SIAM J. IMAGING SCIENCES
Vol. 2, No. 2, pp. 569–592

A Fast Algorithm for Edge-Preserving Variational Multichannel Image Restoration*

Junfeng Yang†, Wotao Yin‡, Yin Zhang‡, and Yilun Wang‡

Abstract. Variational models with ℓ_1-norm based regularization, in particular total variation (TV) and its variants, have long been known to offer superior image restoration quality, but processing speed remained a bottleneck, preventing their widespread use in the practice of color image processing. In this paper, by extending the grayscale image deblurring algorithm proposed in [Y. Wang, J. Yang, W. Yin, and Y. Zhang, *SIAM J. Imaging Sci.*, 1 (2008), pp. 248–272], we construct a simple and efficient algorithm for multichannel image deblurring and denoising, applicable to both within-channel and cross-channel blurs in the presence of additive Gaussian noise. The algorithm restores an image by minimizing an energy function consisting of an ℓ_2-norm fidelity term and a regularization term that can be either TV, weighted TV, or regularization functions based on higher-order derivatives. Specifically, we use a multichannel extension of the classic TV regularizer (MTV) and derive our algorithm from an extended half-quadratic transform of Geman and Yang [*IEEE Trans. Image Process.*, 4 (1995), pp. 932–946]. For three-channel color images, the per-iteration computation of this algorithm is dominated by six fast Fourier transforms. The convergence results in [Y. Wang, J. Yang, W. Yin, and Y. Zhang, *SIAM J. Imaging Sci.*, 1 (2008), pp. 248–272] for single-channel images, including global convergence with a strong q-linear rate and finite convergence for some quantities, are extended to this algorithm. We present numerical results including images recovered from various types of blurs, comparisons between our results and those obtained from the deblurring functions in MATLAB's Image Processing Toolbox, as well as images recovered by our algorithm using weighted MTV and higher-order regularization. Our numerical results indicate that the processing speed, as attained by the proposed algorithm, of variational models with TV-like regularization can be made comparable to that of less sophisticated but widely used methods for color image restoration.

Key words. half-quadratic, cross-channel, image deblurring, total variation, fast Fourier transform

AMS subject classifications. 68U10, 65J22, 65K10, 65T50, 90C25

DOI. 10.1137/080730421

1. Introduction.

The multichannel (e.g., color) image restoration problem has recently attracted much attention in the imaging community (cf. [4, 16, 40, 24, 15]). In this paper, we study an *alternating minimization algorithm* for recovering multichannel images from their blurry and noisy observations.

*Received by the editors July 17, 2008; accepted for publication (in revised form) January 20, 2009; published electronically May 6, 2009.
http://www.siam.org/journals/siims/2-2/73042.html

†Department of Mathematics, Nanjing University, 22 Hankou Road, Nanjing, Jiangsu Province, 210093, People's Republic of China (jfyang2992@gmail.com). This author's work was supported by the Chinese Scholarship Council during his visit to Rice University.

‡Department of Computational and Applied Mathematics, Rice University, 6100 Main Street, MS-134, Houston, TX 77005 (wotao.yin@rice.edu, yin.zhang@rice.edu, yilun.wang@rice.edu). The work of the second author was supported in part by NSF CAREER grant DMS-0748839 and ONR grant N00014-08-1-1101. The work of the third author was supported in part by NSF grant DMS-0811188 and ONR grant N00014-08-1-1101. The work of the fourth author was supported by NSF CAREER grant DMS-0748839.

Blurs in a multichannel image can be more complicated than those in a single-channel (e.g., grayscale) image because they can exist either within or across channels. In this paper, we consider both within- and cross-channel blurs. We assume that the underlying images have square domains and let an $n \times n$ image with m channels be denoted by $\bar{u} = [\bar{u}^{(1)}; \ldots; \bar{u}^{(m)}] \in \mathbb{R}^{mn^2}$, where $\bar{u}^{(j)} \in \mathbb{R}^{n^2}$ represents the jth channel for $j = 1, \ldots, m$, and for any two vectors v_1, v_2, notation $(v_1; v_2)$ represents the vector formed by stacking v_1 on top of v_2. An observation of $\bar{u}$ is

$$f = K\bar{u} + \omega, \tag{1.1}$$

where $f \in \mathbb{R}^{mn^2}$ has the same size and number of channels as $\bar{u}$, ω represents the additive noise, and K is a blurring operator in the form of

$$K = \begin{bmatrix} K_{11} & K_{12} & \cdots & K_{1m} \\ K_{21} & K_{22} & \cdots & K_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ K_{m1} & K_{m2} & \cdots & K_{mm} \end{bmatrix} \in \mathbb{R}^{mn^2 \times mn^2}, \tag{1.2}$$

where $K_{ij} \in \mathbb{R}^{n^2 \times n^2}$, each diagonal submatrix K_{ii} defines the blurring operator within the ith channel, and each off-diagonal matrix K_{ij}, $i \neq j$, defines how the jth channel affects the ith channel.

It is well known that recovering $\bar{u}$ from f by inverting (1.1) is an ill-posed problem because the solution is highly sensitive to the noise ω. To stabilize the recovery of $\bar{u}$, one must utilize some prior information. In such a stabilization scheme, $\bar{u}$ is obtained as the solution of

$$\min_u \Phi_{\text{reg}}(u) + \mu\Phi_{\text{fid}}(u, f), \tag{1.3}$$

where in the objective function, $\Phi_{\text{reg}}(u)$ regularizes the solution by enforcing certain prior constraints on $\bar{u}$, $\Phi_{\text{fid}}(u, f)$ measures the violation of the relation between u and its observation f (e.g., (1.1)), and μ is a positive constant that weighs the two terms in the minimization. Traditional regularization techniques such as the Tikhonov regularization [41] and the total variation regularization [33] have been carefully studied for grayscale images. In the literature, for the Gaussian noise, the common fidelity term used is

$$\Phi_{\text{fid}}(u, f) = \frac{1}{2}\|Ku - f\|_2^2 \tag{1.4}$$

corresponding to the maximum likelihood estimation of $\bar{u}$. For impulsive noise, e.g., the salt-and-pepper noise, the common fidelity term is based on the 1-norm (cf. [1, 9, 26]) instead of the square of the 2-norm as in (1.4). This paper studies the case in which ω is Gaussian and the data fidelity term $\Phi_{\text{reg}}(u, f)$ is given by (1.4).

In what follows, we give a brief review of the total variation regularization, summarize the contributions of our work, and then describe the organization of this paper.

1.1. Total variation regularization. Among all regularization techniques, the total variation regularization, first introduced in [33], is well known for preserving discontinuities in

recovered images. Let Ω be a square region in $\mathbb{R}^2$. The total variation (TV) of a grayscale image $u(x) : \Omega \to [0, 1]$ can be defined as

$$\mathrm{TV}(u) = \int_\Omega \|\nabla u\| \mathrm{d}x, \tag{1.5}$$

whenever the gradient ∇u exists, where $\|\cdot\|$ is a norm in $\mathbb{R}^2$. For more general functions, the TV is defined using a dual formulation (cf. [47]), which is equivalent to (1.5) when u is differentiable. In practical computation, a discrete form of (1.5) is always used, given by $\mathrm{TV}(u) = \sum_i \|D_i u\|$ in which $u \in \mathbb{R}^{n^2}$ represents an $n \times n$ grayscale image, $D_i u \in \mathbb{R}^2$ represents certain first-order finite differences of u at pixel i in horizontal and vertical directions, and the summation is taken over all pixels. If $\|\cdot\|$ is the 2-norm, (1.5) defines an isotropic TV, which means that (1.5) is invariant to rotation, reflection, and changing of positions of an image. If $\|\cdot\|$ is the 1-norm, (1.5) defines an anisotropic TV. Usually, the isotropic discretization is preferred over any anisotropic ones. We use the first-order forward finite differences and the 2-norm throughout this paper, and our algorithm can be easily extended for an anisotropic discretization of TV. Various methods based on TV regularization have been proposed and studied for recovering grayscale images; see, e.g., [42, 8, 10, 14].

Since many TV based algorithms have been proved effective for reducing noise and blur without smearing sharp edges for grayscale images, it is natural to extend the TV regularization to multichannel images. Several approaches for this purpose have been proposed. Among them, the simplest applies the TV regularization to each channel independently. In addition, the authors of [34, 35, 36] extended TV to deal with vector-valued images based on anisotropic diffusion and geometric active contours. Also in [3], the authors proposed the so-called color TV (see (2.2) below) and used an explicit time marching scheme similar to that in [33, 32] to minimize it. In this paper, we use a different extension of the single-channel TV, called MTV (see (2.1) below and [5, 6, 11, 13, 39]), which preserves the desirable properties of (1.5) for multichannel image reconstruction and permits very efficient solution of (1.3).

1.2. Contributions. Over the last two decades or so, variational models with ℓ_1-norm based regularization, such as TV and its variants, have been extensively studied and shown to offer sound mathematical properties and superior image restoration quality. An obstacle that has prevented these models from becoming practically viable technologies in color image processing has been the lack of adequate processing speed. The main contribution of this paper is the construction and implementation of an efficient and versatile algorithm for multichannel TV-like deblurring and denoising. In addition to TV, this algorithm can also effectively handle weighted TV, as well as high-order regularization terms. Under the periodic boundary condition, the algorithm takes advantages of fast operations such as high-dimensional shrinkage and the fast Fourier transform (FFT).

The proposed algorithm is an extension to the grayscale image restoration algorithm proposed in [43]. The strong convergence results in [43] for single-channel images are also extended to the proposed multichannel algorithm. We provide a detailed derivation of the algorithm based on the classic half-quadratic transform of Geman and Yang [18]. Although such extensions and derivation themselves are rather straightforward mathematically, the construction, implementation, and experimentation of the proposed algorithm constitute, in our view, an

important and long-overdue step towards achieving an adequate processing speed necessary for edge-preserving variational color image restoration models to become practically viable technologies.

Our algorithm was implemented in MATLAB. The numerical results include images recovered from various types of blurs, comparisons between our images and those obtained from the deblurring functions in MATLAB's Image Processing Toolbox, as well as images recovered using weighted MTV and higher-order regularization. We chose to compare with MATLAB's Image Processing Toolbox because our search failed to locate a well-documented and stable academic code for color image processing based on TV-like regularization.

1.3. Organization. The paper is organized as follows. In section 2, we focus on an extension of TV to vector-valued functions in general and discrete multichannel images in particular, and present the discrete formulation of (1.3) for deblurring. We also compare this extended TV with the "color TV" proposed in [3]. In section 3, we apply a *half-quadratic transform* to a general discrete formulation of (1.3) to derive our *alternating minimization algorithm* and present its convergence properties. Numerical results, including a comparison between the proposed algorithm and algorithms in the MATLAB Image Processing Toolbox, are presented in section 4. In addition, this section demonstrates advantages of regularization models using weighted TV and higher-order derivatives. Finally, concluding remarks are given in section 5.

2. Multichannel TV regularization problem.

Before giving the definition of multichannel TV, we introduce some notation. Let $D^{(1)}, D^{(2)} \in \mathbb{R}^{n^2\times n^2}$ be the first-order forward finite difference matrices in horizontal and vertical directions, respectively. As used in the discretized form of (1.5), $D_i \in \mathbb{R}^{2\times n^2}$ is a two-row matrix formed by stacking the ith row of $D^{(1)}$ on that of $D^{(2)}$. As in the notation $\bar{u}$ in (1.1), we let $v = (v_1; v_2) \triangleq (v_1^\top, v_2^\top)^\top$, and similarly for matrices with identical numbers of columns, e.g., $D = (D^{(1)}; D^{(2)}) \triangleq ((D^{(1)})^\top, (D^{(2)})^\top)^\top$. The spectral radius of a matrix, which is defined as the maximum magnitude of its eigenvalues, is denoted by $\rho(\cdot)$. From here on, the norm $\|\cdot\|$ refers to the 2-norm. Additional notation will be introduced as the paper progresses.

To present the definition of multichannel TV and compare it with the "color TV" regularizer (CTV) proposed in [3], we assume temporarily that u is a differentiable function defined on a square region $\Omega \subset \mathbb{R}^2$. Let $u(x) = (u^{(1)}(x); \ldots; u^{(m)}(x)) : \Omega \to \mathbb{R}^m$ be an m-channel image. It is natural to generalize (1.5) to multichannel images as follows:

$$\mathrm{MTV}(u) \triangleq \int_\Omega \|\nabla u\| \mathrm{d}x = \int_\Omega \sqrt{\|\nabla u^{(1)}\|^2 + \cdots + \|\nabla u^{(m)}\|^2}\mathrm{d}x, \tag{2.1}$$

where $\nabla u \in \mathbb{R}^{2m}$ applies ∇ to all m channels, namely, $\nabla u \triangleq (\nabla u^{(1)}; \ldots; \nabla u^{(m)})$. $\mathrm{MTV}(u)$ has already been used in the literature for red-green-blue (RGB) images; see, e.g., [6, 11, 13, 39]. For the clearness of comparison, the CTV proposed in [3] is

$$\mathrm{CTV}(u) = \sqrt{\left(\int_\Omega \|\nabla u^{(1)}\| \mathrm{d}x\right)^2 + \cdots + \left(\int_\Omega \|\nabla u^{(m)}\| \mathrm{d}x\right)^2}. \tag{2.2}$$

It is easy to see from (2.1) and (2.2) that both MTV and CTV preserve the two basic properties of (1.5), namely, (i) not overly penalizing discontinuities and (ii) rotationally invariant.

Although both MTV and CTV reduce to (1.5) for single-channel images, they are different in several aspects. First, they treat channels and pixels in different orders. While CTV computes the TV of each channel separately and then combines them, MTV first computes the variation at each pixel with respect to all channels and then sums the variations over all pixels. Second, MTV and CTV have different geometric interpretations. To illustrate this, let us assume that $\Omega = [a, b] \subset \mathbb{R}$ and $u(t) : \Omega \to \mathbb{R}^m$ is a spatial curve in $\mathbb{R}^m$. Suppose each component of $u(t)$ changes monotonically in Ω. Then, it is easy to see from (2.2) that $\mathrm{CTV}(u) = \|u(a) - u(b)\|$, the Euclidean distance between $u(a)$ and $u(b)$, and from (2.1) that $\mathrm{MTV}(u) = |\widehat{u(a)u(b)}|$, the length of the arc $\widehat{u(a)u(b)}$. Most importantly, the MTV regularization problem allows a fast alternating algorithm as we will show below, while for the CTV regularization problem the most efficient algorithm so far, to the best of our knowledge, is the lagged diffusivity (LD) method [42], which is much slower when the blurring kernel is relatively large (cf. [43]). At each iteration LD needs to solve a large linear system which is usually dense and ill-conditioned; e.g., this linear system is solved by a preconditioned conjugate gradient method in the package NUMIPAD [31]. Recently, we noted that an iterative reweighted method was proposed in [45] which minimizes a generalized TV functional with the ℓ_p-norm fidelity. Similarly to LD, at each iteration this method also needs to solve a large linear system of equations iteratively, which restricts the convergence speed. In comparison, the aforementioned alternating algorithm avoids solving such a linear system and thus is fast. Taking into account that both MTV and CTV regularization give restoration of similar quality (cf. [2]), we prefer MTV to CTV as a regularizer.

The discretized form of (2.1) is given by

$$\mathrm{MTV}(u) = \sum_i \|(I_m \otimes D_i)u\| = \sum_i \sqrt{\|D_i u^{(1)}\|^2 + \cdots + \|D_i u^{(m)}\|^2}, \tag{2.3}$$

where $u = (u^{(1)}; \ldots; u^{(m)}) \in \mathbb{R}^{mn^2}$, I_m is the identity matrix of order m, "$\otimes$" represents the Kronecker product, and $(I_m \otimes D_i)u \in \mathbb{R}^{2m}$ is the forward finite difference at pixel i for all m channels. Given $f = [f^{(1)}; \ldots; f^{(m)}] \in \mathbb{R}^{mn^2}$, which is a blurry and noisy observation, and $K = [K_{k,l}]_{k,l=1}^m \in \mathbb{R}^{mn^2 \times mn^2}$, which is a block convolution matrix, we will recover the true image $\bar{u}$ by solving

$$\min_u \sum_i \|(I_m \otimes D_i)u\| + \frac{\mu}{2}\|Ku - f\|^2. \tag{2.4}$$

For RGB images, we let $m = 3$ in (2.4). In section 3, we derive an alternating minimization algorithm based on a more general, locally weighted model:

$$\min_u \sum_i \alpha_i \|G_i u\| + \frac{\mu}{2}\|Ku - f\|^2, \tag{2.5}$$

where, at any pixel i, $G_i \in \mathbb{R}^{q \times mn^2}$ for some positive integer q; $\alpha_i > 0$ is a local weight; and $\mu > 0$. Although μ can be removed in (2.5) by rescaling the objective function, we keep it for convenience. Clearly, (2.5) reduces to (2.4) when $G_i = I_m \otimes D_i$ and $\alpha_i \equiv 1$. It is known that locally weighted TV can better preserve image textures (cf. [38, 37]), and higher-order derivatives regularization can help reduce the so-called staircasing effect sometimes found with plain TV regularization (cf. [8]). The general model (2.5) allows both features.

3. An alternating minimization algorithm. The image-quality advantages of TV over Tikhonov-like regularization are not without a price. The TV-like regularized problem (2.5) is computationally more difficult to solve due to the nondifferentiability and nonlinearity of the regularization term. Despite efforts over the years, algorithms for solving the TV-like regularization model (2.5) are still much slower than those for solving Tikhonov-like regularization models. In this section, we develop our alternating minimization algorithm based on a *half-quadratic approximation* of (2.5). This algorithm makes full use of the structures of the blurring and finite difference operators and thus is computationally highly efficient. It significantly narrows the gap between TV-like and Tikhonov-like regularization in terms of computational costs.

To avoid nondifferentiability caused by the regularization term in (2.5), we consider its smooth approximation problem

$$\min_u J(u) \triangleq \sum_i \alpha_i \, \phi_{\alpha_i}(G_i u) + \frac{\mu}{2}\|Ku - f\|^2, \tag{3.1}$$

where, for $\alpha > 0$ and $\beta \gg 0$, $\phi_\alpha(\cdot) : \mathbb{R}^q \to \mathbb{R}$ is an approximation to $\|\cdot\|$ in $\mathbb{R}^q$ defined by

$$\phi_\alpha(\mathbf{t}) = \begin{cases} \frac{\beta}{2\alpha}\|\mathbf{t}\|^2 & \text{if } \|\mathbf{t}\| \le \frac{\alpha}{\beta}, \\ \|\mathbf{t}\| - \frac{\alpha}{2\beta} & \text{otherwise.} \end{cases} \tag{3.2}$$

If α_i is large, which hints that the underlying image is supposed to be blocky around pixel i, (3.2) aims to regularize pixel i by quadratic function in a larger area. From the definition of $\phi_\alpha(\cdot)$ in (3.2), problem (2.5) is closely approximated by (3.1) when β is large. We will reformulate (3.1) as a *half-quadratic* problem in subsection 3.1 and propose to solve it by the *alternating minimization algorithm* in subsection 3.2.

3.1. Half-quadratic formulation of (3.1). In this subsection, we transform (3.1) into (3.12) below using the half-quadratic technique originally introduced by Geman and Yang in [18]. Consider the following general framework of recovering an image u from its corrupted measurements f:

$$\min_u \sum_i \phi(g_i^\top u) + \frac{\mu}{2}\|Ku - f\|^2, \tag{3.3}$$

where $g_i^\top u \in \mathbb{R}$ is a local finite difference of u, $\phi(g_i^\top \cdot)$ is convex and edge-preserving, and K is a convolution operator. Instead of solving (3.3) directly, the authors of [18] (and also of [17]) proposed to solve an equivalent problem,

$$\min_{u,b} \sum_i \Big(\psi(b_i) + Q(g_i^\top u, b_i)\Big) + \frac{\mu}{2}\|Ku - f\|^2, \tag{3.4}$$

where $Q(t,s)$ and $\psi(s)$ are chosen such that $Q(t,s)$ is quadratic in t and

$$\phi(t) = \min_{s\in\mathbb{R}}(\psi(s) + Q(t,s)) \quad \forall t \in \mathbb{R}. \tag{3.5}$$

The objective function in the right-hand side of (3.5) is called "half-quadratic" because it is quadratic in t, but not in s. The same applies to the objective function of (3.4) with

respect to u and b. In addition, (3.4) is separable in each b_i. Since (3.4) can be solved by minimizing with respect to u and b alternately, it is important to select Q and ψ that give rise to fast minimizations with respect to u and b, respectively and separately. For this purpose, two forms of half-quadratic formulations have been widely studied: the *multiplicative* form $Q(t,s) = \frac{1}{2}t^2 s$ from [17] and the *additive* form $Q(t,s) = \frac{1}{2}(t-s)^2$ from [18]. However, in the half-quadratic model based on the multiplicative form, the computation cost is higher because the Hessian of (3.4) with respect to u depends on b and thus may vary from one iteration to another.

In the following, we transform (3.1) into a half-quadratic problem based on the additive form but in a generalized manner that allows b_i in (3.4) (or s in (3.5)) to be vectors. The Hessian with respect to u of the new formulation is independent of b and has a block circulant structure, which is important because such a matrix can be diagonalized by discrete Fourier transforms; see, e.g., [20]. As such, for $\phi_\alpha(\mathbf{t})$ defined in (3.2), we need a function $\psi_\alpha(\cdot) : \mathbb{R}^q \to \mathbb{R}$ that satisfies

$$\alpha\,\phi_\alpha(\mathbf{t}) = \min_{\mathbf{s}\in\mathbb{R}^q} \left\{ \alpha\,\psi_\alpha(\mathbf{s}) + \frac{\beta}{2}\|\mathbf{s}-\mathbf{t}\|^2 \right\}. \tag{3.6}$$

For convenience, we let

$$\eta_\alpha(\mathbf{t}) = \frac{1}{2}\|\mathbf{t}\|^2 - \frac{\alpha}{\beta}\phi_\alpha(\mathbf{t}) \quad \text{and} \quad \zeta_\alpha(\mathbf{s}) = \frac{\alpha}{\beta}\psi_\alpha(\mathbf{s}) + \frac{1}{2}\|\mathbf{s}\|^2. \tag{3.7}$$

Simple manipulation shows that for ϕ_α and ψ_α to satisfy (3.6), it is necessary and sufficient to have $\eta_\alpha = \zeta_\alpha^*$, where ζ_α^* is the convex conjugate (cf. [30]) of ζ_α defined as

$$\zeta_\alpha^*(\mathbf{s}) = \sup_{\mathbf{t}\in\mathbb{R}^q} \left\{ \mathbf{s}^\top\mathbf{t} - \zeta_\alpha(\mathbf{t}) \right\}, \tag{3.8}$$

which shows how to construct ψ_α from ϕ_α through computing ζ_α from η_α.

Lemma 3.1. *For $x \in \mathbb{R}^q$ and $A \in \mathbb{R}^{p\times q}$, the subdifferential of $f(x) \triangleq \|Ax\|$ is*

$$\partial f(x) = \begin{cases} \{A^\top Ax/\|Ax\|\} & \text{if } Ax \neq 0, \\ \{A^\top h : \|h\| \le 1,\ h \in \mathbb{R}^p\} & \text{otherwise.} \end{cases} \tag{3.9}$$

The proof of Lemma 3.1 is elementary and thus omitted.

Lemma 3.2. *For $\phi_\alpha(\mathbf{t})$ defined in (3.2) and $\eta_\alpha(\mathbf{t})$ defined in (3.7), we have*

$$\eta_\alpha^*(\mathbf{s}) = \frac{\alpha}{\beta}\|\mathbf{s}\| + \frac{1}{2}\|\mathbf{s}\|^2. \tag{3.10}$$

Proof. According to (3.8), (3.7), and (3.2), we have

$$\begin{aligned} \eta_\alpha^*(\mathbf{s}) = \sup_{\mathbf{t}} \left\{ \mathbf{s}^\top\mathbf{t} - \eta_\alpha(\mathbf{t}) \right\} &= \max\left\{ \sup_{\|\mathbf{t}\|\le\alpha/\beta} \left\{ \mathbf{s}^\top\mathbf{t} - \eta_\alpha(\mathbf{t}) \right\}, \sup_{\|\mathbf{t}\|>\alpha/\beta} \left\{ \mathbf{s}^\top\mathbf{t} - \eta_\alpha(\mathbf{t}) \right\} \right\} \\ &= \max\left\{ (\alpha/\beta)\|\mathbf{s}\|, \sup_{\|\mathbf{t}\|>\alpha/\beta} \left\{ \mathbf{s}^\top\mathbf{t} - (\|\mathbf{t}\| - \alpha/\beta)^2/2 \right\} \right\}. \end{aligned} \tag{3.11}$$

If $\mathbf{s} = 0$, it is obvious that $\eta_\alpha^*(0) = 0$ and (3.10) holds. In what follows, we assume $\mathbf{s} \neq 0$. The inside supremum on the above is attained only if there exists some $\mathbf{t}$ that satisfies $\|\mathbf{t}\| > \alpha/\beta$ such that the subdifferential of its argument contains the origin, which in light of (3.9) with A being the identity is equivalent to $\mathbf{s} = \mathbf{t} - \alpha\mathbf{t}/(\beta\|\mathbf{t}\|)$. Since $\mathbf{s} \neq 0$, both conditions are satisfied by defining $\mathbf{t} = \gamma\mathbf{s}$ with $\gamma = 1 + \alpha/(\beta\|s\|)$. First, $\|\mathbf{t}\| = \gamma\|s\| = \|s\| + \alpha/\beta > \alpha/\beta$. Second, from $\mathbf{t} = \gamma\mathbf{s}$, we have $\mathbf{s} = \mathbf{t} - \alpha\mathbf{s}/(\beta\|\mathbf{s}\|) = \mathbf{t} - \alpha\mathbf{t}/(\beta\|\mathbf{t}\|)$, where the second equality comes from $\mathbf{t}/\|\mathbf{t}\| = \mathbf{s}/\|\mathbf{s}\|$. Plugging $\mathbf{t} = \gamma\mathbf{s}$ into the inside supremum in (3.11), we get (3.10). ∎

Given Lemma 3.2, we are able to find $\psi_\alpha(\cdot)$ that satisfies (3.6) in a simple way. It is easy to see from (3.2) and (3.7) that $\eta_\alpha(\mathbf{t})$ is proper, continuous, and convex. Therefore, the self-biconjugacy holds for $\eta_\alpha(\mathbf{t})$; i.e., $(\eta_\alpha^*)^* = \eta_\alpha$ (cf. [30]). As such, the requirement (3.6) or, equivalently, $\eta_\alpha = \zeta_\alpha^*$ is fulfilled by letting $\zeta_\alpha = \eta_\alpha^*$. Comparing the definition of $\zeta_\alpha(\cdot)$ with (3.10), letting $\psi_\alpha(\mathbf{s}) \equiv \|\mathbf{s}\|$ will satisfy (3.6). As such, the approximation problem (3.1) becomes an equivalent augmented problem

$$\min_{u,\mathbf{w}} \mathcal{J}(u, \mathbf{w}) = \sum_i \left\{ \alpha_i \|\mathbf{w}_i\| + \frac{\beta}{2}\|\mathbf{w}_i - G_i u\|^2 \right\} + \frac{\mu}{2}\|Ku - f\|^2, \tag{3.12}$$

where $\mathbf{w}_j \in \mathbb{R}^q$, $j = 1, \ldots, n^2$, and $\mathbf{w} = [\mathbf{w}_1; \mathbf{w}_2; \ldots; \mathbf{w}_{n^2}] \in \mathbb{R}^{qn^2}$. Although derived from the half-quadratic framework and the theory of conjugate functions, (3.12) is simply a splitting and penalty formulation of the original problem (2.5) which can be explained as follows. By introducing a collection of auxiliary variables $\{\mathbf{w}_i : i = 1, \ldots, n^2\}$, problem (2.5) is easily transformed into an equivalent constrained problem

$$\min_{u,\mathbf{w}} \left\{ \sum_i \alpha_i \|\mathbf{w}_i\| + \frac{\mu}{2}\|Ku - f\|^2 : \ \mathbf{w}_i = G_i u, \ i = 1, \ldots, n^2 \right\}. \tag{3.13}$$

By comparing (3.13) with (3.12), it is easy to see that (3.12) is nothing but a quadratic penalty method for (3.13). Below our analysis focuses on (3.12).

3.2. Alternating minimization. Now we are ready to apply the alternating minimization to (3.12). On the one hand, there is no interaction between different $\mathbf{w}_i$'s in (3.12), so minimizing with respect to $\mathbf{w}$ for fixed u reduces to solving a collection of low-dimensional problems

$$\min_{\mathbf{w}_i \in \mathbb{R}^q} \alpha_i \|\mathbf{w}_i\| + \frac{\beta}{2}\|\mathbf{w}_i - G_i u\|^2, \quad i = 1, \ldots, n^2. \tag{3.14}$$

On the other hand, minimizing with respect to u for fixed $\mathbf{w}$ becomes

$$\min_u \frac{\beta}{2} \sum_i \|\mathbf{w}_i - G_i u\|^2 + \frac{\mu}{2}\|Ku - f\|^2,$$

which is a least squares problem equivalent to

$$\left(\sum_i G_i^\top G_i + \frac{\mu}{\beta} K^\top K \right) u = \sum_i G_i^\top \mathbf{w}_i + \frac{\mu}{\beta} K^\top f. \tag{3.15}$$

We note that a similar splitting technique was also used in [19] where, unlike in our approach, a continuation scheme was not used; instead, Bregman iterations (see [28, 46] for more details) were used to obtain better approximate solutions.

To simplify analysis, we introduce some additional notation. For $j = 1, \ldots, q$, let $G^{(j)} \in \mathbb{R}^{n^2 \times mn^2}$ be the matrix formed by staking the jth rows of $G_1, G_2, \ldots, G_{n^2}$. Denote

$$G \triangleq \begin{pmatrix} G^{(1)} \\ \vdots \\ G^{(q)} \end{pmatrix} \in \mathbb{R}^{qn^2 \times mn^2} \quad \text{and} \quad \mathbf{W} \triangleq \begin{pmatrix} \mathbf{w}_1^\top \\ \vdots \\ \mathbf{w}_{n^2}^\top \end{pmatrix} \triangleq [w_1, w_2, \ldots, w_q] \in \mathbb{R}^{n^2 \times q}; \tag{3.16}$$

namely, $w_j \in \mathbb{R}^{n^2}$ is the jth column of $\mathbf{W}$ and formed by stacking the jth components of $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{n^2}$. Furthermore, let $w = (w_1; \ldots; w_q) = \mathbf{W}(:) \in \mathbb{R}^{qn^2}$ be the vectorization of $\mathbf{W}$ (which is just a reordering of $\mathbf{w}$). Given this notation, (3.15) can be rewritten as

$$\left(G^\top G + \frac{\mu}{\beta} K^\top K\right) u = \sum_{j=1}^{q} (G^{(j)})^\top w_j + \frac{\mu}{\beta} K^\top f = G^\top w + \frac{\mu}{\beta} K^\top f. \tag{3.17}$$

In what follows, we show that the problems in (3.14) admit closed form solutions and (3.17) can also be solved easily as long as G defined in (3.16) has certain special structures.

Lemma 3.3. *For any $\alpha, \beta > 0$ and $\mathbf{t} \in \mathbb{R}^q$, the minimizer of*

$$\min_{\mathbf{s} \in \mathbb{R}^q} \alpha \|\mathbf{s}\| + \frac{\beta}{2} \|\mathbf{s} - \mathbf{t}\|^2 \tag{3.18}$$

is given by

$$\mathbf{s}(\mathbf{t}) = \max\left\{ \|\mathbf{t}\| - \frac{\alpha}{\beta}, 0 \right\} \frac{\mathbf{t}}{\|\mathbf{t}\|}, \tag{3.19}$$

where we follow the convention $0 \cdot (0/0) = 0$.

Proof. Since the objective function is strictly convex, bounded below, and coercive, problem (3.18) has unique minimizer $\mathbf{s}$. According to the optimality condition for convex optimization, the subdifferential of the objective function at the minimizer should contain the origin. In light of (3.9) with $A = I$, $\mathbf{s}$ must satisfy

$$\begin{cases} \alpha \mathbf{s}/\|\mathbf{s}\| + \beta(\mathbf{s} - \mathbf{t}) = 0 & \text{if } \mathbf{s} \neq 0, \\ \beta \|\mathbf{t}\| \leq \alpha & \text{otherwise.} \end{cases} \tag{3.20}$$

If $\mathbf{s} \neq 0$, it is obvious from the first equation in (3.20) that $\mathbf{t} = \gamma \mathbf{s}$ with $\gamma = 1 + \alpha/(\beta \|\mathbf{s}\|)$, from which we have $\|\mathbf{t}\| = \|\mathbf{s}\| + \alpha/\beta$ and $\mathbf{s}/\|\mathbf{s}\| = \mathbf{t}/\|\mathbf{t}\|$. Thus,

$$\mathbf{s} = \|\mathbf{s}\| \cdot \frac{\mathbf{s}}{\|\mathbf{s}\|} = \|\mathbf{s}\| \cdot \frac{\mathbf{t}}{\|\mathbf{t}\|} = \left(\|\mathbf{t}\| - \frac{\alpha}{\beta}\right) \frac{\mathbf{t}}{\|\mathbf{t}\|}. \tag{3.21}$$

Furthermore, $\mathbf{s} = 0$ if and only if $\|\mathbf{t}\| \leq \alpha/\beta$. Combining this with (3.21), we obtain (3.19). ∎

According to Lemma 3.3, the problems in (3.14) have unique solutions given explicitly by

$$\mathbf{w}_i = \max\left\{\|G_i u\| - \frac{\alpha_i}{\beta}, 0\right\}\frac{G_i u}{\|G_i u\|}, \quad i = 1, \ldots, n^2. \tag{3.22}$$

Now we show that (3.17) can also be easily solved under many circumstances. For simplicity, we assume that u is an RGB color image, and the general case can be discussed similarly. In this case, we have $u = (u^r; u^g; u^b) \in \mathbb{R}^{3n^2}$ and $K = [K_{k,l}]_{k,l=1}^3 \in \mathbb{R}^{3n^2\times 3n^2}$. Also, we assume that $G_i = I_3 \otimes D_i \in \mathbb{R}^{6\times 3n^2}$; namely, $G_i u$ contains only the first-order finite differences of u at pixel i. For higher-order finite differences, the following argument also applies. Recall that $D = (D^{(1)}; D^{(2)})$ and D_i is a two-row matrix formed by stacking the ith rows of $D^{(1)}$ and $D^{(2)}$. Following the notation in (3.16), equations (3.17) reduce to

$$\left(I_3 \otimes (D^\top D) + \frac{\mu}{\beta}K^\top K\right)u = (I_3 \otimes D)^\top w + \frac{\mu}{\beta}K^\top f. \tag{3.23}$$

Under the periodic boundary conditions for u, $D^{(1)}$, $D^{(2)}$, $K_{i,j}$, $i, j = 1, 2, 3$, and their transpose matrices are all block circulant (see [25], for example). Therefore, each block in the coefficient matrix of (3.23) can be diagonalized by the two-dimensional discrete Fourier transform $\mathbf{F}$. Applying $I_3 \otimes \mathbf{F}$ to both sides of (3.23) yields

$$\begin{bmatrix} \Lambda_{11} & \Lambda_{12} & \Lambda_{13} \\ \Lambda_{21} & \Lambda_{22} & \Lambda_{23} \\ \Lambda_{31} & \Lambda_{32} & \Lambda_{33} \end{bmatrix}\begin{pmatrix} \mathbf{F}(u^r) \\ \mathbf{F}(u^g) \\ \mathbf{F}(u^b) \end{pmatrix} = \begin{pmatrix} \mathbf{F}\left[(D^{(1)})^\top w_1 + (D^{(2)})^\top w_2\right] \\ \mathbf{F}\left[(D^{(1)})^\top w_3 + (D^{(2)})^\top w_4\right] \\ \mathbf{F}\left[(D^{(1)})^\top w_5 + (D^{(2)})^\top w_6\right] \end{pmatrix} + \frac{\mu}{\beta}(I_3 \otimes \mathbf{F})[K^\top f], \tag{3.24}$$

where Λ_{ij}, $i, j = 1, 2, 3$, are diagonal matrices. For any $M \in \mathbb{R}^{n^2\times n^2}$, we let $\mathcal{F}(M) = \mathbf{F}M\mathbf{F}^{-1}$. It is easy to see that each Λ_{ij} is a linear combination of $\mathcal{F}(D^{(1)})^*\mathcal{F}(D^{(1)})$, $\mathcal{F}(D^{(2)})^*\mathcal{F}(D^{(2)})$, and $\mathcal{F}(K_{k,l})^*\mathcal{F}(K_{p,q})$, $k, l, p, q = 1, 2, 3$, where "*" represents conjugate transpose. Therefore, at each iteration the block diagonalized coefficient matrix of (3.24) can be easily obtained by combining $\mathcal{F}(D^{(1)})^*\mathcal{F}(D^{(1)})$, $\mathcal{F}(D^{(2)})^*\mathcal{F}(D^{(2)})$, and $\mathcal{F}(K_{k,l})^*\mathcal{F}(K_{p,q})$, $k, l, p, q = 1, 2, 3$, which, along with the constant vector on the right-hand side of (3.24), need to be computed only once before the iterations. At each iteration, (3.23) is solved in three steps. First, compute the right-hand side vector of (3.24) which mainly costs several finite differences on the auxiliary variables and 3 FFTs. Second, solve a block diagonal system of the form (3.24) to obtain $\mathbf{F}(u^\sigma)$, for $\sigma = r, g, b$, which can be easily done by the Gaussian elimination method. Third, apply $\mathbf{F}^{-1}$ to each $\mathbf{F}(u^\sigma)$ to get the updated u. The total number of FFTs required is 6 (including 3 inverse FFTs).

Alternatively, under the Neumann boundary conditions and assuming that all the blurring kernels are symmetric, the forward and inverse FFTs shall be replaced by the forward and inverse discrete cosine transforms (DCTs), respectively (cf. [25]). In this case, a longer CPU time is needed for solving (3.17) because DCT is generally 3–4 times slower than FFT in MATLAB. In our experiments, we assumed the periodic boundary conditions and used FFTs. We point out that when Dirichlet boundary conditions are assumed, equations (3.17) may not be easily solved by fast transforms because in this case the coefficient matrix is usually an ill-conditioned block Toeplitz matrix with Toeplitz blocks. For large-sized images blurred by

kernels with relatively small supports, different boundary conditions such as Dirichlet and periodic boundary conditions lead to similar solutions of problem (2.4).

Now we are ready to formally present our algorithm. For a fixed β, (3.12) is solved by an alternating minimization scheme given below.

Algorithm 1. *Input* f, K, $\mu > 0$, $\beta > 0$, *and* $\alpha_i > 0$, $i = 1, \ldots, n^2$. *Initialize* $u = f$.

While *"not converged,"* **Do**

1) *Compute* $\mathbf{w}$ *by* (3.22) *for given* u.
2) *Solve* (3.17) *to get* u *for given* $\mathbf{w}$.

End Do

The stopping criterion is specified in the next subsection. A practical implementation of Algorithm 1 including the setting of parameters and a continuation scheme is presented in section 4.

3.3. Optimality conditions and stopping criterion. Since the objective function in (3.12) is convex, $(\mathbf{w}, u)$ solves (3.12) if and only if the subdifferential of the objective function at $(\mathbf{w}, u)$ contains the origin. This gives rise to the following optimality conditions in light of Lemma 3.1:

$$\begin{cases} \alpha_i \mathbf{w}_i / \|\mathbf{w}_i\| + \beta(\mathbf{w}_i - G_i u) = 0, & i \in I_1 \triangleq \{i : \mathbf{w}_i \neq \mathbf{0}\}, \\ \beta \|G_i u\| \leq \alpha_i, & i \in I_2 \triangleq \{i : \mathbf{w}_i = \mathbf{0}\}, \end{cases} \tag{3.25}$$

$$\beta G^\top (Gu - w) + \mu K^\top (Ku - f) = 0. \tag{3.26}$$

Our stopping criterion for Algorithm 1 is based on the optimality conditions (3.25) and (3.26). Let

$$\begin{cases} r_1(i) \triangleq (\alpha_i \mathbf{w}_i / \|\mathbf{w}_i\|)/\beta + \mathbf{w}_i - G_i u, & i \in I_1, \\ r_2(i) \triangleq \|G_i u\| - \alpha_i/\beta, & i \in I_2, \\ r_3 \triangleq \|\beta G^\top (Gu - w) + \mu K^\top (Ku - f)\| / \|\beta G^\top w + \mu K^\top f\|, \end{cases} \tag{3.27}$$

where I_1 and I_2 are defined in (3.25). Algorithm 1 is terminated once

$$Res \triangleq \max \left\{ \max_{i \in I_1} \{\|r_1(i)\|\}, \max_{i \in I_2} \{r_2(i)\}, r_3 \right\} \leq \epsilon \tag{3.28}$$

is met, where *Res* measures the total residual and $\epsilon > 0$ is a prescribed tolerance. We note that in practice the relative residue r_3 defined in (3.27) is always quite small because we solved (3.26) exactly by FFTs for fixed w. Combining (3.25) and (3.26) to eliminate w, we can derive

$$\sum_{i \in I_1} \alpha_i G_i^\top \frac{G_i u}{\|G_i u\|} + \sum_{i \in I_2} G_i^\top h_i + \mu K^\top (Ku - f) = 0, \tag{3.29}$$

where $h_i \triangleq \beta G_i u$ satisfies $\|h_i\| \leq \alpha_i$. Let u^* be any solution of (2.5). Define $I_1^* = \{i, G_i u^* \neq 0\}$ and $I_2^* = \{1, \ldots, n^2\} \setminus I_1^*$. Then, from Lemma 3.1, for all $i \in I_2^*$ there exist $h_i^* \in \mathbb{R}^q$ satisfying $\|h_i^*\| \leq \alpha_i$ such that

$$\sum_{i \in I_1^*} \alpha_i G_i^\top \frac{G_i u^*}{\|G_i u^*\|} + \sum_{i \in I_2^*} G_i^\top h_i^* + \mu K^\top (Ku^* - f) = 0. \tag{3.30}$$

Equation (3.29) differs from (3.30) only in the index sets over which the summations are taken. As β increases, I_1 will approach I_1^*. In subsection 3.4, we present the convergence properties of Algorithm 1.

3.4. Convergence results. The convergence of the quadratic penalty method as the penalty parameter goes to infinity is well known (see Theorem 17.1 in [27], for example). That is, as $\beta \to \infty$, the solution of (3.1) or (3.12) converges to that of (2.5). However, in practice a sufficiently large value for β should be adequate. We will specify how to choose β empirically in section 4. In this subsection, we present convergence results of Algorithm 1 for a fixed β without proofs since these results are rather straightforward generalizations of the results in [43] to higher dimensions. First, we present a technical assumption and necessary notation.

Assumption 1. *$\mathcal{N}(K) \cap \mathcal{N}(G) = \{0\}$, where $\mathcal{N}(\cdot)$ is the null space of a matrix.*

Define

$$M = G^\top G + \frac{\mu}{\beta} K^\top K \quad \text{and} \quad T = GM^{-1}G^\top. \tag{3.31}$$

Assumption 1 ensures that M is nonsingular, and therefore T is well defined. It is not difficult to show that $\rho(T) \le 1$. We will make use of the following two index sets:

$$L = \left\{ i \ : \ \|G_i u^*\| < \frac{\alpha_i}{\beta} \right\} \quad \text{and} \quad E = \{1, \dots, n^2\} \setminus L. \tag{3.32}$$

Denote $w_E = ((w_1)_E; \dots; (w_q)_E)$, where $(w_j)_E$ is the subvector of w_j with components in E. For $k, l = 1, \dots, q$, let $B^{(k,l)} = G^{(k)} M^{-1} (G^{(l)})^\top$ and $B_{EE}^{(k,l)} = [B_{i,j}^{(k,l)}]_{i,j \in E}$ be the minor of $B^{(k,l)}$ with indices in E. From the definition of T, $T = [B^{(k,l)}]_{k,l=1}^q$. Let $T_{EE} = [B_{EE}^{(k,l)}]_{k,l=1}^q$ be a minor of T. Similar notation applies to $(T^2)_{EE}$. Now we are ready to present the convergence results.

Theorem 3.4 (convergence). *Under Assumption 1, the sequence $\{(w^k, u^k)\}$ generated by Algorithm 1 from any starting point (w^0, u^0) converges to a solution (w^*, u^*) of (3.12).*

Theorem 3.5 (finite convergence). *Under Assumption 1, the sequence $\{(w^k, u^k)\}$ generated by Algorithm 1 from any starting point (w^0, u^0) satisfies $\mathbf{w}_i^k = \mathbf{w}_i^* = 0$ for all $i \in L$, for all but a finite number of iterations that does not exceed $\|w^0 - w^*\|^2/\omega^2$, where*

$$\omega \triangleq \min_{i \in L} \left\{ \frac{\alpha_i}{\beta} - \|h_i(w^*)\| \right\} > 0.$$

Theorem 3.6 (q-linear convergence). *Let M and T be defined as in (3.31). Under Assumption 1, the sequence $\{(w^k, u^k)\}$ generated by Algorithm 1 satisfies*

1. $\|w_E^{k+1} - w_E^*\| \le \sqrt{\rho((T^2)_{EE})} \|w_E^k - w_E^*\|$;
2. $\|u^{k+1} - u^*\|_M \le \sqrt{\rho(T_{EE})} \|u^k - u^*\|_M$

for all k sufficiently large, where $\|v\|_M^2 \triangleq v^\top M v$.

Theorem 3.6 states that Algorithm 1 converges q-linearly at a rate depending on the spectral radii of the submatrices T_{EE} and $(T^2)_{EE}$ rather than on that of the whole matrix T. Since $\rho(T) \le 1$ and T_{EE} is a minor of T, it holds that $\rho(T_{EE}) \le \rho(T) \le 1$. Similarly,

$\rho((T^2)_{EE}) \le \rho(T^2) \le 1$. As pointed out in [43], $\rho(T_{EE})$ is often much smaller than $\rho(T)$ in practice. Since convergence results for the quadratic penalty method were typically stated in terms of iteration matrices as a whole, our results, stated in terms of submatrices, are stronger thanks to the sparsity in the solutions.

4. Numerical experiments. In this section, we present numerical results to show the efficiency of the proposed alternating minimization algorithm in recovering multichannel images. Specifically, we experimented with different regularization approaches on recovering several RGB images with different blurs and noise levels. The first set of tests involved different blurs including both within-channel and cross-channel blurs. The second set of tests involved different regularization functions including weighted TV and measures of higher-order derivatives.

4.1. Test images, platform, and practical implementation. We tested several images including Lena (512×512), Rose (303×250), and Sunset (338×460). Image Lena has a nice mixture of flat regions, shading area, textures, and other details. Rose and Sunset were used in weighted TV regularization and higher-order derivative regularization problems. We implemented Algorithm 1 in MATLAB and generated blurring effects using the MATLAB function "`imfilter`" with the periodic boundary conditions. The experiments were performed under Windows Vista Premium and MATLAB v7.6 (R2008a) running on a Lenovo laptop with an Intel Core 2 Duo CPU at 1.8 GHz and 2 GB of memory.

As is usually done, the quality of restoration is measured by the signal-to-noise ratio (SNR)

$$\mathrm{SNR} \triangleq 10 * \log_{10} \frac{\|\bar{u} - \mathbf{E}(\bar{u})\|^2}{\|\bar{u} - u\|^2},$$

where $\bar{u}$ is the original image, $\mathbf{E}(\bar{u})$ is the mean intensity value of $\bar{u}$, and u is the restored image. We tested three kinds of blurring kernels provided by MATLAB: motion blur, which shifts image pixels linearly in a certain direction and with a fixed length; Gaussian blur, which is rotationally symmetric Gaussian lowpass filter; and average blur, which smooths an image by replacing the intensity value of pixel i by the average intensity value of the pixels within a certain neighborhood centered at pixel i. For simplicity, let the motion blur with a motion length `len` and an angle `theta` in the counterclockwise direction be denoted by $\mathrm{M}(\texttt{len}, \texttt{theta})$. Similarly, the Gaussian blur with a square support size `hsize` and a standard deviation `sigma` is denoted by $\mathrm{G}(\texttt{hsize}, \texttt{sigma})$, and the average blur with a square support size `hsize` by $\mathrm{A}(\texttt{hsize})$. For all experiments, we first rescaled the intensity values of original images to $[0, 1]$, then blurred them and added Gaussian noise with mean zero and different standard deviation (`std`) to the blurry images. To determine a good pair of parameters in (3.12), we fixed $\alpha_i \equiv 1$ and tested our code on a series of combinations of β and μ. Specially, we tested on image Rose (which has a relatively small size) with different blurs and noise levels using a relatively strict stopping criterion: $\epsilon = 10^{-3}$ in (3.28). Figure 1 gives the recovered SNR results for different combinations of β and μ, where we used cross-channel blurring with the kernels generated in three steps.

Step 1. Generate 9 kernels:

$$\{\mathrm{M}(11, 45), \mathrm{M}(21, 90), \mathrm{M}(41, 135), \mathrm{G}(7, 5), \mathrm{G}(9, 5), \mathrm{G}(11, 5), \mathrm{A}(13), \mathrm{A}(15), \mathrm{A}(17)\}.$$

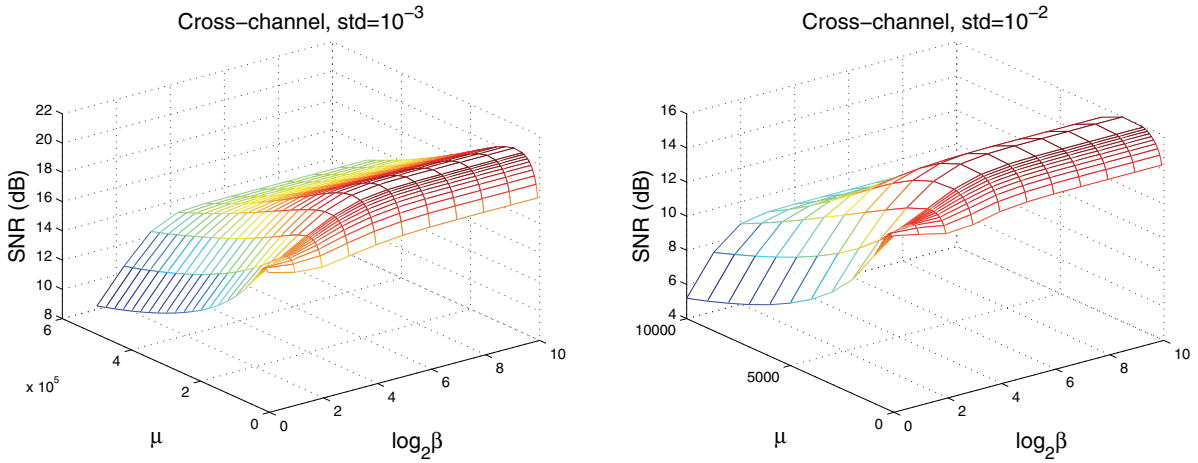

Figure 1. *SNR results recovered from cross-channel blurring for different β and μ. Left:* `std` $= 10^{-3}$*; SNR of the blurry and noisy image is* $9.21dB$*. Right:* `std` $= 10^{-2}$*; SNR of the blurry and noisy image is* $8.96dB$*.*

Step 2. Randomly assign the above 9 kernels to $\{H_{11}, H_{12}, H_{13}, H_{21}, H_{22}, H_{23}, H_{31}, H_{32}, H_{33}\}$.

Step 3. Multiply diagonal kernels by 0.8 and off-diagonal kernels by 0.1.

The left- and right-hand side plots in Figure 1 are, respectively, the SNR results with additive noise `std` of 10^{-3} and 10^{-2}.

As can be seen from Figure 1, in order to obtain good SNRs β does not need to be extremely large for any fixed μ. Specifically, a β value of 2^7 is sufficiently large to get an SNR that is very close to the highest one for each μ. For $\beta \geq 2^7$, the SNRs remain almost constant. For larger noise and/or within-channel blurring, similar phenomena were observed. Hence, letting β be larger than 2^7 will merely increase computational cost but not solution quality. Therefore, we set $\beta = 2^7$ by default in Algorithm 1.

The above approach of generating cross-channel kernels was used in all of our experiments involving cross-channel blurs. In Step 2, randomness is introduced to avoid possible bias. We point out that the same set of kernels generated in Step 1 may result in blurry images of different degrees because kernels may be assigned to different cross-channel positions in Step 2. We have observed that the performance of our algorithm is indifferent to such kernel configurations. In our experiments presented in the following subsections, we tested kernels of different sizes and at different locations to validate the above assertions. In Step 3, we choose weights added to the kernels to be diagonally dominant considering the fact that within-channel blurs are usually stronger than cross-channel blurs. Similar methods for choosing kernel weights are used in the literature; see, e.g., [15, 16]. As a matter of fact, our algorithm converges well for weights selected in a very broad range. The only problematic case is when all weights assigned to the 9 kernels are equal, causing singularity of $I_3 \otimes D^\top D + (\mu/\beta)K^\top K$. To see this, let p and q be positive integers, let $h \in \mathbb{R}^{p\times q}$ be any convolution kernel satisfying $\sum_i \sum_j h_{ij} = 1$, where h_{ij} is the component of h at the ith row and jth column, and let the corresponding convolution matrix be denoted by $H \in \mathbb{R}^{n^2\times n^2}$. Then, under the periodic boundary condition, the sum of each column of H is always 1. Since the two-dimensional

discrete Fourier transform matrix $\mathbf{F}$ of order $n^2 \times n^2$ satisfies $\mathbf{F}_{1j} \equiv 1/n$ for $j = 1, 2, \ldots, n^2$, it is easy to check that the $(1,1)$ element of the spectral decomposition $\mathbf{F}H\mathbf{F}^\top$ is always 1. Similarly, under the periodic boundary condition the $(1,1)$ element of $\mathbf{F}D^{(j)}\mathbf{F}^\top$ is 0, $j = 1, 2$. Based on these two facts and further considering that all three kinds of kernels we tested satisfy $\sum_i \sum_j h_{ij} = 1$, it is not difficult to verify that, when equal weights are assigned to all kernels, the elements located at the positions $(1,1)$ of all Λ_{ij} in (3.24) are equal, $i, j = 1, 2, 3$. In this case, Assumption 1 is violated and (3.23) does not have a unique solution. Equal weights are rarely used in image processing. When they are used in a rare case, we can overcome nonuniqueness by seeking a least squares solution to the 3×3 linear subsystem in (3.24).

To accelerate convergence, we also implemented a continuation scheme for β in which β changes from a small value step by step to the default value as the algorithm proceeds. Continuation techniques are widely used with penalty methods and, for our problem, continuation on β is also theoretically well justified by Theorem 3.6. From the definitions of L and E, it is likely that smaller β yield smaller E and thus fast convergence. As such, earlier subproblems with smaller penalty parameters can be solved quickly, and the later subproblems can also be solved relatively quickly with warm starts from previous solutions. For experimental results on how this continuation scheme improves the overall convergence speed, see, e.g., [43, 21]. The best choice of μ depends on the noise level `std`. For too large values of μ, the restored images will remain noisy. On the other hand, for too small values of μ, some fine details in the images will get lost. In our tests with noise level `std` being 10^{-3}, $\mu = 5 \times 10^4$ seems suitable, as can be seen from Figure 1. Thus, we set $\mu = 5 \times 10^4$ by default for `std` $= 10^{-3}$. For larger noise, we first determined a good μ for the standard TV/L^2 model (2.4) based on experiments and then tested it with different regularization.

The implementation of Algorithm 1 includes two loops: the outer loop increases β from 1 to 2^7, and the inner loop solves (3.12) to a prescribed accuracy for each fixed β. We simply doubled β after every outer iteration and stopped the inner iteration when $Res \le 0.05$ was satisfied in (3.28). Of course this algorithm framework can be more flexible in terms of updating β and stopping the inner iterations, but the above simple implementation already worked very well. Following [43], we give the name fast total variation deconvolution, or FTVd, to Algorithm 1 with the above continuation strategy.

It is interesting to observe from Figure 1 that, compared to the blurry and noisy image, a small β is sufficient to give a nearly optimal SNR when μ is selected appropriately. For example, in the left plot of Figure 1 the SNR result is 17.78dB when $(\mu, \beta) = (10^4, 1)$, which is a little lower than the highest SNR reachable (20.3dB given by $(\mu, \beta) = (5 \cdot 10^4, 2^7)$) but much better than the SNR of the blurry and noisy image (9.21dB). Similarly, in the right plot of Figure 1, the result 14.39dB of $(\mu, \beta) = (2 \cdot 10^2, 1)$ is much better than the SNR of the blurry and noisy image (8.96dB), although it is a little lower than the highest SNR (16dB given by $(\mu, \beta) = (2 \cdot 10^3, 2^7)$). For larger noise and/or different blurs, similar phenomena were observed. Since we increase β from 1 to a prefixed value, there will be only one outer iteration if we set the final (target) value of β to 1. In this case, usually one or two inner iterations are sufficient to obtain a solution of (3.12) with high accuracy. Although the resulting SNR is a little lower, the CPU time consumed is much less in this case. Therefore, in real-time imaging applications where speed is crucial, it is a good choice to assign β a small value with a suitable μ.

4.2. Comparisons with MATLAB functions. In this subsection, we compare the images recovered by FTVd with those recovered by functions from the MATLAB Image Processing Toolbox. We note that although there have been some discussions in the literature on color image deconvolution based on TV regularization (see, e.g., [3, 6, 11, 13, 39]), to the best of our knowledge there is not any released package ready to solve either (2.4) or the "color TV" regularization problem. We noticed the recent work [5] where the authors discussed solving an approximation problem of (2.4) based on Chambolle's dual formulation [7] of TV. Currently, their code is still under development and applicable only to within-channel blurring. Therefore, we decided to compare with MATLAB deconvolution functions including "`deconvreg`," "`deconvwnr`," and "`deconvlucy`," which implement well-known algorithms that are frequently used in the practice of signal and image processing.

We now review briefly the algorithms used in these solvers. For single-channel images, the function "`deconvreg`" solves a discretization of a Tikhonov regularization problem of the form

$$\min_u \int_\Omega \left(|\Delta u|^2 + \mu|\kappa * u - f|^2\right) \mathrm{d}x, \tag{4.1}$$

where Δ is the Laplace operator, κ is a convolution kernel, $\mu > 0$ is a Lagrangian parameter determined automatically in "`deconvreg`" based on noise power of the observed image, and Ω is the domain of interest. For multichannel images, the Laplace operator is applied to each channel separately. To be robust, a certain safeguard technique is implemented in "`deconvreg`" to avoid zero denominators. However, as we have observed, some of the high frequencies get lost due to this safeguard technique when an appropriate estimation of noise power is available. In our experiment, we removed the safeguard by providing noise power of the observed image ($\texttt{std}^2 \times n^2$) to "`deconvreg`" and obtained a better result. The function "`deconvwnr`" implements the well-known Wiener filter, which determines an optimal estimation of the original image in the sense of minimizing the mean square error between the estimated image and the true image using the correlation information between the signal and noise. Finally, the function "`deconvlucy`" implements the Richardson–Lucy algorithm [29, 22], which solves a constrained maximum likelihood problem for Poissonian noise. For details about these algorithms, we refer the interested reader to [20]. Currently, these functions require identical within-channel blurring kernels over all channels, namely, $H_{11} = H_{22} = H_{33} \equiv H$, and no cross-channel blurs for color images. For this simple case, we tested $H = \mathrm{G}(21, 11)$ with $\texttt{std} = 10^{-3}$. The results on image Lena are given in Figure 2.

From Figure 2, the results of "`deconvwnr`" and "`deconvlucy`" have obvious ripples. The result of "`deconvreg`" is better because we provided noise power to the solver, based on which a suitable Lagrangian parameter in (4.1) was estimated. Furthermore, although some ringing effects are still visible in the result of `deconvreg`, it has relatively sharp edges because we removed the "safeguard," which would rather exclude certain high frequencies. The "safeguard" in `deconvreg` helps only when the noise power of the blurry and noisy images is not known. In comparison, FTVd gave the best results in terms of both SNR and visible quality because it regularized the image by TV. Generally, MATLAB functions are faster than FTVd because they solve much simpler models by only several FFTs.

Figure 2. *Comparison results with MATLAB deblurring functions.*

4.3. Cross-channel results. In this subsection, we present the experimental results recovered by FTVd from cross-channel blurred images. The cross-channel kernels were generated in exactly the same way as described in subsection 4.1 except that the 9 kernels used here were

$$\{\mathrm{M}(21,45), \mathrm{M}(41,90), \mathrm{M}(61,135), \mathrm{G}(11,9), \mathrm{G}(21,11), \mathrm{G}(31,13), \mathrm{A}(13), \mathrm{A}(15), \mathrm{A}(17)\}.$$

Observe that the above blurs in all three channels are quite severe. The noise level is still $\mathtt{std} = 10^{-3}$. To the best of our knowledge, proposed approaches in the literature for solving both MTV and "color TV" regularization problems for color image restoration are usually based on smoothing the TV term and linearizing corresponding Euler–Lagrangian equations; see, e.g., [6, 11, 42]. Furthermore, most of the discussions are restricted to the within-channel blurs to avoid costly computation; see [2, 3, 11]. Due to the lack of comparable algorithms and codes, we present results of FTVd for cross-channel blurs without comparing them to other approaches. The recovered Lena images are given in Figure 3.

As can be seen from Figure 3, FTVd with the default settings produced an image with an SNR value of 20.05dB. The CPU time used for cross-channel blurring is approximately the same as that used when the blurring is within-channel. Compared to within-channel blurs, in this case the only additional cost is to solve a block diagonalized linear system of the form (3.23), which is easily solved by Gaussian elimination with no fill-ins.

Figure 3. *Results recovered by FTVd from cross-channel blurring.*

4.4. Weighted TV restoration results. We tested our algorithm on a weighted TV/L^2 model, which enhances the reconstruction of (2.4) by better preserving sharp edges. At pixels near color discontinuities, we penalize the relevant pixels less by assigning relatively small weights to the corresponding terms in TV. Specifically, for RGB images we set $G_i = I_3 \otimes D_i$ and determined α_i in (2.5) by

$$\gamma_i = \frac{1}{1+\tau\|G_i\tilde{u}\|} \quad \text{and} \quad \alpha_i = \frac{n^2\gamma_i}{\sum_j \gamma_j}, \tag{4.2}$$

where $\tilde{u}$ is an estimate of the original image and $\tau > 0$ is determined based on experiments. As an empirical formula, (4.2) may be far from optimal but is sufficient for our purpose to illustrate that FTVd can efficiently solve the weighted model (2.5). For more discussions on how to determine local weights adaptively based on local image features, see [38, 37]. In this test, we set `std` $= 10^{-2}$ and $\mu = 10^3$. The cross-channel kernels were generated in the same manner as in subsection 4.1 except that the 9 kernels used here were

$$\{\mathrm{M}(21,45), \mathrm{M}(41,90), \mathrm{M}(61,135), \mathrm{G}(7,5), \mathrm{G}(9,5), \mathrm{G}(11,5), \mathrm{A}(13), \mathrm{A}(15), \mathrm{A}(17)\}.$$

Compared with the 9 kernels used in subsection 4.3, we used Gaussian blurs of smaller sizes because the noise level was higher. The motion and average blurs had the same sizes as those in subsection 4.3. We solved the unweighted (original MTV) model (2.4) followed by the weighted model (2.5) in which the local weight α_i at pixel i was determined by formula (4.2) using the solution $\tilde{u}$ of the unweighted model and $\tau = 15$ in this test. The results on image Rose are given in Figure 4.

From Figure 4, the image reconstructed by the weighted TV/L^2 model has both higher SNR and better visual quality than that by the unweighted model. The little drops of water on the flower are clearer in the weighted restoration result. When the "correct" weights were used, namely, the weights computed by (4.2) using the original image as $\tilde{u}$, we obtained an image (which was not depicted in Figure 4) with SNR being 16.72dB. Therefore, a better choice of weights can help improve the restoration quality. We did not give CPU time used by FTVd

Figure 4. *Numerical results of weighted TV/L^2. Original (left, 303×250). Blurry and noisy (middle left). Results of unweighted TV/L^2 (middle right). Results of weighted TV/L^2 (right).*

here and after because one can simply judge how fast FTVd is based on the time information given in the previous subsections. In subsection 4.6, we discuss the convergence speed of FTVd and how it is affected by image size, kernel size, and the Lagrangian parameter μ.

4.5. Higher-order derivative regularization. In this subsection, we present images reconstructed by solving regularization models based on higher-order derivatives. In the literature, there have been some discussions on higher-order regularization, e.g., [12, 23], which indicates the effect of preventing staircasing caused by TV. We tested with the RGB image Sunset (338×460) and set $G_i = I_3 \otimes \mathcal{D}_i$, where $\mathcal{D}_i$ is the matrix that computes the first- and the second-order forward finite differences at pixel i; namely, $\mathcal{D}_i u^{(j)} \in \mathbb{R}^6$, $j = r, g, b$, is the vector consisting of the two first-order forward finite differences (approximating u_x and u_y, respectively) and the four second-order finite differences (approximating u_{xx}, u_{xy}, u_{yx}, and u_{yy}, respectively, where $u_{xy} = u_{yx}$) of $u^{(j)}$ at pixel i. Since the weighted model gives better results than the unweighted one, we computed the weighted higher-order model, called HTV/L^2. In all, the problem we solved is

$$\min_u \sum_i \alpha_i \|(I_3 \otimes \mathcal{D}_i)u\| + \frac{\mu}{2}\|Ku - f\|^2.$$

In order to make the staircasing effect of the TV model visible, in this test we used even larger additive noise with $\texttt{std} = 0.1$. On the other hand, to maintain a good recovery quality, we chose to use smaller sizes for blurring kernels. The 9 kernels used in this experiment were

$$\{\mathrm{M}(3,0), \mathrm{M}(3,0), \mathrm{M}(3,0), \mathrm{G}(3,0.5), \mathrm{G}(3,0.5), \mathrm{G}(3,0.5), \mathrm{A}(3), \mathrm{A}(3), \mathrm{A}(3)\}.$$

The weights were generated according to (4.2) using the original image $\tilde{u}$. We set $\mu = 12.5$ and $\tau = 15$. The results on image Sunset are given in Figure 5.

Comparing the middle two images in Figure 5, the right-hand image produced by the HTV/L^2 model has cleaner sky and less staircasing in the clouds. The reduction of staircasing effects can be seen by comparing the two zoom-in images at the bottom of Figure 5.

4.6. Summary and a note. In the framework of Algorithm 1, there are two steps at each iteration. The first step applies the weighted shrinkage operation in (3.22) and has a linear

Figure 5. *Numerical results of weighted higher-order regularization. Original (upper left). Blurry and noisy (upper right). Weighted TV/L^2 result (middle left). Weighted higher-order result (middle right). Zoom-in results of weighted TV/L^2 (bottom left) and HTV/L^2 (bottom right).*

computational complexity. The second step solves a system of linear equations by calling the FFT and Gaussian elimination. The computational cost of Gaussian elimination in solving block diagonalized equations such as (3.24) is very small compared with that of FFTs since no fillings occur and no pivoting is necessary. Therefore, the per-iteration computation cost of Algorithm 1 is dominated by that of the FFTs, each costing $O(n^2 \log n)$. The remaining

question is, What is the total number of inner iterations needed for Algorithm 1 to attain a required accuracy and how does this number vary with the image size? In our experiments, this number for the current implementation of FTVd using the default parameters is almost always around 12. For each $\beta > 1$, since Algorithm 1 has a good starting point from the previous outer iteration, it only takes 1 or 2 inner iterations on average to reach the prescribed accuracy ($Res \leq 0.05$). Given that 6 FFTs are required at each inner iteration, the total number of FFTs taken by FTVd is around 80 for MTV regularized problems. If higher-order derivatives are involved in regularization, more computational work is needed on computing the right-hand side vectors in (3.23), which are mainly several higher-order finite differences of additional auxiliary vectors, but no more FFTs are needed. Furthermore, since Algorithm 1 solves u-subproblems almost exactly and does not depend on any iterative solvers, it is numerically stable and insensitive to ill-conditioning.

Now we discuss several factors that may affect the convergence speed of FTVd. First, it is obvious that the CPU time consumed by FTVd is longer for recovering larger-sized images. However, since FTVd relies on the FFT and operations with linear complexities such as finite difference, the CPU time increases moderately with the image size. Second, for fixed other factors such as the image size and μ, the speed of FTVd hardly changes with the kernel size (though recovery quality clearly depends on the kernel size); see [43] for more details. Finally, just like all other deconvolution algorithms, FTVd slows down when μ decreases. To make this clear, we solved (2.4) with different μ values. We blurred image Rose with the same cross-channel kernels as those in subsection 4.3 and added Gaussian noise with different noise levels to the blurry image. Although the total number of iterations varies with the continuation strategy and inner stopping criteria, we are able to get a general idea of how the convergence speed varies with μ by testing FTVd with the settings given in subsection 4.1. Specifically, we initialized β to be 1 and doubled it to 2^7, and for each fixed β we stopped the inner iterations by (3.28). The results on iteration numbers and SNRs of the recovered images are given in Table 1.

Table 1

Test results on convergence speed with different μ.

$(\mu, \texttt{std}^2)$	$(5 \cdot 10^4, 10^{-6})$	$(10^4, 10^{-5})$	$(10^3, 10^{-4})$	$(10^2, 10^{-3})$	$(25, 10^{-2})$
ϵ	Iter/SNR	Iter/SNR	Iter/SNR	Iter/SNR	Iter/SNR
10^{-2}	36/17.71	38/16.06	45/14.39	29/12.68	20/11.61
10^{-3}	150/17.81	198/16.19	289/14.52	360/12.90	322/11.84
10^{-4}	646/17.82	812/16.19	1849/14.53	2777/12.90	3624/11.82

As can be seen from Table 1, for each fixed μ the total iteration numbers used by FTVd increase with the increase of accuracy. However, the SNR results from $\epsilon = 10^{-4}$ have little improvement compared with those from $\epsilon = 10^{-2}$. When high accuracy is used, say $\epsilon = 10^{-4}$, more iterations are required for smaller μ because problem (2.4) generally becomes more difficult when μ gets smaller. However, in practice it is generally sufficient to let ϵ be 10^{-3} or even 10^{-2} because image quality hardly improves by requiring higher accuracy. In these two cases, more iterations seems to occur in the middle range of μ from our limited experiments; see Table 1.

Recently, a new splitting algorithm was proposed in [44], which also solves (2.4). The augmented problem formed in [44] can be written in the form of

$$\min_u \sum_i \|(I_3 \otimes D_i)v\| + \alpha\|v-u\|^2 + \frac{\mu}{2}\|Ku-f\|^2, \tag{4.3}$$

and is solved by alternating minimizations with respect to u and v. For a fixed v, the minimization with respect to u involves 6 FFTs. However, for a fixed u, the minimization problem with respect to v is a TV denoising problem that does not have a closed-form solution. In [44], the TV denoising problem is solved iteratively by an extended Chambolle's projection algorithm [7]. While the per-iteration computational complexity of our method is dominated by 6 FFTs, that of [44] is dominated by the cost of solving a TV denoising problem in addition to the 6 FFTs. According to the reported numerical results in [44], their algorithm appears to require at least as many outer iterations as ours.

5. Concluding remarks. This paper is an extension to our recent work [43] on single-channel image restoration. We derived, analyzed, implemented, and tested an alternating minimization algorithm for deblurring multichannel (color) images. The algorithm is designed to solve a general variational model where the blurs can take place both within and across channels, and the edge-preserving regularization terms can be locally weighted and use either first-order or higher-order derivatives of images, including TV regularization as a special case.

The proposed algorithm possesses strong convergence properties and, more importantly, is practically efficient as the result of exploiting problem structures to enable the use of multi-dimensional shrinkage and fast transforms in solving subproblems. Our numerical experiments confirm that, with the help of a continuation scheme, a simple MATLAB implementation of our algorithm already achieves a remarkable practical performance.

Processing speed had long been a chief obstacle preventing edge-preserving variational models from being widely used in the practice of color image processing. The construction, implementation, and experimentation of the proposed algorithm constitute, in our view, a necessary step towards making edge-preserving variational models practically viable technologies in color image restoration.

Acknowledgment. We are grateful to three anonymous referees for their many valuable comments and suggestions that have helped improve the paper.

REFERENCES

[1] S. Alliney, *Digital filters as absolute norm regularizers*, IEEE Trans. Signal Process., 40 (1992), pp. 1548–1562.

[2] L. Bar, A. Brook, N. Sochen, and N. Kiryati, *Deblurring of color images corrupted by impulsive noise*, IEEE Trans. Image Process., 16 (2007), pp. 1101–1110.

[3] P. Blomgren and T. F. Chan, *Color TV: Total variation methods for restoration of vector-valued images*, IEEE Trans. Image Process., 7 (1998), pp. 304–309.

[4] K. Boo and N. K. Bose, *Multispectral image restoration with multisensors*, IEEE Trans. Geosci. Remote Sensing, 35 (1997), pp. 1160–1170.

[5] X. Bresson and T. F. Chan, *Fast Minimization of the Vectorial Total Variation Norm and Applications to Color Image Processing*, UCLA CAM Report 07–25, University of California, Los Angeles, CA, 2007.

[6] A. Brook, R. Kimmel, and N. Sochen, *Variational restoration and edge detection for color images*, J. Math. Imaging Vision, 18 (2003), pp. 247–268.
[7] A. Chambolle, *An algorithm for total variation minimization and applications*, J. Math. Imaging Vision, 20 (2004), pp. 89–97.
[8] A. Chambolle and P. L. Lions, *Image recovery via total variation minimization and related problems*, Numer. Math., 76 (1997), pp. 167–188.
[9] T. F. Chan and S. Esedoḡlu, *Aspects of total variation regularized L^1 function approximation*, SIAM J. Appl. Math., 65 (2005), pp. 1817–1837.
[10] T. F. Chan, S. Esedoglu, F. Park, and A. Yip, *Recent developments in total variation image restoration*, in Handbook of Mathematical Models in Computer Vision, Springer, New York, 2005, pp. 17–30.
[11] T. F. Chan, S. H. Kang, and J. Shen, *Total variation denoising and enhancement color images based on the CB and HSV color models*, J. Visual Comm. Image Rep., 12 (2001), pp. 422–435.
[12] T. Chan, A. Marquina, and P. Mulet, *High-order total variation-based image restoration*, SIAM J. Sci. Comput., 22 (2000), pp. 503–516.
[13] T. Chan and J. Shen, *Variational restoration of nonflat image features: Models and algorithms*, SIAM J. Appl. Math., 61 (2000), pp. 1338–1361.
[14] D. C. Dobson and F. Santosa, *Recovery of blocky images from noisy and blurred data*, SIAM J. Appl. Math., 56 (1996), pp. 1181–1198.
[15] H. Fu, M. K. Ng, and J. L. Barlow, *Structured total least squares for color image restoration*, SIAM J. Sci. Comput., 28 (2006), pp. 1100–1119.
[16] N. P. Galatsanos, A. K. Katsaggelos, R. T. Chan, and A. D. Hillery, *Least squares restorations of multichannel images*, IEEE Trans. Signal Process., 39 (1991), pp. 2222–2236.
[17] D. Geman and G. Reynolds, *Constrained restoration and the recovery of discontinuities*, IEEE Trans. Pattern Anal. Mach. Intell., 14 (1992), pp. 367–383.
[18] D. Geman and C. Yang, *Nonlinear image recovery with half-quadratic regularization*, IEEE Trans. Image Process., 4 (1995), pp. 932–946.
[19] T. Goldstein and S. Osher, *The Split Bregman Algorithm for L1 Regularized Problems*, UCLA CAM Report 08–29, University of California, Los Angeles, CA, 2008.
[20] R. Gonzalez and R. Woods, *Digital Image Processing*, Addison–Wesley, Reading, MA, 1992.
[21] E. T. Hale, W. Yin, and Y. Zhang, *A fixed-point continuation for ℓ_1-minimization: Methodology and convergence*, SIAM J. Optim., 19 (2008), pp. 1107–1130.
[22] L. B. Lucy, *An iterative technique for the rectification of observed distributions*, Astronom. J., 79 (1974), pp. 745–754.
[23] O. M. Lysaker and X.-C. Tai, *Iterative image restoration combining total variation minimization and a second-order functional*, Int. J. Comput. Vision, 66 (2006), pp. 5–18.
[24] M. K. Ng and N. K. Bose, *Fast color image restoration with multisensors*, Int. J. Imaging Syst. Technol., 12 (2002), pp. 189–197.
[25] M. K. Ng, R. H. Chan, and W.-C. Tang, *A fast algorithm for deblurring models with Neumann boundary conditions*, SIAM J. Sci. Comput., 21 (1999), pp. 851–866.
[26] M. Nikolova, *Minimizers of cost-functions involving nonsmooth data-fidelity terms. Application to the processing of outliers*, SIAM J. Numer. Anal., 40 (2002), pp. 965–994.
[27] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, New York, 1999.
[28] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin, *An iterative regularization method for total variation-based image restoration*, Multiscale Model. Simul., 4 (2005), pp. 460–489.
[29] W. H. Richardson, *Bayesian-based iterative method of image restoration*, J. Opt. Soc. Amer., 62 (1972), pp. 55–59.
[30] R. T. Rockafellar, *Convex Analysis*, Princeton University Press, Princeton, NJ, 1970.
[31] P. Rodríguez and B. Wohlberg, *Numerical Methods for Inverse Problems and Adaptive Decomposition (NUMIPAD)*, http://numipad.sourceforge.net/.
[32] L. I. Rudin and S. Osher, *Total variation based image restoration with free local constraints*, in Proceedings of the 1st IEEE International Conference on Image Processing (ICIP), Vol. 1, 1994, pp. 31–35.
[33] L. I. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Phys. D, 60 (1992), pp. 259–268.

[34] G. Sapiro, *Color snakes*, Comput. Vis. Image Underst., 68 (1997), pp. 247–253.
[35] G. Sapiro, *Vector-valued active contours*, in Proceedings of the Conference on Computer Vision and Pattern Recognition, IEEE Computer Society, Washington, DC, 1996, pp. 680–685.
[36] G. Sapiro and D. L. Ringach, *Anisotropic diffusion of multivalued images with applications to color filtering*, IEEE Trans. Image Process., 5 (1996), pp. 1582–1586.
[37] D. Strong, P. Blomgren, and T. F. Chan, *Spatially adaptive local feature-driven total variation minimizing image restoration*, in Proceedings of SPIE, Vol. 3137, 1997, pp. 222-233.
[38] D. Strong and T. F. Chan, *Relation of Regularization Parameter and Scale in Total Variation Based Image Denoising*, UCLA CAM Report 96–7, University of California, Los Angeles, CA, 1996.
[39] B. Tang, G. Sapiro, and V. Caselles, *Color image enhancement via chromaticity diffusion*, IEEE Trans. Image Process., 10 (2001), pp. 701–707.
[40] A. Tekalp and G. Pavlovic, *Multichannel image modeling and Kalman filtering for multispectral image restoration*, Signal Process., 19 (1990), pp. 221–232.
[41] A. Tikhonov and V. Arsenin, *Solution of Ill-Posed Problems*, V. H. Winston, Washington, DC, 1977.
[42] C. R. Vogel and M. E. Oman, *Iterative methods for total variation denoising*, SIAM J. Sci. Comput., 17 (1996), pp. 227–238.
[43] Y. Wang, J. Yang, W. Yin, and Y. Zhang, *A new alternating minimization algorithm for total variation image reconstruction*, SIAM J. Imaging Sci., 1 (2008), pp. 248–272.
[44] Y. W. Wen, M. K. Ng, and Y. M. Huang, *Efficient total variation minimization methods for color image restoration*, IEEE Trans. Image Process., 17 (2008), pp. 2081–2088.
[45] B. Wohlberg and P. Rodríguez, *An iteratively reweighted norm algorithm for minimization of total variation functionals*, IEEE Signal Process. Lett., 14 (2007), pp. 948–951.
[46] W. Yin, S. Osher, D. Goldfarb, and J. Darbon, *Bregman iterative algorithms for ℓ_1-minimization with applications to compressed sensing*, SIAM J. Imaging Sci., 1 (2008), pp. 143–168.
[47] W. P. Ziemer, *Weakly Differentiable Functions: Sobolev Spaces and Functions of Bounded Variation*, Grad. Texts in Math. 120, Springer, New York, 1989.