recovery using variable splitting and constrained optimization. *IEEE transactions on image processing*, 19(9):2345–2356.
- Armijo, L. (1966). Minimization of functions having lipschitz continuous first partial derivatives. *Pacific Journal of mathematics*, 16(1):1–3.
- Bertalmio, M., Sapiro, G., Caselles, V., and Ballester, C. (2000). Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, p. 417–424.
- Bonnans, J.-F., Gilbert, J. C., Lemaréchal, C., and Sagastizábal, C. A. (2006). *Numerical optimization: theoretical and practical aspects*. Springer Science & Business Media.
- BROYDEN, C. (1970). The convergence of a class of double-rank minimization algorithms. *IMA Journal of Applied Mathematics*, 6(3):222–231.
- Buades, A., Coll, B., and Morel, J.-M. (2005). A non-local algorithm for image denoising. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 2, p. 60–65. Ieee.
- Chambolle, A., Caselles, V., Cremers, D., Novaga, M., Pock, T., et al. (2010). An introduction to total variation for image analysis. *Theoretical foundations and numerical methods for sparse recovery*, 9(263–340):227.
- Chan, T. F. and Shen, J. (2001). Nontexture inpainting by curvature-driven diffusions. *Journal of visual communication and image representation*, 12(4):436–449.
- Chan, T. F. and Shen, J. (2005). *Image processing and analysis: variational, PDE, wavelet, and stochastic methods*. SIAM.
- Chanda, B. and Majumder, D. D. (2011). *Digital image processing and analysis*. PHI Learning Pvt. Ltd.
- Chudasama, D., Patel, T., Joshi, S., and Prajapati, G. (2014). Survey on various edge detection techniques on noisy images. *IJERT International Journal of Engineering Research & Technology ISSN*, p. 2278–0181.
- Dahl, J., Hansen, P. C., Jensen, S. H., and Jensen, T. L. (2010). Algorithms and software for total variation image reconstruction via first-order methods. *Numerical Algorithms*, 53(1):67–92.
- Davidon, W. (1959). Variable metric method for minimization. Technical report, Argonne National Lab., Lemont, Ill.
- Dong, W., Shi, G., and Li, X. (2012a). Nonlocal image restoration with bilateral variance estimation: a low-rank approach. *IEEE transactions on image processing*, 22(2):700–711.
- Dong, W., Zhang, L., Shi, G., and Li, X. (2012b). Nonlocally centralized sparse representation for image restoration. *IEEE transactions on Image Processing*, 22(4):1620–1630.
- Elharrouss, O., Almaadeed, N., Al-Maadeed, S., and Akbari, Y. (2020). Image inpainting: A review. *Neural Processing Letters*, 51:2007–2028.
- Fletcher, R. (1970). A new approach to variable metric algorithms. *The computer journal*, 13(3):317–322.
- Fletcher, R. and Powell, M. J. (1963). A rapidly convergent descent method for minimization. *The computer journal*, 6(2):163–168.

- Goldfarb, D. (1970). A family of variable-metric methods derived by variational means. *Mathematics of computation*, 24(109):23–26.
- Goldstein, A. A. (1965). On steepest descent. *Journal of the Society for Industrial and Applied Mathematics, Series A: Control*, 3(1):147–151.
- Gonzalez, G. (1984). *Microwave Transistor Amplifiers*. Prentice-Hall, Englewood Cliffs, NJ.
- Guillemot, C. and Le Meur, O. (2013). Image inpainting: Overview and recent advances. *IEEE signal processing magazine*, 31(1):127–144.
- Han, X., Wu, J., Wang, L., Chen, Y., Senhadji, L., and Shu, H. (2014). Linear total variation approximate regularized nuclear norm optimization for matrix completion. In *Abstract and Applied Analysis*, volume 2014, p. 765782. Wiley Online Library.
- Jähne, B. (2005). *Digital image processing*. Springer Science & Business Media.
- Liu, D. C. and Nocedal, J. (1989). On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528.
- Liu, X., Pedersen, M., and Wang, R. (2022). Survey of natural image enhancement techniques: Classification, evaluation, challenges, and perspectives. *Digital Signal Processing*, 127:103547.
- Minaee, S., Boykov, Y., Porikli, F., Plaza, A., Kehtarnavaz, N., and Terzopoulos, D. (2021). Image segmentation using deep learning: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3523–3542.
- Polyak, B. T. (2007). Newton’s method and its use in optimization. *European Journal of Operational Research*, 181(3):1086–1096.
- Rodríguez, P. (2013). Total variation regularization algorithms for images corrupted with different noise models: a review. *Journal of Electrical and Computer Engineering*, 2013(1):217021.
- Rudin, L. I., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268.
- Shanno, D. F. (1970). Conditioning of quasi-newton methods for function minimization. *Mathematics of computation*, 24(111):647–656.
- Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612.
- Wolfe, P. (1969). Convergence conditions for ascent methods. *SIAM review*, 11(2):226–235.
- Wolfe, P. (1971). Convergence conditions for ascent methods. ii: Some corrections. *SIAM review*, 13(2):185–188.