Efficient Minimization Method for a Generalized Total Variation Functional

Paul Rodríguez and Brendt Wohlberg

***Abstract*—Replacing the ℓ^2 data fidelity term of the standard Total Variation (TV) functional with an ℓ^1 data fidelity term has been found to offer a number of theoretical and practical benefits. Efficient algorithms for minimizing this ℓ^1-TV functional have only recently begun to be developed, the fastest of which exploit graph representations, and are restricted to the denoising problem. We describe an alternative approach that minimizes a generalized TV functional, including both ℓ^2-TV and ℓ^1-TV as special cases, and is capable of solving more general inverse problems than denoising (e.g., deconvolution). This algorithm is competitive with the graph-based methods in the denoising case, and is the fastest algorithm of which we are aware for general inverse problems involving a nontrivial forward linear operator.**

***Index Terms*—Image restoration, inverse problem, regularization, total variation.**

I. Introduction

Total Variation (TV) regularization has evolved from an image denoising method [1] into a more general technique for inverse problems [2], including deblurring [3], [4], blind deconvolution [5], and inpainting [6]. The standard ℓ^2-TV regularized solution of the inverse problem involving data $\mathbf{b}$ and forward linear operator A (the identity in the case of denoising) is the minimum of the functional

$$T(\mathbf{u}) = \frac{1}{p}\|A\mathbf{u}-\mathbf{b}\|_p^p + \frac{\lambda}{q}\left\|\sqrt{(D_x\mathbf{u})^2+(D_y\mathbf{u})^2}\right\|_q^q \qquad (1)$$

for $p=2$, $q=1$, and notation as follows:

- $(1/p)\|A\mathbf{u}-\mathbf{b}\|_p^p$ is the data fidelity term;
- $(1/q)\|\sqrt{(D_x\mathbf{u})^2+(D_y\mathbf{u})^2}\|_q^q$ is the regularization term;
- the p-norm of vector $\mathbf{u}$ is denoted by $\|\mathbf{u}\|_p$;
- scalar operations applied to a vector are considered to be applied element-wise, so that, for example, $\mathbf{u}=\mathbf{v}^2 \Rightarrow u_k = v_k^2$ and $\mathbf{u}=\sqrt{\mathbf{v}} \Rightarrow u_k=\sqrt{v_k}$;
- $\sqrt{(D_x\mathbf{u})^2+(D_y\mathbf{u})^2}$ is the discretization of $|\nabla\mathbf{u}|$;
- horizontal and vertical discrete derivative operators are denoted by D_x and D_y, respectively.

A significant recent development has been to use the ℓ^1 norm for the fidelity term, corresponding to choosing $p=1$, $q=1$ in (1). This ℓ^1-TV functional [7]–[10] has attracted attention due to a number of advantages, including superior performance with impulse noise [11].

The Iteratively Reweighted Norm (IRN) algorithm presented here minimizes the generalized TV functional (1), which includes the ℓ^2-TV and ℓ^1-TV as special cases, by representing the ℓ^p and ℓ^q norms by the equivalent weighted ℓ^2 norms. This algorithm is computationally efficient as well as simple to understand and implement. In this paper we expand on our previous results [12], [13] and provide a detailed analysis of the IRN algorithm, including proof of convergence, the development of an Inexact Newton (IN) method [14]–[16], and strategies for setting algorithm-specific parameters.

II. Algorithms for ℓ^2-TV and ℓ^1-TV

In this section, we provide a summary of the most important algorithms for minimizing the ℓ^2-TV and ℓ^1-TV functionals. In general, there are several types of numerical algorithms, based on (i) Euler-Lagrange equations (smooth approximation to the Euclidean norm of the gradient), (ii) primal-dual optimization methods (also called projection methods, first introduced in [17]), and (iii) binary Markov Random Field (MRF) optimization. Most recent algorithms based on a dual formulation [18], [19], or on a binary MRF optimization [20]–[23] do not need to solve a linear system of equations, but lack the ability to handle a nontrivial forward operator A in (1), and are, therefore, restricted to the denoising problem. We use *smooth approximation of the ℓ^1 norm* to describe the approximation of $|\nabla\mathbf{u}|$ by $\sqrt{|\nabla\mathbf{u}|^2+\epsilon^2}$, or by the Huber function [24], and *anisotropic separable approximation* [25] refers to the approximation $|\nabla\mathbf{u}| = |D_x\mathbf{u}|+|D_y\mathbf{u}|$.

A. ℓ^2-TV Numerical Algorithms

- *Artificial time marching/steepest descent* [1], ([26], Algorithm 8.2.1): Uses a smooth approximation of the ℓ^1 norm. Needs a linear solver. A in (1) may be nontrivial. May use a line-search to resolve the time-step. Slow compared to all other algorithms.
- *Primal-dual method* [17]: Uses a smooth approximation of the ℓ^1 norm. Needs a linear solver. A in (1) may be nontrivial.
- *Lagged-diffusivity* [3], [4], [26]: Uses a smooth approximation of the ℓ^1 norm. Needs a linear solver. A in (1) may be nontrivial.
- *Newton's method* [26]: Uses a smooth approximation of the ℓ^1 norm. Needs a linear solver. A in (1) may be nontrivial. Needs a line-search to resolve the time-step.

Manuscript received September 06, 2007; revised September 29, 2008. First published December 22, 2008; current version published January 09, 2009. This work was supported by the auspices of the National Nuclear Security Administration of the U.S. Department of Energy at Los Alamos National Laboratory under Contract DE-AC52-06NA25396 and was supported in part by the NNSA's Laboratory Directed Research and Development Program. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Stanley J. Reeves.

P. Rodríguez is with the Digital Signal Processing Group, Pontificia Universidad Católica del Perú, Lima, Peru (e-mail: prodrig@pucp.edu.pe).

B. Wohlberg is with T-7 Mathematical Modeling and Analysis, Los Alamos National Laboratory, Los Alamos, NM 87545 USA (e-mail: brendt@tmail.lanl.gov).

Digital Object Identifier 10.1109/TIP.2008.2008420

- *Chambolle* [18]: Does not need a linear solver. Formulation is in dual space (different approach than *primal-dual method* [17]). A in (1) is the identity.
- *Aujol's ℓ^2-TV approximation or A^2BC model* [19]: Does not need a linear solver. Uses Chambolle's projection method. A in (1) is the identity.
- *Majorization-Minimization (MM) method* [27], [28]: A in (1) may be nontrivial. Needs a linear solver. (The MM algorithm [29] can be considered a generalization of the Expectation-Maximization (EM) algorithm [30]).
- *TwIST method* [31]: Similar to MM method described above. Uses a predetermined Preconditioned CG (PCG) splitting method.
- *FTVd method* [32]: FFT based algorithm to solved the deconvolution TV problem for nontrivial A in (1).
- *Linear programming via interior point method* [33]: Needs a linear solver. A in (1) may be nontrivial. Due to nonnegativity constraints, the linear system to be solved has twice as many variables as the original problem.

B. *ℓ^1-TV Numerical Algorithms*

- *Standard approximation for ℓ^1-TV* [10], [11]: Uses a smooth approximation of the ℓ^1 norm. Similar to the artificial time marching algorithm for ℓ^2-TV. A in (1) may be nontrivial. May use a line-search to resolve the time-step. Slow compared to all other algorithms.
- *Chambolle's MRF (Markov Random Field) model* [20]: Does not need a linear solver. Operates over integer-valued images. A in (1) is the identity. Uses the anisotropic separable approximation. Similar ideas in [21]–[23], related to earlier algorithm in [34].
- *Second-Order Cone Programming* [35]: Needs a linear solver. A in (1) may be nontrivial. Memory requirements are high.
- *Darbon-Sigelle's graph-cut method* [21], [22]: Does not need a linear solver. Operates over integer-valued images. A in (1) is the identity. Uses the anisotropic separable approximation. Similar ideas in [20], [23].
- *Aujol's ℓ^1-TV approximation* [19]: Does not need a linear solver. Uses the anisotropic separable approximation. A in (1) is the identity.
- *Linear programming via interior point method* [33]: Needs a linear solver. A in (1) may be nontrivial. Due to nonnegativity constraints, the linear system to be solved has four times as many variables as the original problem.
- *Goldfarb–Yin parametric maximum flow* [23], [36]: Does not need a linear solver. Operates over integer-valued images. Uses the anisotropic separable approximation. A in (1) is the identity. Similar ideas in [20] and [21].

III. Technicalities

We represent 2-D images by 1-D vectors, obtained via any ordering (although the most reasonable choices are row-major or column-major) of the image pixels. The choice of ordering has no impact on the linear algebra, but the corresponding linear operators must be structured accordingly, which can affect the computational efficiency of a specific implementation of the derived algorithms.

The difference operators D_x and D_y are defined by applying the same 1-D discrete derivative along image rows and columns respectively. There are several possible choices for this operator (e.g., forward/backward difference, central difference, [37, Ch. 3] and several choices of how to handle the boundary conditions (zero boundary conditions, periodic boundary conditions, reflective/anti-reflective boundary conditions [38]). Throughout this paper, we use one of the two following definitions for the discrete derivative of $\mathbf{u} \in \mathbb{R}^N$:

- backward difference operator and fourth order difference operator with zero boundary conditions for endpoints

$$D\mathbf{u} = \begin{cases} u_{k+1} - u_k, & k \in \{0, 1, \ldots, N-2\} \\ (-u_{N-3} + 8u_{N-2})/12, & k = N-1; \end{cases}$$

- backward difference operator and half-sample symmetric boundary conditions for endpoints

$$D\mathbf{u} = \begin{cases} u_{k+1} - u_k, & k \in \{0, 1, \ldots, N-2\} \\ 0, & k = N-1. \end{cases}$$

The choice of the boundary conditions may have a dramatic impact in the quality of the reconstruction, particularly close to the boundaries [38]. Empirically, we have found that for some specific images the second definition gives a superior performance in terms of reconstruction SNR. Nevertheless, if large values are needed for the regularization parameter λ, the first definition is preferred due to invertibility of the diffusion operator $D_x^T D_x + D_y^T D_y$. In general both definitions give similar SNR results and time-performance, and unless specifically mentioned, we make no distinction between them.

Except where specified otherwise, test program run times were obtained on a 3 GHz Intel Pentium 4 processor with 1024 K if L2 cache and 2 G of RAM. All of the results presented here may be reproduced using the scripts in the `interfaces/matlab/tip` subdirectory of the NUMIPAD distribution [39], an open-source implementation of IRN and related algorithms. The original 512×512 pixel Peppers and Goldhill test images and the denoised/deconvolved ones from which these results were generated are also available [39] in file `nmp_tip2008_images.tgz`.

IV. Iteratively Reweighted Norm Approach

A. *Previous Related Work*

The IRN approach is closely related to the Iteratively Reweighted Least Squares (IRLS) method [40]–[44] (similar ideas have also been applied [45], [46] to solving the Basis Pursuit and Basis Pursuit denoising problems [47] for sparse representations). IRLS minimizes the ℓ^p norm

$$F(\mathbf{u}) = \frac{1}{p} \|A\mathbf{u} - \mathbf{b}\|_p^p \tag{2}$$

for $p \leq 2$ by approximating it, within an iterative scheme, by a weighted ℓ^2 norm. At iteration k the solution $\mathbf{u}^{(k)}$ is the minimizer of $(1/2)\|W^{(k)^{1/2}}(A\mathbf{u} - \mathbf{b})\|_2^2$, with weighting matrix $W^{(k)} = \mathrm{diag}\left(|A\mathbf{u}^{(k)} - \mathbf{b}|^{p-2}\right)$, and the iteration

$$\mathbf{u}^{(k+1)} = \left(A^T W^{(k)} A\right)^{-1} A^T W^{(k)} \mathbf{b}$$

which minimizes the weighted version of (2) using the weights derived from the previous iteration, converges to the minimizer of $F(\mathbf{u})$ [44].

When $p < 2$, the definition of the weighting matrix $W^{(k)}$ must be modified to avoid the possibility of division by zero. For $p = 1$, it may be shown [43] that the choice

$$W_{n,n}^{(k)} = \begin{cases} \left|r_n^{(k)}\right|^{-1} & \text{if } \left|r_n^{(k)}\right| \geq \epsilon \\ \epsilon^{-1} & \text{if } \left|r_n^{(k)}\right| < \epsilon \end{cases}$$

where $\mathbf{r}^{(k)} = A\mathbf{u}^{(k)} - \mathbf{b}$, and ϵ is a small positive number, guarantees global convergence to the minimizer of $\sum_n \rho_\epsilon(r_n)$, where

$$\rho_\epsilon(r_n) = \begin{cases} \epsilon^{-1} r_n^2 & \text{if } |r_n| \leq \epsilon^2 \\ 2|r_n| - \epsilon & \text{if } |r_n| > \epsilon^2 \end{cases}$$

is the Huber function [24].

B. Data Fidelity Term

The data fidelity term of the generalized TV functional (1) is the same as the term that the IRLS functional (2) seeks to minimize. In order to replace the ℓ^p norm by a ℓ^2 norm, we define the quadratic functional

$$Q_F^{(k)}(\mathbf{u}) = \frac{1}{2}\left\|W_F^{(k)^{1/2}}(A\mathbf{u} - \mathbf{b})\right\|_2^2 + \left(1 - \frac{p}{2}\right) F\left(\mathbf{u}^{(k)}\right) \quad (3)$$

where $\mathbf{u}^{(k)}$ is a constant representing the solution of the previous iteration, $F(\cdot)$ is defined in (2), and

$$W_F^{(k)} = \text{diag}\left(\tau_{F,\epsilon_F}\left(A\mathbf{u}^{(k)} - \mathbf{b}\right)\right). \quad (4)$$

Following a common strategy in IRLS type algorithms [41], the function

$$\tau_{F,\epsilon_F}(x) = \begin{cases} |x|^{p-2} & \text{if } |x| > \epsilon_F \\ \epsilon_F^{p-2} & \text{if } |x| \leq \epsilon_F \end{cases} \quad (5)$$

is defined to avoid numerical problems when $p < 2$ and $A\mathbf{u}^{(k)} - \mathbf{b}$ has zero-valued components. In Section IV-G, we propose an automated method for choosing the threshold ϵ_F.

The constant (with respect to $\mathbf{u}$) term $(1 - (p/2))F(\mathbf{u}^{(k)})$ is added in (3) so that, neglecting numerical precision issues in (3) and (4)

$$F\left(\mathbf{u}^{(k)}\right) = Q_F^{(k)}\left(\mathbf{u}^{(k)}\right) \quad (6)$$

as $\epsilon_F \to 0$. In other words, the weighted ℓ^2 norm tends to the original ℓ^p norm fidelity term at $\mathbf{u} = \mathbf{u}^{(k)}$. The bound (see the Appendix of [44])

$$F(\mathbf{u}) < Q_F^{(k)}(\mathbf{u}) \quad \forall \mathbf{u} \neq \mathbf{u}^{(k)}\ p \leq 2 \quad (7)$$

and the Fréchet derivatives of $F(\mathbf{u})$ and $Q_F^{(k)}(\mathbf{u})$

$$\nabla_{\mathbf{u}} F(\mathbf{u}) = A^T(A\mathbf{u} - \mathbf{b})^{p-1}$$
$$\nabla_{\mathbf{u}} Q_F^{(k)}(\mathbf{u}) = A^T W_F^{(k)}(A\mathbf{u} - \mathbf{b}).$$

play an important role in the convergence proof in Section IV-F. Observe also that

$$\nabla_{\mathbf{u}} F(\mathbf{u})\big|_{\mathbf{u}=\mathbf{u}^{(k)}} = \nabla_{\mathbf{u}} Q_F^{(k)}(\mathbf{u})\Big|_{\mathbf{u}=\mathbf{u}^{(k)}} \quad (8)$$

when $\epsilon_F \to 0$, and note that the original fidelity term in (2) and its quadratic version in (3) have the same value and tangent direction at $\mathbf{u} = \mathbf{u}^{(k)}$.

C. Regularization Term

It is not quite as obvious how to express the TV regularization term from (1)

$$R(\mathbf{u}) = \frac{1}{q}\left\|\sqrt{(D_x\mathbf{u})^2 + (D_y\mathbf{u})^2}\right\|_q^q \quad (9)$$

as a weighted ℓ^2 norm. Given vectors $\boldsymbol{\xi}$ and $\boldsymbol{\chi}$, and diagonal matrix Ω_R with entries ω_k, we have

$$\left\|\begin{pmatrix} \Omega_R^{1/2} & 0 \\ 0 & \Omega_R^{1/2} \end{pmatrix}\begin{pmatrix} \boldsymbol{\xi} \\ \boldsymbol{\chi} \end{pmatrix}\right\|_2^2 = \sum_k \omega_k \xi_k^2 + \omega_k \chi_k^2$$

so that when $\Omega_R = \text{diag}\left((\boldsymbol{\xi}^2 + \boldsymbol{\chi}^2)^{(q-2)/2}\right)$, we have

$$\left\|\begin{pmatrix} \Omega_R^{1/2} & 0 \\ 0 & \Omega_R^{1/2} \end{pmatrix}\begin{pmatrix} \boldsymbol{\xi} \\ \boldsymbol{\chi} \end{pmatrix}\right\|_2^2 = \|\sqrt{\boldsymbol{\xi}^2 + \boldsymbol{\chi}^2}\|_q^q.$$

We, therefore, set $\boldsymbol{\xi} = D_x\mathbf{u}$, $\boldsymbol{\chi} = D_y\mathbf{u}$, and define the matrices

$$D = \begin{pmatrix} D_x \\ D_y \end{pmatrix} W_R^{(k)} = \begin{pmatrix} \Omega_R^{(k)} & 0 \\ 0 & \Omega_R^{(k)} \end{pmatrix} \quad (10)$$
$$\Omega_R^{(k)} = \text{diag}\left(\left(\left(D_x\mathbf{u}^{(k)}\right)^2 + \left(D_y\mathbf{u}^{(k)}\right)^2\right)^{(q-2)/2}\right)$$

which gives the desired term

$$\left\|W_R^{(k)^{1/2}} D\mathbf{u}\right\|_2^2 = \left\|\begin{pmatrix} \Omega_R^{(k)^{1/2}} & 0 \\ 0 & \Omega_R^{(k)^{1/2}} \end{pmatrix}\begin{pmatrix} D_x \\ D_y \end{pmatrix}\mathbf{u}\right\|_2^2.$$

It is important emphasize that is *not* the anisotropic separable approximation [25].

Now, define the quadratic functional

$$Q_R^{(k)}(\mathbf{u}) = \frac{1}{2}\left\|W_R^{(k)^{1/2}} D\mathbf{u}\right\|_2^2 + \left(1 - \frac{q}{2}\right) R(\mathbf{u}^{(k)}) \quad (11)$$

where $\mathbf{u}^{(k)}$ is a constant representing the solution of the previous iteration. As in the case of the data fidelity term, care needs to be taken when $q < 2$ and $(D_x\mathbf{u})^2 + (D_y\mathbf{u})^2$ has zero-valued components. We, therefore, define

$$\tau_{R,\epsilon_R}(x) = \begin{cases} |x|^{(q-2)/2} & \text{if } |x| > \epsilon_R \\ 0 & \text{if } |x| \leq \epsilon_R \end{cases} \quad (12)$$

and set

$$\Omega_R^{(k)} = \text{diag}\left(\tau_{R,\epsilon_R}\left(\left(D_x\mathbf{u}^{(k)}\right)^2 + \left(D_y\mathbf{u}^{(k)}\right)^2\right)\right) \quad (13)$$

so that

$$R\left(\mathbf{u}^{(k)}\right) = Q_R^{(k)}\left(\mathbf{u}^{(k)}\right) \quad (14)$$

when $\epsilon_R \to 0$. Note that τ_{R,ϵ_R} sets values smaller than the threshold, ϵ_R, to zero, as opposed to τ_{F,ϵ_F}, which sets values smaller than the threshold, ϵ_F, to ϵ_F^{p-2}. The motivation for this choice is that a region with very small or zero gradient should be allowed to have zero contribution to the regularization term, rather than be clamped to some minimum value. In practice, however, this choice does not give significantly different results than the standard IRLS approach represented by τ_{F,ϵ_F}.

We now consider some results required for the convergence proof in Section IV-F, starting with a proof that

$$R(\mathbf{u}) < Q_R^{(k)}(\mathbf{u}) \quad \forall \mathbf{u} \neq \mathbf{u}^{(k)} \quad q \leq 2. \tag{15}$$

This is easily shown by defining the scalar functions

$$f(v_n) = \frac{1}{q}(v_n)^{\frac{q}{2}}$$
$$g^{(k)}(v_n) = \frac{1}{2}\left(v_n^{(k)}\right)^{\frac{q}{2}-1} v_n + (1-0.5q) f\left(v_n^{(k)}\right).$$

Function $g(v_n)$ defines a line tangent to $f(v_n)$ at $v_n = v_n^{(k)}$, and since $q \leq 2$, function $f(v_n)$ is concave, so that $f(v_n) \leq g^{(k)}(v_n)\ \forall v_n$, with equality only for $v_n = v_n^{(k)}$. Let $\mathbf{v} = (D_x\mathbf{u})^2 + (D_y\mathbf{u})^2$, and define $R(\mathbf{u})$ and $Q_R^{(k)}(\mathbf{u})$ as

$$R(\mathbf{u}) = \sum_n f(v_n) \quad Q_R^{(k)}(\mathbf{u}) = \sum_n g(v_n)$$

which are equivalent to definitions in (9) and (11), and imply (15). Some algebraic manipulation is needed to compute the Fréchet derivatives of $R(\mathbf{u})$ and $Q_R^{(k)}(\mathbf{u})$

$$\nabla_{\mathbf{u}} R(\mathbf{u}) = \left(D_x^T \Omega D_x + D_y^T \Omega D_y\right)\mathbf{u}$$
$$\nabla_{\mathbf{u}} Q_R^{(k)}(\mathbf{u}) = D^T W_R^{(k)} D\mathbf{u},$$

where $\Omega = \operatorname{diag}\left(((D_x\mathbf{u})^2 + (D_y\mathbf{u})^2)^{(q-2)/2}\right)$. Note that the *weighting* matrix Ω for $\nabla R(\mathbf{u})$ does not have the superscript (k) and is the result of re-arranging the terms when computing the gradient. It is straightforward to check that

$$\nabla_{\mathbf{u}} R(\mathbf{u})\big|_{\mathbf{u}=\mathbf{u}^{(k)}} = \nabla_{\mathbf{u}} Q_R^{(k)}(\mathbf{u})\Big|_{\mathbf{u}=\mathbf{u}^{(k)}} \tag{16}$$

when $\epsilon_R \to 0$. As for the fidelity term, note that the original regularization term (9) and its quadratic version (11) have the same value and tangent direction at $\mathbf{u} = \mathbf{u}^{(k)}$.

D. General Algorithm

Combining the terms described in Sections IV-B and IV-C, gives the functional

$$T^{(k)}(\mathbf{u}) = \frac{1}{2}\left\| W_F^{(k)^{1/2}}(A\mathbf{u} - \mathbf{b})\right\|_2^2 + \frac{\lambda}{2}\left\| W_R^{(k)^{1/2}} D\mathbf{u}\right\|_2^2 + C\left(\mathbf{u}^{(k)}\right) \tag{17}$$

where $C(\mathbf{u}^{(k)})$ combines the constants, with respect to $\mathbf{u}$, in (3) and (11). This functional may be expressed as

$$T^{(k)}(\mathbf{u}) = \frac{1}{2}\left\| W^{(k)^{1/2}}\left(\tilde{A}\mathbf{u} - \tilde{\mathbf{b}}\right)\right\|_2^2 + C\left(\mathbf{u}^{(k)}\right) \tag{18}$$

where

$$W^{(k)} = \begin{pmatrix} W_F^{(k)} & 0 \\ 0 & W_R^{(k)} \end{pmatrix}, \tilde{A} = \begin{pmatrix} A \\ \sqrt{\lambda} D \end{pmatrix}, \text{and}$$
$$\tilde{\mathbf{b}} = \begin{pmatrix} \mathbf{b} \\ 0 \end{pmatrix}$$

which has the same form as a standard IRLS problem, but differs in the computation of the weighting matrix. Since (17) represents a quadratic functional, the gradient and Hessian

$$\nabla_{\mathbf{u}} T^{(k)}(\mathbf{u}) = \left(A^T W_F^{(k)} A + \lambda D^T W_R^{(k)} D\right)\mathbf{u} + A^T W_F^{(k)}\mathbf{b} \tag{19}$$
$$\nabla_{\mathbf{u}}^2 T^{(k)}(\mathbf{u}) = A^T W_F^{(k)} A + \lambda D^T W_R^{(k)} D \tag{20}$$

are easily derived.

The resulting procedure to iteratively minimize (17), derived by setting (19) equal to zero, is summarized in Algorithm 1. It should also be noted that, after development of this algorithm, first published in [12], we became aware of an independent approach restricted to the ℓ^2-TV problem [27], developed within the Majorization-Minimization framework [29], which results in a similar algorithm to IRN for the $p = 2$, $q = 1$ case, but was not extended to the general problem.

This algorithm does not include an explicit termination condition since a number of possibilities exist, depending on the specific problem context. The obvious choice is based on the fractional change of the functional value at each iteration, but we note that a smaller functional cost does not always imply a better reconstruction quality. For the experiments reported here, we have utilized a fixed number of iterations, which simplifies comparisons across variations in other parameters.

While functional (1) does not necessarily have a unique minimizer when $p, q \leq 1$, functional (17), being quadratic, has a unique minimizer for all $p, q \leq 2$ *at each iteration* of the IRN algorithm. When solving a problem without a unique minimizer, the solution found by the IRN algorithm will depend on the initial solution and the trajectory followed by the iterations, which depends on parameters ϵ_F and ϵ_R, and on the accuracy with which the linear system derived from the gradient of (17) is solved at each iteration.

Algorithm 1: General case of IRN algorithm. Matrix D is defined in (10).

Inputs Linear operator: A Noisy image: $\mathbf{b}$

Initialize

$$\mathbf{u}^{(0)} = (A^T A + \lambda D^T D)^{-1} A^T \mathbf{b}$$

for $k = 0, 1, \ldots$

$$W_F^{(k)} = \operatorname{diag}\left(\tau_{F,\epsilon_F}\left(A\mathbf{u}^{(k)} - \mathbf{b}\right)\right)$$

```
        $\Omega_R^{(k)} = \mathrm{diag}\left(\tau_{R,\epsilon_R}\left(\left(D_x\mathbf{u}^{(k)}\right)^2 + \left(D_y\mathbf{u}^{(k)}\right)^2\right)\right)$
        $W_R^{(k)} = \begin{pmatrix}\Omega_R^{(k)} & 0\\ 0 & \Omega_R^{(k)}\end{pmatrix}$
        $\mathbf{u}^{(k+1)} = \left(A^TW_F^{(k)}A + \lambda D^TW_R^{(k)}D\right)^{-1}A^TW_F^{(k)}\mathbf{b}$
    end
```

The matrix inversion in Algorithm 1 can be achieved using the Conjugate Gradient (CG) or a Preconditioned CG (PCG) method such as Jacobi line relaxation (JLR) or symmetric Gauss-Seidel line relaxation (SLGS) [48]. We have found that a significant speed improvement per iteration may be achieved by starting with a high CG (PCG) tolerance which is decreased with each main iteration until a preset value is reached. This behavior (i.e., speed improvement and better quality reconstruction for variable CG/PCG tolerance) is observed for other iterative methods, such as Newton's method [14].

Ideas described above led to the implementation of an Inexact Newton (IN) method [14]–[16] to solve the linear system described in the Algorithm 1, in which the tolerance is automatically adapted at each iteration [49]. First we notice that $\mathbf{u}^{(k+1)}$ may be found via Newton's method [15, Ch.5]. Setting (19) to zero and noting that $\nabla^2_{\mathbf{u}}T^{(k)}(\mathbf{u}) = A^TW_F^{(k)}A + \lambda D^TW_R^{(k)}D$, then

$$\left(\nabla^2_{\mathbf{u}}T^{(k)}(\mathbf{u})\right)\mathbf{u}^{(k+1)} = A^TW_F^{(k)}\mathbf{b}$$

and by adding and subtracting the Hessian times a current estimate for the solution we get $(\nabla^2_{\mathbf{u}}T^{(k)}(\mathbf{u}))\mathbf{u}^{(k+1,n+1)} = A^TW_F^{(k)}\mathbf{b} + (\nabla^2_{\mathbf{u}}T^{(k)}(\mathbf{u}))\mathbf{u}^{(k+1,n)} - (\nabla^2_{\mathbf{u}}T^{(k)}(\mathbf{u}))\mathbf{u}^{(k+1,n)}$. After some algebraic manipulation we obtain the Newton form

$$\mathbf{u}^{(k+1,n+1)} = \mathbf{u}^{(k+1,n)} - \left(\nabla^2_{\mathbf{u}}T^{(k)}(\mathbf{u})\right)^{-1}\nabla_{\mathbf{u}}T^{(k)}(\mathbf{u}). \tag{21}$$

While this equation resembles the core equation of the Lagged-Diffusivity (LD) algorithm for ℓ^2-TV (particularly the description found in [26]), we emphasize that IRN is not simply a generalization of the LD algorithm. The derivation of the LD method is motivated by finding the gradient of $T_{LD}(\mathbf{u}) = (1/2)\|A\mathbf{u}-\mathbf{b}\|_2^2 + \lambda\psi((D_x\mathbf{u})^2 + (D_y\mathbf{u})^2)$ where a choice such as $\psi(\mathbf{v}) = \sqrt{\mathbf{v}^2+\epsilon^2}$ gives a smooth approximation to the square root, whereas the IRN method is motivated by finding the gradient of (17), leading to a different choice of weighting matrix. In addition, the substitution described in Section IV-E (first described in [12] and [13]), which has a dramatic impact in the computational and quality performance of the IRN method for ℓ^1-TV denoising, has no obvious motivation within the LD framework.

Algorithm 2 summarizes how the IRN approach takes advantage of the IN method. Details of the full derivation are not provided here since they correspond to simple algebraic manipulation of (21), following the descriptions found in [14], [15], and [49]. Once again, we do not include any fixed condition to stop the main iteration and leave it open to accommodate the particular needs of a specific problem. However, we do include a simple condition to terminate the inner loop in Algorithm 2; this condition, proposed in [49], along with the constants defined in Algorithm 2, adapts the tolerance used to solve the linear system at every outer and inner loop.

To show the differences between Algorithm 1 and 2, and emphasize the effect of different CG tolerance policies, we ℓ^1-TV deconvolved a blurred and noisy version of the Peppers image using three different choices of regularization parameter λ (0.75, 1.0 and 1.25), employing a fixed CG tolerance (10^{-13}), a variable hand-tuned CG tolerance (exponentially decreasing between 10^{-8} and 10^{-13}) and variable auto-adapted CG tolerance (via the IN method, with $\alpha_G, \alpha_L = 1.3$, see Algorithm 2). The blurred and noisy Peppers image was constructed by convolving the noise-free Peppers image by a separable smoothing filter having 9 taps and approximating a Gaussian with standard deviation of 2 (i.e., $\exp(-k^2/2\cdot\sigma^2)/2\pi\sigma^2$ with $k \in [-4, 4]$ and $\sigma = 2$), and then corrupting the result with salt and pepper noise (10% of the pixels), giving an image with an SNR of 1.4 dB with respect to the original. Fig. 1 and Table I display the reconstruction qualities and run times respectively for $\lambda = 10^{-3}$ (similar results are obtained for 10^{-2} and 10^{-1}) and the three CG accuracy policies of the simulation previously described. Clearly, when a variable CG tolerance policy is used, the reconstruction quality is not only improved, but the elapsed time is reduced, independent of parameter λ. Similar results are obtained for other images.

Algorithm 2: IRN Algorithm via Inexact Newton. SOLVER is any procedure to solve a linear system, e.g., CG, PCG, IN, where $\eta^{(k,n)}$ sets its tolerance. $\nabla^2_{\mathbf{u}}T^{(k)}$ is defined in (20).

```
Inputs Linear operator: $A$ Noisy image: $\mathbf{b}$
Constants
    $\eta^{(k,0)} = 0.5$   (initial tolerance for SOLVER)
    $\alpha_G, \alpha_L = (1+\sqrt{5})/2$   (see [49])
    $\gamma_G, \gamma_L = 0.5$
Initialize
    $\mathbf{u}^{(0)} = (A^TA + \lambda D^TD)^{-1}A^T\mathbf{b}$
for $k = 0, 1, \ldots$
        $W_F^{(k)} = \mathrm{diag}\left(\tau_{F,\epsilon_F}\left(A\mathbf{u}^{(k)} - \mathbf{b}\right)\right)$
        $\mathbf{b}^{(k)} = W_F^{(k)}\mathbf{b}$
        $\mathbf{r}^{(k)} = \left(\nabla^2_{\mathbf{u}}T^{(k)}(\mathbf{u})\right)\mathbf{u}^{(k)} - \mathbf{b}^{(k)}$
        $\eta^{(k)} = \gamma_G\left(\frac{\|\mathbf{r}^{(k)}\|}{\|\mathbf{b}^{(k)}\|}\right)^{\alpha_G}$   (global tolerance)
    $\mathbf{u}^{(k+1,0)} = \mathbf{b}^{(k)}$
    $\mathbf{b}^{(k,0)} = \left(\nabla^2_{\mathbf{u}}T^{(k)}(\mathbf{u})\right)\mathbf{b}^{(k)} - \mathbf{b}^{(k)}$
    $\mathbf{x}^{(k,0)} = \mathrm{SOLVER}\left(\nabla^2_{\mathbf{u}}T^{(k)}(\mathbf{u}), \mathbf{b}^{(k,0)}, \eta^{(k,0)}\right)$
    $\mathbf{u}^{(k+1,1)} = \mathbf{u}^{(k,0)} - \mathbf{x}^{(k+1,0)}$
```

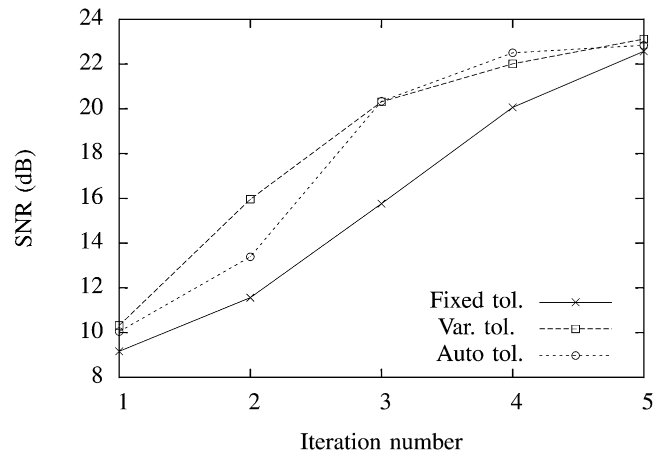

Fig. 1. ℓ^1-TV deconvolution SNR values against algorithm iteration number with $\lambda = 10^{-3}$ and for fixed (10^{-13}), variable (hand-tuned, exponentially decreasing between 10^{-8} and 10^{-13}) and auto (via the Inexact Newton method) CG tolerance. Input image was Peppers, convolved with a blurring Gaussian filter and then corrupted with salt and pepper noise (10% of pixels, SNR 1.2 dB).

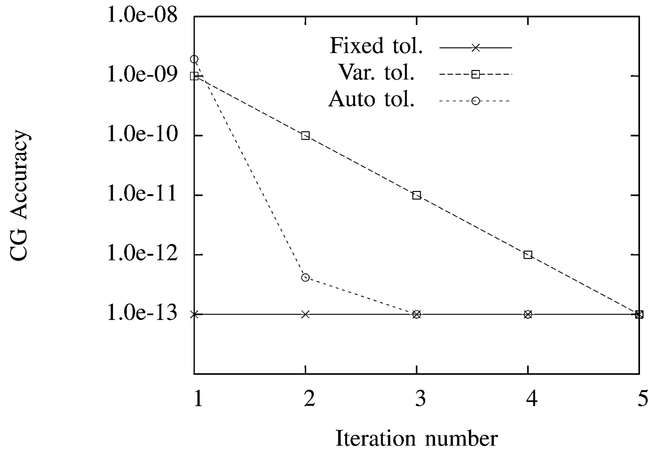

Fig. 2. Evolution of the three different the CG accuracy policies for simulation described in Fig. 1.

TABLE I
ℓ^1-TV DECONVOLUTION ELAPSED TIME (S) AGAINST ALGORITHM ITERATION NUMBER, CORRESPONDING TO THE SIMULATION DESCRIBED IN FIG. 1

Iter. / Method	1	2	3	4	5
Fixed tol.	66.3	173.7	283.7	362.0	418.6
Var tol.	18.8	41.2	66.8	100.4	150.1
Auto tol.	24.7	67.2	86.3	141.6	181.9

```
    for $n = 1, 2, \ldots$
        $\mathbf{b}^{(k,n)} = \mathbf{b}^{(k,n-1)} - A\mathbf{x}^{(k,n-1)}$
        if $\left(\frac{\|\mathbf{b}^{(k,n)}\|}{\|\mathbf{b}\|} < \eta^{(k)}\right) \left\{\mathbf{u}^{(k+1)} = \mathbf{u}^{(k+1,n)}\text{BREAK}\right\}$
        $\eta^{(k,n)} = \gamma_L \left(\frac{\|\mathbf{b}^{(k,n)}\|}{\|\mathbf{b}^{(k,n-1)}\|}\right)^{\alpha_L}$
        $\mathbf{x}^{(k,n)} = \text{SOLVER}\left(\nabla^2_{\mathbf{u}} T^{(k)}(\mathbf{u}), \mathbf{b}^{(k,n)}, \eta^{(k,n)}\right)$
        $\mathbf{u}^{(k+1,n+1)} = \mathbf{u}^{(k+1,n)} - \mathbf{x}^{(k,n)}$
    end
end
```

In Fig. 2, we show the evolution for the automatically adapted tolerance when solving the problem described for Fig. 1, with variable (exponentially decreasing between 10^{-8} and 10^{-13}) and fixed (10^{-13}) CG tolerances shown for comparison. Note that the automatically adapted tolerance rapidly decreases the tolerance to 10^{-13} within a few iterations. Although these different approaches do not have a significant impact on the reconstruction quality, there are noticeable differences in the elapsed time (see Table I), the hand-tuned, exponentially decreasing CG accuracy policy being about 15% faster than the IN method accuracy, both of which significantly outperform the fixed accuracy policy. While careful hand tuning of the accuracy evolution strategy can improve upon that of the IN method, it represents a very useful method when hand tuning is not practical, or to provide a good starting point for subsequent hand tuning.

TABLE II
ℓ^1-TV DENOISING ELAPSED TIME (S) AGAINST ALGORITHM ITERATION NUMBER, CORRESPONDING TO THE SIMULATION DESCRIBED IN FIG. 3

Iter. / Method	1	2	3	4	5
Fixed	0.60	1.01	1.59	2.67	4.09
Var	0.71	1.36	2.29	5.25	9.50
Auto tol.	1.50	2.40	3.36	6.49	10.53
General	26.04	52.14	72.73	97.14	119.11

E. Denoising Algorithm

In the case of the denoising problem, when $A = I$ and $p \neq 2$ (i.e., not applicable for ℓ^2-TV since matrix W_F will be equal to the identity), we may apply the substitution $\mathbf{v} = W_F^{1/2}\mathbf{u}$ in (17) giving

$$T(\mathbf{v}) = \frac{1}{2}\left\|\mathbf{v} - W_F^{1/2}\mathbf{b}\right\|_2^2 + \frac{\lambda}{2}\left\|W_R^{1/2} D W_F^{-1/2}\mathbf{v}\right\|_2^2$$

with solution

$$\mathbf{v} = \left(I + \lambda W_F^{-1/2} D^T W_R D W_F^{-1/2}\right)^{-1} W_F^{1/2}\mathbf{b}. \quad (22)$$

This substitution not only gives a better time performance than the general algorithm (see Table II) but the resulting linear system (22) to be solved in the ℓ^1-TV case is better conditioned than (19) or (21) which need to be solved for the general case, and, thus, its accuracy requirements can be looser.

In a similar fashion to Algorithm 1, the denoising IRN algorithm can be adapted to take advantage of an auto-adapted tolerance policy through the Inexact Newton method. Neither the derivation of this procedure (denoising IRN algorithm via Inexact Newton) nor the algorithm itself are listed here since it

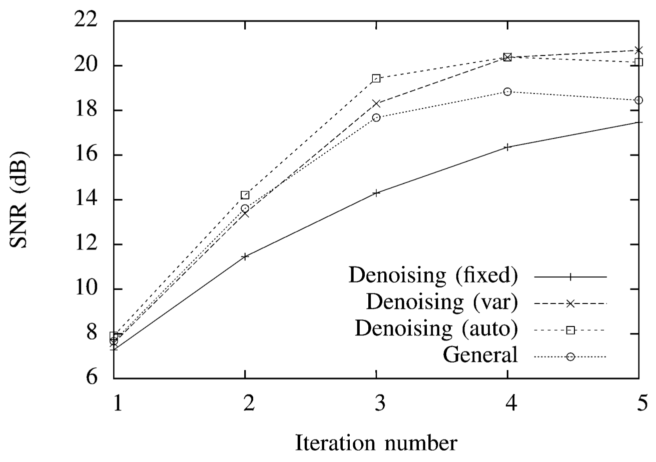

Fig. 3. ℓ^1-TV denoising SNR values against algorithm iteration number, for $\lambda = 1.25$. Three curves correspond to the denoising algorithm, with fixed CG accuracy (10^{-5}), variable (exponentially decreasing between 10^{-3} and 10^{-5}) and auto adapted (via the inexact Newton method) CG accuracy. The last curve corresponds to the general algorithm with fixed CG accuracy (10^{-6}). Input image was Goldhill corrupted with salt and pepper noise (10% of its pixels, SNR 1.2 dB).

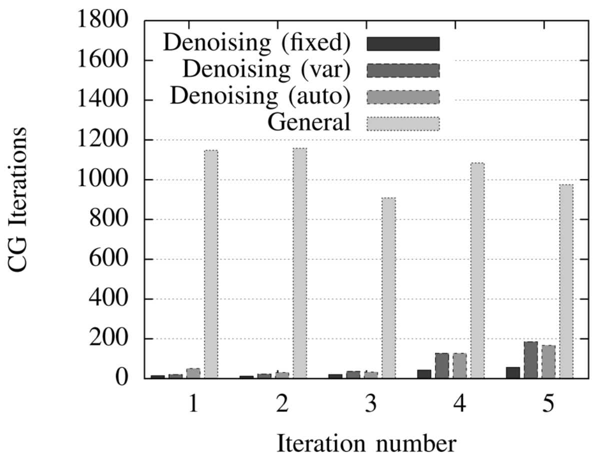

Fig. 4. Comparison between the number of CG iterations for the denoising (Section IV-E) and general (Section IV-D) algorithms, corresponding to the simulation in Fig. 3.

TABLE III
A Comparison Between PMF [23] (4 and 16 Neighbors) and IRN ℓ^1-TV Denoising Run Times for a 512×512 Goldhill Image Corrupted by 10% Salt and Pepper Noise

λ / Method	0.50	0.75	1.00	1.25
PMF-4	0.76 s 19.2 dB	0.9 s 17.5 dB	1.08 s 15.8 dB	1.33 s 14.7 dB
PMF-16	1.42 s 10.8 dB	1.75 s 18.6 dB	2.20 s 16.9 dB	2.45 s 15.5 dB
IRN (denoising)	2.41 s 13.3 dB	3.04 s 15.9 dB	2.06 s 16.6 dB	2.09 s 17.4 dB

is a simple exercise of setting $A = I$ in (20) and (21) and then applying the change of variable described in Section IV-E.

Applying this modification to the general algorithm of Section IV-D for ℓ^1-TV denoising was found to result in a very large reduction in the required number of CG iterations, and also in increased reconstruction quality. The advantages of the denoising-specific algorithm over the general algorithm (Section IV-D) are clearly observed in Figs. 3 and 4, computed using the Goldhill image corrupted with salt and pepper noise (10% of pixels). The denoising-specific algorithm not only outperforms the general algorithm in the number of CG iterations needed (and, thus, in time-performance—about 10 s compared with about 120 s) but in reconstruction quality as well. The general algorithm used a fixed CG accuracy policy (10^{-6}) whereas for the denoising-specific algorithm we used 3 different policies: fixed (10^{-5}), exponentially decreasing between 10^{-3} and 10^{-5}, and auto-accuracy via the Inexact Newton method. The high accuracy used by the general algorithm is needed in order to attain a comparable SNR reconstruction quality. These results (reduction in CG iterations and improved SNR) are observed independent of the λ parameter and CG accuracy policy.

To the best of our knowledge, the Goldfarb–Yin parametric maximum flow algorithm [23] for ℓ^1-TV denoising is the fastest implementation available, with better time performance than Chambolle's MRF model [20] and Darbon-Sigelle's graph-cut method [21], [22]. In Table III, we compare the time-performance of the IRN algorithm and the Goldfarb–Yin parametric maximum flow algorithm (using their implementation [36]) for ℓ^1-TV denoising for several values of the λ parameter. The IRN algorithm was set up to use PCG (via Jacobi line relaxation), and to auto-adapt the the thresholds for the weighting matrices (see Section IV-G) and to run for 5 iterations. Overall results are similar for time-performance and reconstruction quality, but we have to acknowledge that the balance favors the Goldfarb–Yin parametric maximum flow algorithm. Their method has the advantage of providing an exact solution to the TV optimization problem (although this is usually not an issue in practical denoising application), but is not extensible to more general image restoration problems.

F. Convergence Analysis

For convenience, we reproduce the generalized TV cost functional (1)

$$T(\mathbf{u}) = \frac{1}{p}\|A\mathbf{u} - \mathbf{b}\|_p^p + \frac{\lambda}{q}\left\|\sqrt{(D_x\mathbf{u})^2 + (D_y\mathbf{u})^2}\right\|_q^q$$

and its quadratic approximation (17)

$$T^{(k)}(\mathbf{u}) = \frac{1}{2}\left\|W_F^{(k)^{1/2}}(A\mathbf{u} - \mathbf{b})\right\|_2^2 + \frac{\lambda}{2}\left\|W_R^{(k)^{1/2}}D\mathbf{u}\right\|_2^2 + C\left(\mathbf{u}^{(k)}\right)$$

and note that from (6) and (14) it is easy to check that $T(\mathbf{u}^{(k)}) = T^{(k)}(\mathbf{u}^{(k)})$, where $\mathbf{u}^{(k)}$ is the vector used to compute the weights $W_F^{(k)}$ and $\Omega_R^{(k)}$. Moreover, from (7) and (15), we have that

$$T(\mathbf{u}) \leq T^{(k)}(\mathbf{u}) \quad \forall \mathbf{u}\, p, q \leq 2$$

with equality only for $\mathbf{u} = \mathbf{u}^{(k)}$. It is also easy to check [see (8) and (16)] that

$$\nabla_{\mathbf{u}} T(\mathbf{u})|_{\mathbf{u}=\mathbf{u}^{(k)}} = \nabla_{\mathbf{u}} T^{(k)}(\mathbf{u})\Big|_{\mathbf{u}=\mathbf{u}^{(k)}}. \tag{23}$$

In the general $k \geq 1$ case, (20) implies that $\nabla^2_{\mathbf{u}} T^{(k)}(\mathbf{u})$ is positive definite, since $\lambda > 0$, and $W_F^{(k)}$ and $W_R^{(k)}$ are positive diagonal weighting matrices. For the $k = 0$ initialization (corresponding to classical Tikhonov regularization), $\nabla^2_{\mathbf{u}} T^{(0)}(\mathbf{u}) = A^T A + \lambda D^T D$ is also positive definite, since $\lambda > 0$.

The solution to $\nabla_{\mathbf{u}} T^{(k)}(\mathbf{u}) = 0$ [see (19)] is given by

$$\mathbf{u}^{(k+1)} = \left(A^T W_F^{(k)} A + \lambda D^T W_R^{(k)} D\right)^{-1} A^T W_F^{(k)} \mathbf{b}. \tag{24}$$

This solution is the unique minimum of $T^{(k)}(\mathbf{u})$ since $\nabla^2_{\mathbf{u}} T^{(k)}(\mathbf{u}) > 0$. Note that

$$\begin{aligned} T\left(\mathbf{u}^{(k+1)}\right) &\leq T^{(k)}\left(\mathbf{u}^{(k+1)}\right) \leq T^{(k)}\left(\mathbf{u}^{(k)}\right) \\ &= T\left(\mathbf{u}^{(k)}\right) \end{aligned} \tag{25}$$

so the sequence $\{T(\mathbf{u}^{(k)})\}_{k=1}^{\infty}$ is decreasing, and since $T(\mathbf{u}) > 0 \; \forall \mathbf{u}$, it is also convergent. In particular,

$$\left|T\left(\mathbf{u}^{(k+1)}\right) - T\left(\mathbf{u}^{(k)}\right)\right| \to 0 \text{ as } k \to \infty. \tag{26}$$

The limit value will be the minimizer of the generalized TV cost functional (1), if and only if $\|\nabla_{\mathbf{u}} T(\mathbf{u})|_{\mathbf{u}=\mathbf{u}^{(k)}}\|_2^2 \to 0$ as $k \to \infty$. Using a similar approach as for the derivation of (21) we may rewrite (24) as

$$\mathbf{u}^{(k+1)} - \mathbf{u}^{(k)} = -\left(\nabla^2_{\mathbf{u}} T^{(k)}(\mathbf{u})\right)^{-1} \nabla_{\mathbf{u}} T^{(k)}(\mathbf{u}). \tag{27}$$

Since functional $T^{(k)}(\mathbf{u})$ is a quadratic function, its Taylor expansion is exact, so that for $\mathbf{u} = \mathbf{u}^{(k+1)}$, $\boldsymbol{\chi}^{(k+1)} = \mathbf{u}^{(k+1)} - \mathbf{u}^{(k)}$ and using the notation $T^{(k)} = T^{(k)}(\mathbf{u}^{(k)})$, $\nabla T^{(k)} = \nabla_{\mathbf{u}} T^{(k)}(\mathbf{u})|_{\mathbf{u}=\mathbf{u}^{(k)}}$ and $\nabla^2 T^{(k)} = \nabla^2_{\mathbf{u}} T^{(k)}(\mathbf{u})|_{\mathbf{u}=\mathbf{u}^{(k)}}$, we have

$$\begin{aligned} T^{(k)}\left(\mathbf{u}^{(k+1)}\right) = T^{(k)} + {\nabla T^{(k)}}^T \boldsymbol{\chi}^{(k+1)} + \dots \\ \frac{1}{2} {\boldsymbol{\chi}^{(k+1)}}^T \nabla^2 T^{(k)} \boldsymbol{\chi}^{(k+1)}. \end{aligned} \tag{28}$$

Using (27), we have

$$\begin{aligned} &T^{(k)}\left(\mathbf{u}^{(k+1)}\right) \\ &= T^{(k)} - \frac{1}{2} {\nabla T^{(k)}}^T \left(\nabla^2 T^{(k)}\right)^{-1} \nabla T^{(k)} \\ &= T^{(k)} - \left\|\left(\nabla^2 T^{(k)}\right)^{-1/2} \nabla T^{(k)}\right\|_2^2. \end{aligned} \tag{29}$$

Finally, we may compute the norm of $\nabla_{\mathbf{u}} T(\mathbf{u})|_{\mathbf{u}=\mathbf{u}^{(k)}} = \nabla T^{(k)}$ [recall (23)]

$$\begin{aligned} \left\|\nabla T^{(k)}\right\|_2^2 &= \left\|\left(\nabla^2 T^{(k)}\right)^{1/2} \left(\nabla^2 T^{(k)}\right)^{-1/2} \nabla T^{(k)}\right\|_2^2 \\ &\leq \left\|\left(\nabla^2 T^{(k)}\right)^{1/2}\right\|_2^2 \left\|\left(\nabla^2 T^{(k)}\right)^{-1/2} \nabla T^{(k)}\right\|_2^2. \end{aligned}$$

Since weighting matrices $W_F^{(k)}$ and $\Omega^{(k)}$ are bounded [by definition, see (4) and (13)], there exists a constant ζ such that $\|(\nabla^2 T^{(k)})^{1/2}\|_2^2 \leq \zeta$, and using (25) and (29)

$$\begin{aligned} \left\|\nabla T^{(k)}\right\|_2^2 &\leq 2 \cdot C \cdot \left|T^{(k)}\left(\mathbf{u}^{(k+1)}\right) - T^{(k)}\left(\mathbf{u}^{(k)}\right)\right| \\ &\leq 2 \cdot C \cdot \left|T\left(\mathbf{u}^{(k+1)}\right) - T\left(\mathbf{u}^{(k)}\right)\right|. \end{aligned} \tag{30}$$

Therefore, $\|\nabla T^{(k)}\|_2^2 \to 0$ as $k \to \infty$ follows from (26). For the case when the generalized TV functional (1) is strictly convex (e.g., ℓ^2-TV case), the continuous function $\|\nabla_{\mathbf{u}} T(\mathbf{u})\|_2^2$ has a unique zero at $\mathbf{u}^*$ and $\mathbf{u}^{(k)} \to \mathbf{u}^*$ as $k \to \infty$. The quadratic approximation is strictly convex, with a unique minimizer, even when the generalized TV functional itself is not (e.g., ℓ^1-TV).

G. Parameter Selection

Automatic selection for the accuracy needed to solve the linear system described in Algorithms 1 and 2 (and their denoising variants) has already been introduced in Sections IV-D and IV-E. Here we focus on selecting the threshold values for the weighting matrices in the data fidelity and regularization terms [see (5) and (12), respectively]. These threshold values have a great impact in the quality of the results and in the time performance, since the weighted ℓ^2 approximations to the ℓ^p and ℓ^q norms deteriorate as these values become larger, giving poor quality results, and the linear system in (19), (21), or (22) becomes increasingly poorly conditioned as they become smaller, resulting in excessive run time with no obvious increase in the quality of the results.

For the specific case of Huber's M-estimator, $\min \sum_n \rho_\epsilon(r_n)$, where ρ is the Huber function and ϵ its threshold, and the linear ℓ^1 estimator, $\min \sum_n |r_n|$, there is a detailed analysis of the relationship between the solution (minimizer) and the parameter (threshold) ϵ of the Huber function [50], [51]. Moreover, it is shown that the minimizer is a piecewise linear function of ϵ (see [50]) and the proposed algorithms reduce the threshold ϵ at each iteration, following procedures which makes the threshold ϵ a function of the current solution [50], [51].

These procedures do not directly apply to the case of the generalized TV (1), but we notice that reducing the threshold ϵ (for Huber's M-estimator and linear ℓ^1 estimation cases) affects how many points are treated as in a standard least-squares problem and how many as in a ℓ^1 problem, and by reducing the threshold at each iteration, the probability of having too many points below the threshold is reduced as well.

This core idea can be used to adapt the threshold values for the weighting matrices [see (5) and (12)] in (17). Once a solution $\mathbf{u}$ is given, we may compute the histograms $\mathbf{h}_F = \text{HISTOGRAM}(|A\mathbf{u} - \mathbf{b}|^p)$ and $\mathbf{h}_R = \text{HISTOGRAM}(|(D_x\mathbf{u})^2 + (D_y\mathbf{u})^2|^{q/2})$ and decide that only a fixed percentage will be below the thresholds ϵ_F and ϵ_R. This can be easily accomplished by using cumulative histograms of $\mathbf{h}_F$ and $\mathbf{h}_R$ to determine the corresponding values of ϵ_F and ϵ_R respectively. Experimentally, we have found that setting the percentage less than or equal to 5% and 1%, for the regularization and fidelity terms respectively, improves the time-performance and quality of the results.

TABLE IV
ℓ^1-TV DENOISING ELAPSED TIME (S) AGAINST ALGORITHM ITERATION NUMBER, CORRESPONDING TO THE SIMULATION DESCRIBED IN FIG. 6

Iter. / Method	1	2	3	4	5
Fixed tol./thresh.	0.60	1.30	2.53	4.85	7.72
Var tol./fixed thresh.	0.49	0.90	1.59	3.22	6.15
Var tol./auto thresh.	0.47	0.94	1.53	2.34	3.71
Auto tol./thresh.	0.56	0.96	1.57	2.25	3.05

It is worth understanding how these threshold values scale with algorithm input $\mathbf{b}$. This depends on the scaling properties of $\tau_{F,\epsilon_F}(\cdot)$ and $\tau_{R,\epsilon_R}(\cdot)$, which are easily derived

$$\tau_{F,\alpha\epsilon_F}(\alpha x) = \alpha^{p-2}\tau_{F,\epsilon_F}(x)$$
$$\tau_{R,\alpha\epsilon_R}(\alpha x) = \alpha^{\frac{q-2}{2}}\tau_{R,\epsilon_R}(x).$$

It follows that scaling the input $\mathbf{b}$ (and working solution $\mathbf{u}^{(k)}$) by α corresponds to scaling ϵ_F by α, $W_F^{(k)}$ by α^{p-2}, ϵ_R by α^2, and $W_R^{(k)}$ by α^{q-2}. After substituting these properties into the final equation in Algorithm 1, and rearrangement of scaling factors, we obtain the system for scaled input $\alpha\mathbf{b}$ as

$$\hat{\mathbf{u}}^{(k+1)} = \left(A^T W_F^{(k)} A + \alpha^{q-p}\lambda D^T W_R^{(k)} D\right)^{-1} A^T W_F^{(k)}(\alpha\mathbf{b}).$$

This implies that, when $p = q$ such as for ℓ^1-TV (in which case, this is related to the contrast invariance property [9], [10]), the linear system to be inverted is invariant under scaling of the input, if thresholds and weighting matrices are properly scaled. It is also interesting to note that, when $p \neq q$, the scaling change in the linear system can be absorbed as a change in regularization parameter λ.

As an example, the Goldhill image with 10% of its pixels corrupted by salt and pepper noise (SNR 2.2 dB) was denoised (ℓ^1-TV with parameter $\lambda = 1.25$) using four different schemes: (i) fixed tolerance (10^{-5}) with $\epsilon_F = 10^{-4}$ and $\epsilon_R = 10^{-4}$, (ii) variable (exponentially decreasing between 10^{-3} and 10^{-5}) tolerance with same values for ϵ_F and ϵ_R, (iii) variable tolerance (logarithmically decreasing between 10^{-3} and 10^{-5}) and automatically adapted thresholds (as previously described), and (iv) automatically adapted tolerance (via Inexact Newton along with PCG) and automatically adapted thresholds. The reconstruction qualities and run times for these schemes are displayed in Fig. 6 and Table IV respectively. Schemes (iii) and (iv) are the fastest, but scheme (ii) has the best SNR (18.61 dB at iteration 5), and while scheme (i) has a slightly better SNR (but no noticeable visual difference) than scheme (iii), the latter is twice as fast.

The evolution of parameters ϵ_F and ϵ_R for the first 5 iterations of scheme (iii) are shown in Fig. 5. Since these values are greater than 10^{-4} [used for scheme (i) and (ii)] it should be clear that the diagonal values of the weighting matrices W_F and Ω_R have lower variance and, hence, the linear system to be solved [particularly, for denoising, (22)] is better conditioned than the resulting linear system for weighting matrices with $\epsilon_F = \epsilon_R = 10^{-4}$. The number of iterations needed to solve the linear system for schemes (ii) and (iii) are shown in Fig. 7.

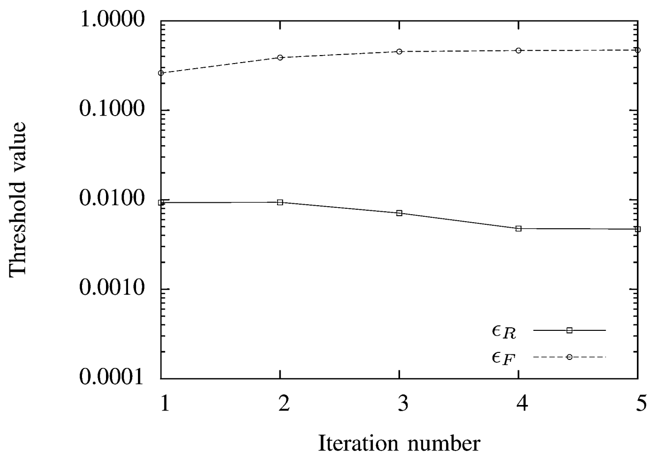

Fig. 5. Evolution of threshold value against iteration number for variable tolerance and automatically adapted thresholds [scheme (iii)], corresponding to the simulation described for Fig. 6.

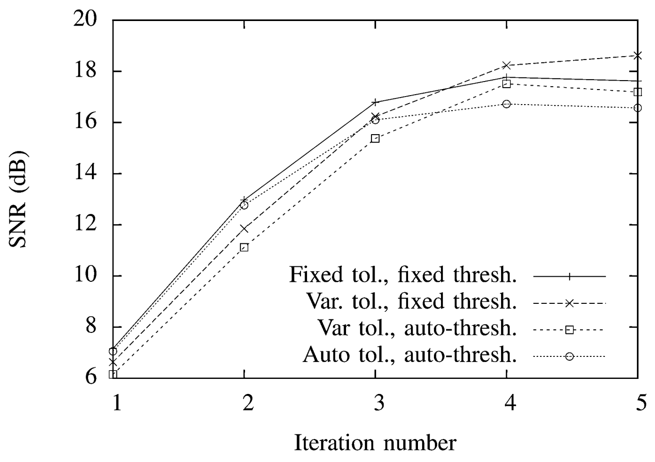

Fig. 6. ℓ^1-TV denoising SNR values with parameter $\lambda = 1.0$ for fixed, variable (exponentially decreasing between 10^{-3} and 10^{-5}) and automatically adapted CG tolerance and for fixed and automatically adapted thresholds [see (5) and (12)]. Input image was Goldhill, corrupted with salt and pepper noise (10% of its pixels, SNR 2.2 dB).

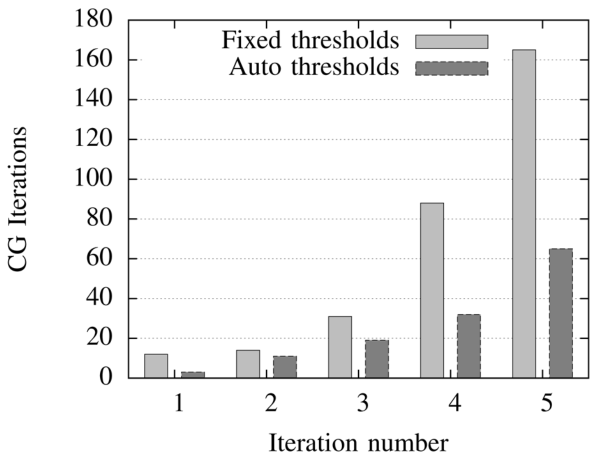

Fig. 7. Comparison of the number of CG iterations for the denoising corresponding to the simulation described for Fig. 6.

V. Conclusion

The IRN approach provides a simple but computationally efficient method for TV regularized optimization problems, including both denoising and those having a linear operator in the data fidelity term, such as deconvolution. This method is very flexible, and most of its parameters (solver accuracy and thresholds for weighting matrices) can be automatically adapted to the particular input dataset. Furthermore, it can be applied to regularized inversions with a wide variety of norms for the data fidelity and regularization terms, including the standard ℓ^2-TV, and more recently proposed ℓ^1-TV formulations, and, in particular, provides a very fast algorithm for the ℓ^1-TV case, where it is competitive with the state of the art for denoising, and is the fastest of which we are aware for more general inverse problems such as deconvolution.

Acknowledgment

The authors would like to thank M. Berndt for valuable discussions on CG preconditioning methods.

References

[1] L. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Phys. D*, vol. 60, pp. 259–268, 1992.

[2] T. Chan, S. Esedoglu, F. Park, and A. Yip, "Recent developments in total variation image restoration," in *The Handbook of Mathematical Models in Computer Vision*. New York: Springer, 2005.

[3] C. Vogel and M. Oman, "Iterative methods for total variation denoising," *SIAM J. Sci. Comput.*, vol. 17, no. 1–4, pp. 227–238, Jan. 1996.

[4] C. Vogel and M. Oman, "Fast, robust total variation-based reconstruction of noisy, blurred images," *IEEE Trans. Image Process.*, vol. 7, no. 6, pp. 813–824, Jun. 1998.

[5] T. Chan and C.-K. Wong, "Total variation blind deconvolution," *IEEE Trans. Image Process.*, vol. 7, no. 3, pp. 370–375, Mar. 1998.

[6] J. Shen and T. F. Chan, "Mathematical models for local nontexture inpaintings," *SIAM J. Appl. Math.*, vol. 62, no. 3, pp. 1019–1043, 2002.

[7] S. Alliney, "Digital filters as absolute norm regularizers," *IEEE Trans. Signal Process.*, vol. 40, no. 6, pp. 1548–1562, Jun. 1992.

[8] S. Alliney, "A property of the minimum vectors of a regularizing functional defined by means of the absolute norm," *IEEE Trans. Signal Process.*, vol. 45, no. 4, pp. 913–917, Apr. 1997.

[9] M. Nikolova, "Minimizers of cost-functions involving nonsmooth data-fidelity terms. Application to the processing of outliers," *SIAM J. Numer. Anal.*, vol. 40, no. 3, pp. 965–994, 2002.

[10] T. F. Chan and S. Esedoglu, "Aspects of total variation regularized L^1 function approximation," *SIAM J. Appl. Math.*, vol. 65, no. 5, pp. 1817–1837, 2005.

[11] M. Nikolova, "A variational approach to remove outliers and impulse noise," *J. Math. Imag. Vis.*, vol. 20, pp. 99–120, 2004.

[12] P. Rodríguez and B. Wohlberg, "An iteratively weighted norm algorithm for total variation regularization," in *Proc. Asilomar Conf. Sig. Sys. Comp.*, Pacific Grove, CA, Oct. 2006, pp. 892–896.

[13] B. Wohlberg and P. Rodríguez, "An iteratively reweighted norm algorithm for minimization of total variation functionals," *IEEE Signal Process. Lett.*, vol. 14, no. 12, pp. 948–951, Dec. 2007.

[14] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, "Inexact Newton methods," *SIAM J. Sci. Comput.*, vol. 19, no. 2, pp. 400–408, 1982.

[15] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*. Philadelphia, PA: SIAM, 1995.

[16] C. T. Kelley, *Solving Nonlinear Equations with Newton's Method*. Philadelphia, PA: SIAM, 2003.

[17] T. F. Chan, G. H. Golub, and P. Mulet, "A nonlinear primal-dual method for total variation-based image restoration," *SIAM J. Sci. Comput.*, vol. 20, no. 6, pp. 1964–1977, 1999.

[18] A. Chambolle, "An algorithm for total variation minimization and applications," *J. Math. Imag. Vis.*, vol. 20, pp. 89–97, 2004.

[19] J. F. Aujol, G. Gilboa, T. Chan, and S. Osher, "Structure-texture image decomposition—Modeling, algorithms, and parameter selection," *Int. J. Comput. Vis.*, vol. 67, no. 1, pp. 111–136, 2006.

[20] A. Chambolle, "Total variation minimization and a class of binary MRF models," in *Proc. 5th Int. Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, 2005, vol. 3757, pp. 136–152.

[21] J. Darbon and M. Sigelle, "Image restoration with discrete constrained total variation part I: Fast and exact optimization," *J. Math. Imag. Vis.*, vol. 26, no. 3, pp. 261–276, 2006.

[22] J. Darbon and M. Sigelle, "Image restoration with discrete constrained total variation part II: Levelable functions, convex priors and non-convex cases," *J. Math. Imag. Vis.*, vol. 26, no. 3, pp. 277–291, 2006.

[23] D. Goldfarb and W. Yin, Parametric Maximum Flow Algorithms for Fast Total Variation Minimization, Dept. Comput. Appl. Math., Rice Univ., Houston, TX, Tech. Rep. CAAM TR07-09, 2007.

[24] P. J. Huber, "Robust regression: Asymptotics, conjectures and monte carlo," *Ann. Statist.*, vol. 1, no. 5, pp. 799–821, 1973.

[25] Y. Li and F. Santosa, "A computational algorithm for minimizing total variation in image restoration," *IEEE Trans. Image Process.*, vol. 5, no. 6, pp. 987–995, Jun. 1996.

[26] C. Vogel, *Computational Methods for Inverse Problems*. Philadelphia, PA: SIAM, 2002.

[27] J. Bioucas-Dias, M. Figueiredo, and J. Oliveira, "Total variation image deconvolution: A majorization-minimization approach," presented at the ICASSP, Toulouse, France, May 2006.

[28] M. Figueiredo, J. Bioucas-Dias, J. Oliveira, and R. Nowa, "On total-variation denoising: A new majorization-minimization algorithm and an experimental comparison with wavelet denoising," presented at the Int. Conf. Image Process., Atlanta, GA, Oct. 2006.

[29] D. Hunter and K. Lange, "A tutorial on MM algorithms," *Amer. Statist.*, vol. 58, no. 1, pp. 30–37, 2004.

[30] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *J. Roy. Statist. Soc. B*, vol. 39, no. 1, pp. 1–38, 1977.

[31] J. Bioucas-Dias and M. Figueiredo, "A new TwIST: Two-step iterative shrinkage/thresholding algorithms for image restoration," *IEEE Trans. Image Process.*, vol. 16, no. 12, pp. 2992–3004, Dec. 2007.

[32] Y. Wang, W. Yin, and Y. Zhang, A Fast Algorithm for Image Deblurring With Total Variation Regularization, Rice Univ., Houston. TX, Tech. Rep., 2007.

[33] H. Fu, M. K. Ng, M. Nikolova, and J. L. Barlow, "Efficient minimization methods of mixed $\ell2-\ell1$ and $\ell1-\ell1$ norms for image restoration," *SIAM J. Sci. Comput.*, vol. 27, no. 6, pp. 1881–1902, 2006.

[34] D. Hochbaum, "An efficient algorithm for image segmentation, markov random fields and related problems," *J. ACM*, vol. 48, no. 4, pp. 686–701, 2001.

[35] D. Goldfarb and W. Yin, "Second-order cone programming methods for total variation based image restoration," *SIAM J. Sci. Comput.*, vol. 27, no. 2, pp. 622–645, 2005.

[36] W. Yin, Parametric Max-Flow Method and Total Variation Software library available from [Online]. Available: http://www.caam.rice.edu/wy1/ParaMaxFlow/

[37] W. Gautschi, *Numerical Analysis: An Introduction*. Cambridge, MA: Birkhauser Boston, 1997.

[38] S. Serra-Capizzano, "A note on antireflective boundary conditions and fast deblurring models," *SIAM J. Sci. Comput.*, vol. 25, no. 4, pp. 1307–1325, 2003.

[39] P. Rodríguez and B. Wohlberg, Numerical Methods for Inverse Problems and Adaptive Decomposition (NUMIPAD) [Online]. Available: http://numipad.sourceforge.net/

[40] A. E. Beaton and J. W. Tukey, "The fitting of power series, meaning polynomials, illustrated on band-spectroscopic data," *Technometrics*, no. 16, pp. 147–185, 1974.

[41] J. A. Scales and A. Gersztenkorn, "Robust methods in inverse theory," *Inv. Probl.*, vol. 4, no. 4, pp. 1071–1091, Oct. 1988.

[42] R. Wolke and H. Schwetlick, "Iteratively reweighted least squares: Algorithms, convergence analysis, and numerical comparisons," *SIAM J. Sci. Statist. Comput.*, vol. 9, no. 5, pp. 907–921, Sept. 1988.

[43] S. Ruzinsky and E. Olsen, "L_1 and L_∞ minimization via a variant of Karmarkar's algorithm," *IEEE Trans. Acoust. Speech Signal Process.*, vol. 37, no. 2, pp. 245–253, Feb. 1989.

[44] K. Bube and R. Langan, "Hybrid ℓ^1/ℓ^2 minimization with applications to tomography," *Geophysics*, vol. 62, no. 4, pp. 1183–1195, 1997.

[45] I. Gorodnitsky and B. Rao, "A new iterative weighted norm minimization algorithm and its applications," presented at the IEEE 6Th SP Workshop on Statistical Signal and Array Processing, Victoria, BC, Canada, Oct. 1992.

[46] B. D. Rao and K. Kreutz-Delgado, "An affine scaling methodology for best basis selection," *IEEE Trans. Signal Process.*, vol. 47, no. 1, pp. 187–200, Jan. 1999.

[47] S. Chen, D. Donoho, and M. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 33–61, 1998.

[48] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, 2nd ed. Philadelphia, PA: SIAM, 2000.

[49] S. Eisenstat and H. Walker, "Choosing the forcing terms in an inexact Newton method," *SIAM J. Sci. Comput.*, vol. 17, no. 1, pp. 16–32, 1996.

[50] D. I. Clark and M. R. Osborne, "Finite algorithms for Huber's M estimator," *SIAM J. Sci. Statist. Comput.*, vol. 7, no. 1, pp. 72–85, 1986.

[51] K. Madsen and H. B. Nielsen, "A finite smoothing algorithm for linear l_1 estimation," *SIAM J. Opt.*, vol. 3, no. 2, pp. 223–235, 1993.

Paul Rodríguez received the B.Sc. degree in electrical engineering from the Pontificia Universidad Católica del Perú, Lima, Peru, in 1997, and the M.Sc. and Ph.D. degrees in electrical engineering from the University of New Mexico in 2003 and 2005, respectively.

He spent two years as a postdoctoral researcher at Los Alamos National Laboratory, Losa Alamos, NM, and is currently an Associate Professor with the Department of Electrical Engineering at the Pontificia Universidad Católica del Perú. His research interests include AM-FM models, SIMD algorithms, adaptive signal decompositions, and inverse problems in signal and image processing.

Brendt Wohlberg received the B.Sc. (Hons.) degree in applied mathematics and the M.Sc. (applied science) and Ph.D. degrees in electrical engineering from the University of Cape Town, South Africa, in 1990, 1993, and 1996, respectively.

He is currently a technical staff member in the Mathematical Modeling and Analysis Group (T-7), Los Alamos National Laboratory, Los Alamos, NM. His research interests include image coding, wavelets, sparse representations, image restoration, and pattern recognition.