SIAM J. SCI. COMPUT.
Vol. 32, No. 2, pp. 1043–1063

A MULTILEVEL ALGORITHM FOR SIMULTANEOUSLY DENOISING AND DEBLURRING IMAGES[*]

RAYMOND H. CHAN[†] AND KE CHEN[‡]

Abstract. In this paper, we develop a fast multilevel algorithm for simultaneously denoising and deblurring images under the total variation regularization. Although much effort has been devoted to developing fast algorithms for the numerical solution and the denoising problem was satisfactorily solved, fast algorithms for the combined denoising and deblurring model remain to be a challenge. Recently several successful studies of approximating this model and subsequently finding fast algorithms were conducted which have partially solved this problem. The aim of this paper is to generalize a fast multilevel denoising method to solving the minimization model for simultaneously denoising and deblurring. Our new idea is to overcome the complexity issue by a detailed study of the structured matrices that are associated with the blurring operator. A fast algorithm can then be obtained for directly solving the variational model. Supporting numerical experiments on gray scale images are presented.

Key words. image restoration, denoising and deblurring, total variation, regularization, multilevel methods

AMS subject classifications. 68U10, 65F10, 65K10

DOI. 10.1137/080741410

1. Introduction.

Image denoising and deblurring often coexist in image processing problems [11, 29, 57, 65]. In the continuous formulation, denote by $u = u(x, y)$ the true image and $z = z(x, y)$ the observed image, both defined in the bounded and rectangular domain $\Omega = [0, 1] \times [0, 1] \subset \mathbb{R}^2$. In practice, only $z \in \mathbb{R}^{n^2}$ is available in a discrete form [3, 38, 58]. The observed image z has been contaminated in the data collection stage, such as in acquisition and quantization. The purpose of image restoration is to recover u, as much as we can, using the degradation model

$$Ku - z = \eta, \tag{1}$$

where η is an unknown Gaussian white noise and K is a known linear degradation operator. For deblurring problems K is a convolution operator and for denoising problems $K = I$. There exist many efficient solvers for the latter problem of denoising [14, 25, 23, 33, 30, 36, 34, 37, 59] and for the former problem of (pure) deblurring with $\eta = 0$ [16, 35, 12, 17]. It is the general case where both noise and blur coexist that remains challenging to solve efficiently. The purpose of this paper is to address this general case.

As image restoration is an inverse problem that may not have a unique solution, some regularity condition has to be imposed on the solution space in order to turn the underlying ill-posed problem to a well-posed one [65]. We shall use the well-known total variation (TV) regularization to ensure that sharp features of an image are

[*]Received by the editors November 19, 2008; accepted for publication (in revised form) January 20, 2010; published electronically March 31, 2010.
http://www.siam.org/journals/sisc/32-2/74141.html

[†]Department of Mathematics, The Chinese University of Hong Kong, Shatin, Hong Kong SAR, China (rchan@math.cuhk.edu.hk). This research was supported in part by HKRGC grant 400708 and CUHK DAG grant 2060257.

[‡]Corresponding author. Centre for Mathematical Imaging Techniques and Department of Mathematical Sciences, University of Liverpool, Peach Street, Liverpool L69 7ZL, United Kingdom (k.chen@liverpool.ac.uk).

preserved [57]. However, we note that there are many other regularization functionals (some beyond the variational framework) that might be used; see [2, 48, 26, 9, 54, 49, 10] and the references therein.

Below we briefly review the common methodology to set the context for our algorithm in the following sections. Following early work [27], we choose the Tikhonov direct regularization technique to solve the inverse problem (1),

$$\min_u J(u), \qquad J(u) = \overline{\alpha} R(u) + \frac{1}{2}\|Ku - z\|_2^2, \tag{2}$$

where the regularization functional $R(u)$ is chosen as the TV seminorm [57, 27]

$$R(u) = \|u\|_{TV} = \int_\Omega |\nabla u| dxdy = \int_\Omega \sqrt{u_x^2 + u_y^2}\, dxdy. \tag{3}$$

Here the parameter $\overline{\alpha}$ represents a trade-off between the quality of the solution and the fit to the observed data. Thus the overall image restoration problem is the following:

$$\min_u \overline{\alpha}\,\|u\|_{TV} + \frac{1}{2}\,\|Ku - z\|_2^2\,. \tag{4}$$

The solution to problem (4) is given by the Euler–Lagrange equation

$$\overline{\alpha} \nabla \cdot \left(\frac{\nabla u}{|\nabla u|} \right) - K^* K u = -K^* z, \tag{5}$$

where K^* is the adjoint operator of K. Notice that the nonlinear coefficient may have a zero denominator so the equation is not defined at such points (corresponding to flat regions of the solution). A commonly adopted idea to deal with $|\nabla u| = 0$ was to introduce (yet) another parameter β to (4) and (5) so the new Euler–Lagrange equation becomes

$$\overline{\alpha} \nabla \cdot \left(\frac{\nabla u}{\sqrt{|\nabla u|^2 + \beta}} \right) - K^* K u = -K^* z. \tag{6}$$

It corresponds to minimizing, instead of (4),

$$\min_u J_\beta(u), \qquad J_\beta(u) = \int_\Omega \left[\overline{\alpha} \sqrt{u_x^2 + u_y^2 + \beta} + \frac{1}{2}(Ku - z)^2 \right] dxdy, \tag{7}$$

and in theory $u = u_\beta(x, y)$ differs from u in (5). Observe that when $\beta = 0$, (6) reduces to (5); moreover $u_\beta \to u$ as $\beta \to 0$ as shown in [1].

The existing solution methods for solving (6) or problem (4) differ in how to deal with nonlinearities, and they fall into these categories:

- *Fixed point iteration* [1, 63, 66, 67, 64, 65]: Solve a lagged diffusion problem until $u^{k+1} - u^k$ is small:

$$\overline{\alpha} \nabla \cdot \left(\frac{\nabla u^{k+1}}{\sqrt{|\nabla u^k|^2 + \beta}} \right) - K^* K u^{k+1} = -K^* z. \tag{8}$$

  There exists a large literature on this topic, mainly due to wide interest in developing fast iterative solvers for the above linear equations (once discretized). When K is a convolution operator, the challenge is to solve the resulting linear system without forming the discretized matrix of K^*K (mimicking the capability of the fast multipole method) [66, 67, 17, 18, 16, 49]. Further improvements on robustness of these solvers and on preconditioning are still needed.

- *Explicit time marching scheme* [57, 50]: The original idea in [57] was refined in [50] as solving the following parabolic PDE until a steady state has been reached:

$$u_t = |\nabla u| \left[\overline{\alpha} \nabla \cdot \left(\frac{\nabla u}{\sqrt{|\nabla u|^2 + \beta}} \right) - K^* K u + K^* z \right]. \tag{9}$$

  As remarked in [55], for linear problems, this type of idea represents a kind of relaxation scheme. The drawback is that the artificial time step Δt must be small due to stability requirement.
- *Primal-dual method* [27, 28, 7]: As discussed in [7], the Newton method for (6) leads to very slow or no convergence because z is often not a sufficiently close initial guess for u. Introducing the dual variable (vector) $\omega = \nabla u / \sqrt{|\nabla u|^2 + \beta}$ appears to have made the combined system

$$\begin{cases} \overline{\alpha} \nabla \cdot \omega - K^* K u = -K^* z, \\ \omega \sqrt{|\nabla u|^2 + \beta} - \nabla u = 0, \end{cases}$$

  in two variables (u, ω) more amenable to Newton iterations as the new system is nearly "linear" in the two variables (after elimination of w the equation in u is less linear). Note that ω is constrained in each iteration step so the overall algorithm needs some careful implementation. However, up to now, an efficient multilevel implementation of this method remains to be developed. The same is true for an alternative primal-dual formulation [43].
- *Alternating minimization method* [68, 44]: As it appears difficult to propose more efficient methods for solving (6), one recent idea was to separate denoising from deblurring in an iterative way. To this end, the original functional (2) is replaced (approximated) by

$$\min_{u,f} J^A(u, f), \qquad J^A(u, f) = \overline{\alpha} R(u) + \overline{\beta} \|u - f\|_2^2 + \frac{1}{2} \|Kf - z\|_2^2, \tag{10}$$

  which is no easier to solve if coupling is implemented. Instead, the idea is to solve alternatively the following (given some initial $\widetilde{f}$):

$$\begin{cases} \widetilde{u} = \text{argmin}_u J^A\left(u, \widetilde{f}\right) = \overline{\alpha} R(u) + \overline{\beta} \|u - \widetilde{f}\|_2^2, \\ \widetilde{f} = \text{argmin}_f J^A\left(\widetilde{u}, f\right) = \overline{\beta} \|f - \widetilde{u}\|_2^2 + \frac{1}{2} \|Kf - z\|_2^2, \end{cases} \tag{11}$$

  which is indeed calling two separate (efficient and existing) solvers, respectively, for denoising and deblurring.

In addition, for the denoising case, there exist the powerful dual formulations [14, 42] that replace the primal variable u by its dual variable $\mathbf{p} = (p_1, p_2)$. The work of [15, 46, 56] modifies the TV formulation in other ways so that the new equations become more amenable to numerical implementation. For the dual formulation, efficient algorithms can be developed [25].

As far as fast nonlinear multilevel methods for (4) and (6) are concerned, two approaches tested for denoising are possible candidates to be generalized to the deblurring case:

- Full approximation scheme (FAS) based methods. The original FAS method was proposed by Brandt in the 1970s and was generalized to optimization problems by Ta'asan [60]. Further studies of the approach can be found in [51, 8, 39].
- Non-FAS based methods. While the FAS methods design coarse level optimizations using the residual information of the first order conditions of a fine level, non-FAS methods try to derive coarse level optimizations by using coarse subspaces in coordinate descent fashion. The earlier references include [6, 62, 61, 47, 53]. Application to the denoising problem was considered in [13, 23].

In what follows we shall follow the latter approach and attempt to solve the original primal and optimization formulation (4) directly by a (non-FAS based) multilevel algorithm and thus generalize our previous denoising algorithm [22] with $K = I$ to the combined case of denoising and deblurring. The former approach for this combined case is not yet considered in the literature, but various potential difficulties were discussed in [16].

The plan is to review our recently proposed multilevel method for the Gaussian noise removal [22] in section 2. The section also shows that its direct generalization to the combined case of denoising and deblurring suffers from loss of multilevel efficiency. Then section 3 presents a new multilevel method that utilizes the fast multipole ideas to overcome the complexity issue and hence recover the multilevel efficiency. The step of implementing the patch level minimization is particularly challenging to work out. Numerical experiments are reported in section 4, where we shall compare the new method with the primal-dual method [27, 28, 7].

2. Review of a multilevel method for optimization.

We now briefly review the multilevel optimization method proposed in [22] for removing Gaussian noise, applied in [19] to Poisson noise removal and in [20] to impulse noise removal. One nice advantage about the method is that it can be applied to nonsmooth functionals as it does not require their first order conditions (or global gradient).

For a minimization problem, the important issue in designing a multilevel method is how to make use of an approximate solution $\widetilde{u}$ to improve it further, or how to measure the "distance" from the true minimizer so that this information can be passed onto the coarse levels somehow. (We note that for differentiable functions, first order conditions can be used to define a residual of $\widetilde{u}$, and then it is passed onto the modification of a coarse level minimization functional; see [60, 8, 51].) That is to say, here, there is no obvious way to define a residual correction functional (as done for an operator equation [32]). Here we shall use the coordinate descent method [6, 41, 53, 62, 45, 13, 24] as a smoothing method to find a solution near the true minimizer (not to find the minimizer itself).

In this paper, to simplify the notation, we assume that the image z is given in a vector form $z \in \mathbb{R}^{n^2}$ with its entries $z_{i,j}$ ordered lexicographically. Similarly we shall use u to denote the reconstructed image in a vector from ordering its entries $u_{i,j}$ lexicographically. Also we shall use $\widetilde{z}, \widetilde{u}$ to mean, respectively, the known vectors of u, z at a point of discussion. However, for continuous formulations, we reuse z, u to mean, respectively, the given image function and the reconstruction function.

Assume $n = 2^L$ and let the standard coarsening be used, giving rise to $L+1$ levels $k = 1$ (finest)$, 2, \ldots, L, L+1$ (coarsest). Denote the dimension of level k by $\tau_k \times \tau_k$ with $\tau_k = n/2^{k-1}$.

We illustrate the method in solving the standard TV *denoising* model [57]:

$$\min_u J(u), \qquad J(u) = \int_\Omega \left(\overline{\alpha}\sqrt{u_x^2 + u_y^2} + \frac{1}{2}(u-z)^2 \right),$$

which is discretized to give rise to the optimization problem,

$$\min_{u \in \mathbb{R}^{n^2}} J(u), \tag{12}$$

$$J(u) = \alpha \sum_{i=1}^{n-1}\sum_{j=1}^{n-1} \sqrt{(u_{i,j}-u_{i,j+1})^2 + (u_{i,j}-u_{i+1,j})^2} + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(u_{i,j}-z_{i,j})^2,$$

with $\alpha = \overline{\alpha}/h$ and $h = 1/(n-1)$.

As a prelude to multilevel methods, consider the minimization of (12) by the coordinate descent method on the finest level 1:

$$\begin{cases} \text{Given} & u^{(0)} = (u_{i,j}^{(0)}) = (z_{i,j}) \text{ with } l = 0, \\ \text{Solve} & u_{i,j}^{(l)} = \operatorname{argmin}_{u_{i,j}\in\mathbb{R}} J^{\text{loc}}(u_{i,j}) \text{ for } i,j = 1,2,\ldots,n, \\ \text{Set} & u^{(l+1)} = (u_{i,j}^{(l)}) \text{ and repeat the above step with } l = l+1 \\ & \quad \text{until a prescribed stopping step on } l, \end{cases} \tag{13}$$

where

$$\begin{aligned} J^{\text{loc}}(u_{i,j}) = \alpha \Bigg[& \sqrt{\left(u_{i,j} - u_{i+1,j}^{(l)}\right)^2 + \left(u_{i,j} - u_{i,j+1}^{(l)}\right)^2} \\ & + \sqrt{\left(u_{i,j} - u_{i-1,j}^{(l)}\right)^2 + \left(u_{i-1,j}^{(l)} - u_{i-1,j+1}^{(l)}\right)^2} \\ & + \sqrt{\left(u_{i,j} - u_{i,j-1}^{(l)}\right)^2 + \left(u_{i,j-1}^{(l)} - u_{i+1,j-1}^{(l)}\right)^2} \ \Bigg] + \tfrac{1}{2}(u_{i,j} - z_{i,j})^2. \end{aligned} \tag{14}$$

Note that each subproblem in (13) is only one dimensional.

To introduce the multilevel algorithm, we rewrite (13) as

$$\begin{cases} \text{Given} & u^{(0)} = (u_{i,j}^{(0)}) = (z_{i,j}) \text{ with } l = 0, \\ \text{Solve} & \hat{c} = \operatorname{argmin}_{c\in\mathbb{R}} J^{\text{loc}}(u_{i,j}^{(l)} + c), \ \text{ set } u_{i,j}^{(l)} = u_{i,j}^{(l)} + \hat{c}, \\ \text{Set} & u^{(l+1)} = (u_{i,j}^{(l)}) \text{ and repeat the above step with} \\ & \quad l = l+1 \ \text{ until a prescribed stopping step on } l, \end{cases} \tag{15}$$

where $i, j = 1, 2, \ldots, n$. Here each subproblem can be interpreted as finding the best correction constant $\hat{c}$ at the current approximate $u_{i,j}^{(l)}$ on level 1. Likewise one may consider a 2×2 block of pixels with pixel values denoted by the current approximate $\widetilde{u}$. Our multilevel method for $k = 2$ is to look for the best correction constant to update this block so that the underlying merit functional (relating to all four pixels) achieves a local minimum. One sees that this idea operates on level 2. If we repeat the idea with larger blocks, we arrive at levels 3 and 4 with respective 4×4 and 8×8 blocks.

If we write down the above idea in formulae, it may appear complicated but the idea is simple. On level k, set $b = 2^{k-1}$, $k_1 = (i-1)b+1$, $k_2 = ib$, $\ell_1 = (j-1)b+1$, $\ell_2 = jb$. Then the (i,j)th computational block (stencil) involving the single constant $c_{i,j}$ on level k can be depicted in terms of pixels of level 1 as follows:

$$\begin{array}{c|ccc|c}
\vdots & \vdots & \cdots & \vdots & \vdots \\
\widetilde{u}_{k_1-1,\ell_2+1}+c_{i-1,j+1} & \widetilde{u}_{k_1,\ell_2+1}+c_{i,j+1} & \cdots & \widetilde{u}_{k_2,\ell_2+1}+c_{i,j+1} & \widetilde{u}_{k_2+1,\ell_2+1}+c_{i+1,j+1} \\ \hline
\widetilde{u}_{k_1-1,\ell_2}+c_{i-1,j} & \widetilde{u}_{k_1,\ell_2}+c_{i,j} & \cdots & \widetilde{u}_{k_2,\ell_2}+c_{i,j} & \widetilde{u}_{k_2+1,\ell_2}+c_{i+1,j} \\
\cdots & \vdots & \cdots & \vdots & \cdots \\
\widetilde{u}_{k_1-1,\ell_1}+c_{i-1,j} & \widetilde{u}_{k_1,\ell_1}+c_{i,j} & \cdots & \widetilde{u}_{k_2,\ell_1}+c_{i,j} & \widetilde{u}_{k_2+1,\ell_1}+c_{i+1,j} \\ \hline
\widetilde{u}_{k_1-1,\ell_1-1}+c_{i-1,j-1} & \widetilde{u}_{k_1,\ell_1-1}+c_{i,j-1} & \cdots & \widetilde{u}_{k_2,\ell_1-1}+c_{i,j-1} & \widetilde{u}_{k_2+1,\ell_1-1}+c_{i+1,j-1} \\
\vdots & \vdots & \cdots & \vdots & \vdots
\end{array}. \tag{16}$$

Clearly there is only one unknown constant $c_{i,j}$ in each block, and we shall obtain a one-dimensional subproblem. After some algebraic manipulation [22, 23], we find that the local minimization problem $\min_{c_{i,j}} J(\widetilde{u} + P_k c_{i,j})$ (with P_k an interpolation operator distributing $c_{i,j}$ to a block on level k (which has $b \times b$ pixels on level 1) as illustrated above) is equivalent to a one-dimensional problem.

For later use, consider the general case of a piecewise constant $c_{i,j}$ update on a certain block of $b_1 \times b_2$ pixels, occupying the index range of $[k_1, k_2] \times [\ell_1, \ell_2]$ on level 1. The one-dimensional (1D) problem is $\min_{c_{i,j}} G(c_{i,j})$, where

$$G(c_{i,j}) = \alpha T(c_{i,j}) + \frac{b_1 b_2}{2}(c_{i,j} - \widetilde{q}_{i,j})^2, \tag{17}$$

$$\begin{aligned}
T(c_{i,j}) = {} & \sum_{\ell=\ell_1}^{\ell_2} \sqrt{(c_{i,j} - h_{k_1-1,\ell})^2 + v_{k_1-1,\ell}^2} + \sum_{m=k_1}^{k_2-1} \sqrt{(c_{i,j} - v_{m,\ell_2})^2 + h_{m,\ell_2}^2} \\
& + \sum_{\ell=\ell_1}^{\ell_2-1} \sqrt{(c_{i,j} - h_{k_2,\ell})^2 + v_{k_2,\ell}^2} + \sum_{m=k_1}^{k_2} \sqrt{(c_{i,j} - v_{m,\ell_1-1})^2 + v_{m,\ell_1-1}^2} \\
& + \sqrt{2}\sqrt{(c_{i,j} - \overline{v}_{k_2,\ell_2})^2 + \overline{h}_{k_2,\ell_2}^2},
\end{aligned}$$

$\widetilde{z}_{m,\ell} = z_{m,\ell} - \widetilde{u}_{m,\ell}$, and

$$\left\{\begin{array}{ll}
\widetilde{q}_{i,j} = \operatorname{mean}\Big(\widetilde{z}(k_1 : k_2, \ell_1 : \ell_2)\Big) = \displaystyle\sum_{m=k_1}^{k_2} \sum_{\ell=\ell_1}^{\ell_2} \frac{\widetilde{z}(m,\ell)}{b_1 b_2}, & \\
\widetilde{v}_{m,\ell} = \widetilde{u}_{m,\ell+1} - \widetilde{u}_{k,\ell}, & \overline{v}_{k_2,\ell_2} = \dfrac{v_{k_2,\ell_2} + h_{k_2,\ell_2}}{2}, \\
\widetilde{h}_{m,\ell} = \widetilde{u}_{m+1,\ell} - \widetilde{u}_{m,\ell}, & \overline{h}_{k_2,\ell_2} = \dfrac{v_{k_2,\ell_2} - h_{k_2,\ell_2}}{2}.
\end{array}\right. \tag{18}$$

The solution of the above 1D minimization problem, solved by a Richardson iteration [23], defines the updated solution of $u = \widetilde{u} + P_k c_{i,j}$. Then we obtain a multilevel method if we cycle through all levels and all blocks on each level.

We now briefly comment on the optimization multilevel method. First, the convergence is generally not guaranteed, if the functional J is nonsmooth. However, in [22], we found that the wrongly converged solution is only incorrect near flat patches of the solution. Such patches are related to the hemi-variateness of the unknown solution [53]. The idea of detecting such flat patches during iteration and incorporating new local minimizations based on the patches was suggested in [22]. Essentially we implement a new coarse level. Second, how would one solve the one-dimensional minimization problem? Our experience suggests either a fixed point based Richardson iteration or the Newton method [5]. On the coarsest level, the TV term is unchanged by adding c, so the problem has an exact solution. To avoid the gradient becoming zero on other levels, we need a regularizing parameter δ [22] which does not influence the final convergence as long as it is small (e.g., 10^{-20}). Here the solution of each local minimization problem is only needed to be approximate as with smoothing steps of a multigrid method for an operator equation. One might question the advantage of solving (12) this way and suggest to solve a regularized version of (12) by adding the parameter δ at the very beginning. While this is feasible, the newly regularized problem is theoretically smooth and the resulting multilevel method will converge. However, the convergence will be slow if δ is small, and in general the method is sensitive to changes in δ. Moreover, our patch detection idea will not work as local patches are suppressed. Hence we do not propose to regularize (12).

Overall the revised multilevel method [22] for solving (12)—the denoising problem—is the following algorithm.

```
ALGORITHM 1. Given $z$ and an initial guess $\widetilde{u} = z$, with $L+1$ levels,
1. Iteration starts with $u_{old} = \widetilde{u}$ ($\widetilde{u}$ contains the initial guess before the first iteration
   and the updated solution at all later iterations).
2. Smooth for $\nu$ iterations the approximation on the finest level 1, i.e., solve (13) for
   $i, j = 1, 2, \dots, n$.
3. Iterate for $\nu$ times on each coarse level, i.e.,
   for $k = 2, 3, \dots, L+1$:
      • compute $\widetilde{z} = z - \widetilde{u}$, $\widetilde{q}_{i,j}$, $\widetilde{v}_{m,\ell}$, and $\widetilde{h}_{m,\ell}$ via (18),
      • compute the minimizer $c$ of (17) for each block on level $k$ if $k \le L$; on the
        coarsest level $k = L+1$, the correction constant (of the single block) is simply
        $c = mean(\widetilde{q}) = mean(z - \widetilde{u})$,
      • add the correction, $\widetilde{u} = \widetilde{u} + P_k c$, where $P_k$ is the interpolation operator
        distributing $c_{i,j}$ to the corresponding $b \times b$ block on level $k$ as illustrated
        in (16).
   end
4. On level $k = 1$, find the maximum possible patch at each position $(i, j)$ for some
   small $\varepsilon$,

      $$patch = \{(i_\ell, j_\ell) \ : \ |u_{i_\ell, j_\ell} - u_{i,j}| < \varepsilon\},$$

   which is assumed to be a generic block of $b_1 \times b_2$ pixels with index $[k_1, k_2] \times [\ell_1, \ell_2]$.
   for each of such blocks,
      • compute the minimizer $c$ of (17) for the block;
      • add the constant correction $c$ to $\widetilde{u}$ as with step 3.
   end
5. If $\|\widetilde{u} - u_{old}\|_2$ is small enough, exit with $u = \widetilde{u}$ or return to step 1 and continue
   iterations.
```

We note that whenever the TV seminorm is used (resulting in a nonsmooth J), the solution will allow local constants. Such local constants lead to the hemivariateness of the solution, which may prevent local minimizations reaching the global minimizer [53]. Step 4 here is to overcome this; see [23]. Finally, we remark that the above method can also be adapted for solving the Poisson denoising model [19].

Here Algorithm 1 has a complexity of $O(n^2 \log n)$ because each level costs $O(n^2)$ and there are up to $O(\log n)$ levels. Our aim is to achieve the same complexity of $O(n^2 \log n)$ for the new problem (4)—the problem with deblurring.

Complexity issues associated with a generalized multilevel method. We now consider how to generalize the above multilevel algorithm to the combined denoising and deblurring case. In fact, it suffices to show that even implementing the finest level minimization alone cannot be made efficient because the complexity is already about $O(n^4)$, while the expected is only $O(n^2 \log n)$. We remark that if the TV seminorm is replaced by the L_2 norm, efficient multigrid methods can be developed easily; see [16] and the references therein. The TV seminorm for deblurring remained a hard problem to solve efficiently up to now. The most difficult aspect is a lack of suitable smoothers as all local smoothers turn out to be global in terms of complexity.

Below we show the issues to be addressed when Algorithm 1 is directly applied to (4). Recall that the standard TV model [57] in the continuous form is

$$\min_u J(u), \qquad J(u) = \int_\Omega \left(\overline{\alpha}\sqrt{u_x^2 + u_y^2} + \frac{1}{2}(Ku - z)^2 \right) dxdy.$$

The above TV problem can be discretized to give rise to the optimization problem,

$$\min_{u \in \mathbb{R}^{n^2}} J(u), \tag{19}$$

$$J(u) = \alpha \sum_{i=1}^{n-1}\sum_{j=1}^{n-1} \sqrt{(u_{i,j} - u_{i,j+1})^2 + (u_{i,j} - u_{i+1,j})^2} + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\left(\sum_{\ell=1}^{n}\sum_{m=1}^{n} K_{i,j;\ell,m} u_{\ell,m} - z_{i,j}\right)^2,$$

with $\alpha = \overline{\alpha}/h$ and $h = 1/(n-1)$. Here the structure of the dense matrix $K = (K_{i,j;\ell,m})$ of size $n^2 \times n^2$ depends on the assumption of the blurring operator in (1) and the boundary condition for u; see [65, 29].

Below we shall assume that K is a block circulant matrix with circulant blocks (BCCB). This is the case if we adopt the periodic boundary condition [52].

Now solve (19) by the coordinate descent method on the finest level 1:

$$\begin{cases} \text{Given} \quad u^{(0)} = (u_{i,j}^{(0)}) = (z_{i,j}) \text{ with } l = 0, \\ \text{Solve} \quad u_{i,j}^{(l)} = \text{argmin}_{u_{i,j}\in\mathbb{R}} J^{\text{loc}}(u_{i,j}) \text{ for } i,j = 1,2,\ldots,n, \\ \text{Set} \quad u^{(l+1)} = (u_{i,j}^{(l)}) \text{ and repeat the above step with } l = l+1 \\ \qquad \text{until a prescribed stopping step on } l, \end{cases} \tag{20}$$

where

$$J^{\text{loc}}(u_{i,j}) = \frac{1}{2}\|Ku - z\|^2 + \alpha \left[\sqrt{\left(u_{i,j} - u^{(l)}_{i+1,j}\right)^2 + \left(u_{i,j} - u^{(l)}_{i,j+1}\right)^2} \right.$$
$$+ \sqrt{\left(u_{i,j} - u^{(l)}_{i-1,j}\right)^2 + \left(u^{(l)}_{i-1,j} - u^{(l)}_{i-1,j+1}\right)^2}$$
$$\left. + \sqrt{\left(u_{i,j} - u^{(l)}_{i,j-1}\right)^2 + \left(u^{(l)}_{i,j-1} - u^{(l)}_{i+1,j-1}\right)^2} \right]. \tag{21}$$

Although each subproblem in (20) is only one dimensional, we see that it has an $O(n^2)$ complexity because the fitting term involves vectors of length n^2 and, in particular, $Ku = u_{i,j}w_t + \widetilde{w}_t$, where $t = (j-1)n + i$, $w_t \in \mathbb{R}^{n^2}$ is the tth column of K, and $\widetilde{w}_t$ is a vector not involving $u_{i,j}$ (i.e., a weighted sum of all columns of K except t).

The same complexity problem persists on level k, where $\min_{c_{i,j}} J(\widetilde{u}+P_k c_{i,j})$ leads to minimization of the local subproblem

$$J^{\text{loc}}(c_{i,j}) = \alpha T(c_{i,j}) + \frac{1}{2}\|c_{i,j}w_t + K\widetilde{u} - z\|^2 = \alpha T(c_{i,j}) + \frac{1}{2}\|c_{i,j}w_t - \widetilde{z}\|^2, \tag{22}$$

where $\widetilde{z} = z - K\widetilde{u}$ is known, $T(c_{i,j})$ is as defined in (17), and the vector $w_t \in \mathbb{R}^{n^2}$ denotes the summation of all columns of K corresponding to the entries inside the (i,j) block on level k.

The complexity issue prevents us from achieving the expected multilevel efficiency of $O(n^2 \log n)$, because the accumulated complexity of solving all subproblems amounts to $O(n^4 \log n)$. This is one of the main reasons why no one has implemented such a method. Below we shall address the question of how to solve subproblem (22) without explicitly forming or storing the long vectors w_t. It turns out that this is possible, and consequently a working multilevel method is obtained.

3. An efficient multilevel method.

As we know, if each subproblem must be solved in isolation, the overall algorithm does not have the expected multilevel efficiency. However, we have found that all the time-consuming parts in solving these subproblems lie in vector products related to the blurring operator and such products can be arranged in a nested way, in the spirit of a fast multipole method [40]. The resulting algorithm can achieve the same efficiency as for the denoising case.

The starting point of our idea lies in considering how to minimize the subproblem (22), not individually but somehow collectively in terms of precomputation steps. Note that its first order condition takes the form

$$\alpha T'(c_{i,j}) + w_t^T w_t c_{i,j} = w_t^T \widetilde{z}. \tag{23}$$

We anticipate that once all such quantities $w_t^T w_t$ and $w_t^T \widetilde{z}$ (for all w_t recursively as one deals with partial sums in the fast multipole method [40]) are computed and stored first before each multilevel cycle, the local solvers will not be expensive to proceed. Below we investigate such quantities.

3.1. Fast construction of partial columns of the blurring matrix K.

We first address the problem of how to compute $w_t^T w_t$ efficiently, when w_t denotes the sum of columns of K corresponding to the block on level k. It turns out that these vector quantities w_t are structured for any level. Typically the number of such columns to be summed on level k is 4^{k-1}, with each column of size n^2. This is because the number of entries in any block on level k is 4^{k-1}.

To explain the notation and to illustrate the idea of w_t being structured, consider $n = 4$, i.e., a small image with 4×4 pixels. Denote matrix K and its w_1 on level 2, respectively, as follows:

$$\left[\begin{array}{cc|cc|cc|cc||cc|cc|cc|cc}
a & b & c & d & e & f & g & h & i & j & k & l & m & n & o & p \\
h & a & b & c & d & e & f & g & p & i & j & k & l & m & n & o \\ \hline
g & h & a & b & c & d & e & f & o & p & i & j & k & l & m & n \\
f & g & h & a & b & c & d & e & n & o & p & i & j & k & l & m \\ \hline
e & f & g & h & a & b & c & d & m & n & o & p & i & j & k & l \\
d & e & f & g & h & a & b & c & l & m & n & o & p & i & j & k \\ \hline
c & d & e & f & g & h & a & b & k & l & m & n & o & p & i & j \\
b & c & d & e & f & g & h & a & j & k & l & m & n & o & p & i \\ \hline\hline
i & j & k & l & m & n & o & p & a & b & c & d & e & f & g & h \\
p & i & j & k & l & m & n & o & h & a & b & c & d & e & f & g \\ \hline
o & p & i & j & k & l & m & n & g & h & a & b & c & d & e & f \\
n & o & p & i & j & k & l & m & f & g & h & a & b & c & d & e \\ \hline
m & n & o & p & i & j & k & l & e & f & g & h & a & b & c & d \\
l & m & n & o & p & i & j & k & d & e & f & g & h & a & b & c \\ \hline
k & l & m & n & o & p & i & j & c & d & e & f & g & h & a & b \\
j & k & l & m & n & o & p & i & b & c & d & e & f & g & h & a
\end{array}\right], \quad
\begin{bmatrix}
a+b+e+f \\ h+a+d+e \\ g+h+c+d \\ f+g+b+c \\ e+f+a+b \\ d+e+h+a \\ c+d+g+h \\ b+c+f+g \\ i+j+m+n \\ p+i+l+m \\ o+p+k+l \\ n+o+j+k \\ m+n+i+j \\ l+m+p+i \\ k+l+o+p \\ j+k+n+o
\end{bmatrix}. \tag{24}$$

First observe that K is block circulant in blocks of sizes 1×1, 2×2, 4×4, and 8×8. Since a circulant matrix can be represented by its root matrix which consists of elements from its column 1 (see [65, 32, 21]), the root matrix on level 1 and on level 2 for the above K may be, respectively, represented by

$$T_1 = \left[\begin{array}{c|c|c|c} a & e & i & m \\ h & d & p & l \\ g & c & o & k \\ f & b & n & j \end{array}\right], \qquad T_2 = \text{reshape}(w_1, 4, 4),$$

where "reshape" denotes the operation of reshaping the vector w_1 into a 4×4 (root) matrix. Second, we claim that the new matrix made from the partial sums of columns of K has also a block circulant structure. Consider the first subproblem (from block $(1,1)$) on level 2. The local update required is

$$u = \widetilde{u} + c_{1,1}[1,\ 1,\ 0,\ 0,\ \ 1,\ 1,\ 0,\ 0,\ \ 0,\ 0,\ 0,\ 0,\ \ 0,\ 0,\ 0,\ 0]^T \equiv \widetilde{u} + c_{1,1}e_{1,1},$$

and from (22) the local functional is

$$J^{\text{loc}}(c_{1,1}) = \alpha T(c_{1,1}) + \frac{1}{2}\|c_{1,1}w_t + K\widetilde{u} - z\|^2 = \alpha T(c_{1,1}) + \frac{1}{2}\|c_{1,1}w_t - \widetilde{z}\|^2,$$

with $w_t = Ke_{1,1}$ shown in (24) as a sum of the four columns of K (in particular, columns $1, 2, 5, 6$) and $t = (j-1)n + 1 = 1$. Hence the new matrix $[w_1, w_2, w_3, w_4]$ of all partial sums of columns of K on level 2 is determined completely by its first column.

The above illustration suggests that the matrix of all w_t has a circulant structure, so storing this matrix amounts to keeping a copy of its root matrix (which is formed by reshaping its first column).

Thus on each level, we only need to form the column 1 of this matrix of partial sums. Also we may form such matrices recursively starting from the finest level.

ALGORITHM 2 (computation of partial sums of K). *Let $T_1 \in \mathbb{R}^{n\times n}$ be the root matrix of the BCCB matrix K on level* 1 *(which is reshaped from column* 1 *of K). Below for $k \geq 2$, we shall form the root matrix $T_k \in \mathbb{R}^{n\times n}$ that contains partial sums of K's 4^{k-1} columns and $W_k = \|T_k\|_F^2$.*

```
for $k = 2, \ldots, L+1$
    • Set $b = 2^{k-2}$ (the width of blocks on level $k-1$).
    • Assign $Y = T_{k-1}$.
        1. Inter-block summation
            – Set the first block $X_1 = Y(:, 1:b) + Y(:, n-b+1:n)$
            – for $\ell = b+1, 2b+1, 3b+1, \ldots, (n-b+1)$
                  Set block $\ell$:      $X_\ell = Y(:, \ell:\ell+b-1) + Y(:, \ell-b:\ell-1)$
              end
            – Set the matrix $X = [X_1,\ X_{b+1},\ X_{2b+1},\ X_{3b+1}, \ldots,\ X_{(n-b+1)}]$
        2. Inner-block summation
            – Set the first block $Y_1 = X(1:b, :) + X(n-b+1:n, :)$
            – for $\ell = b+1, 2b+1, 3b+1, \ldots, (n-b+1)$
                  Set block $\ell$:      $Y_\ell = X(\ell:\ell+b-1, :) + X(\ell-b:\ell-1, :)$
              end
            – Set the matrix $Y = [Y_1;\ Y_{b+1};\ Y_{2b+1};\ Y_{3b+1}; \ldots,\ Y_{(n-b+1)}]^T$
        3. Compute $W_k = \|Y\|_F^2$ which is the same as $w_t^T w_t$ on level $k$.
    • Store $T_k = Y$.
end
```

3.2. Fast computation of the product $K^T\widetilde{z}$ on all levels. We next consider how to compute $\widetilde{z} = z - K\widetilde{u}$ and $w_t^T\widetilde{z}$. Since K as a BCCB matrix is diagonalizable by the fast Fourier transform (FFT) [32, 52, 65], computing $\widetilde{z}$ requires only $O(n^2 \log n)$ operations on the finest level. So the remaining task is to compute such products on other coarse levels efficiently.

To compute $w_t^T\widetilde{z}$ on level k with the latest solution $\widetilde{u}$, we note that each w_t vector can be recovered by the stored root matrix T_k on level k from Algorithm 2. But it is not advisable to use T_k directly for multiplication. A feasible method seems to be the following, where we first generate the vector $K^T\widetilde{z}$ on the finest level 1 and then add appropriate rows to produce the products $w_t^T\widetilde{z}$ on level k which is a vector of size $\tau_k = n^2/4^{k-1} = (n/2^{k-1})^2$.

ALGORITHM 3 (computation of products $w_t^T\widetilde{z}$). *Let fft2(K, v) denote the process of computing the matrix-vector product K^Tv via the FFT. Suppose that $\widetilde{z}$ has been updated before starting level k. Now on level k, we shall compute the vector $V = w_t^T\widetilde{z}$ by the following steps:*

```
1. Complete $W = fft2(K, \widetilde{z})$; here $W \in \mathbb{R}^{n^2}$.
   If $k = 1$, exit the algorithm with $V = W$, otherwise continue;
2. for $\ell = 0, \ldots, k-2$
       • Set the counter $j = 0$, the block size $b = 2^\ell$ and the number of
             blocks $m = n/b$ along one way.

       • for column $c = 1, 3, \ldots, m-1$
             - set $s = (c-1)m$;
             - for row $r = 1, 3, \ldots, m-1$
                   j=j+1;
```

```
            compute the partial sum $p_j = \sum_{i=0}^{1}(W_{s+r+i} + W_{s+r+m+i})$.
        end
    end
  • Set $W = [p_1, p_2, \ldots, p_j]^T$ and the new size $m = m/2$;
  end
3. End the algorithm with $V = W$.
```

The algorithm costs $O(kn^2) = O(n^2 \log n)$ per multilevel iteration because $k \leq \log n$.

Thus with Algorithms 2 and 3, we can ensure that the local solvers (as smoothers of a multilevel method) do not accumulate more than $O(n^2 \log n)$ work.

3.3. Efficient implementation on the patch (coarse) level. We have thus far covered all steps of Algorithm 1 for a deblurring model except its step 4 which involves patches. Fortunately in many experiments, we have found that such a step is not needed because no large patches have been observed. However, for a complete algorithm with full generality, it remains to address the more technical problem of how to implement such a patch step, where a patch of pixels is not aligned with any coarse grids.

Our idea is as follows. Since all partial sums of K on coarse levels are stored (via their root matrices), all we need to consider is to connect a patch to the concerned partial sums across coarse levels—the more coarser levels are involved, the more efficient it becomes to form the vector w_t of the sum of all those columns of K corresponding to the detected patch. Below we shall show first how to distribute the patch onto coarse levels, and then how to identify the coarse entries for adding toward w_t.

Suppose that step 4 of Algorithm 1 has detected a patch with indices

$$\mathcal{S} = \{(\ell, m) \mid i_1 \leq \ell \leq i_2,\ j_1 \leq m \leq j_2\} \tag{25}$$

that in turn requires the solution of a corresponding problem $\min_{c_{i,j}} J(\widetilde{u} + P_k c_{i,j})$ similar to the subproblem (22). This task of working out w_t appears quite difficult if we are to maintain the expected efficiency of $O(n^2 \log n)$. First, direct computation of the vector w_t (the sum of columns of K corresponding to the pixels in the patch) is not feasible, because theoretically the number of pixels in a patch $\mathcal{S}$ may be up to $O(n^2)$. Second, to motivate the ideas, suppose $\mathcal{S}$ can be grouped into blocks of pixels aligned with level k_1 plus the remaining ones aligned with level k_2. Then the vector w_t that we are to compute is simply a sum of those (block) columns from levels k_1 and k_2. Therefore we only have to think of a way to divide $\mathcal{S}$ into a set of largest possible blocks. It turns out that this is possible to implement, as illustrated, respectively, in Figures 1 ($n = 8$) and 2 ($n = 16$) for two cases of small images $n \times n$, where the blocks (from different levels) are shown as boxed. Clearly finding the large blocks mimics the idea of assigning far field interaction boxes in a fast multipole method [40, 32].

Two algorithms are given below. The first one is to work out the blocks as shown in Figures 1 and 2, while the second is to further work out the vector w_t and then $w_t^T w_t$ and $w_t^T \widetilde{z}$ that are needed to solve a local patch level minimization. Here the left plots from Figures 1 and 2 show how an interval in each coordinate direction is covered by the largest possible mesh lines from coarse levels (Stage 1 of Algorithm 4) and the right plots show how to convert the rectangular coverings into regular boxes on coarse levels (Stage 2 of Algorithm 4).

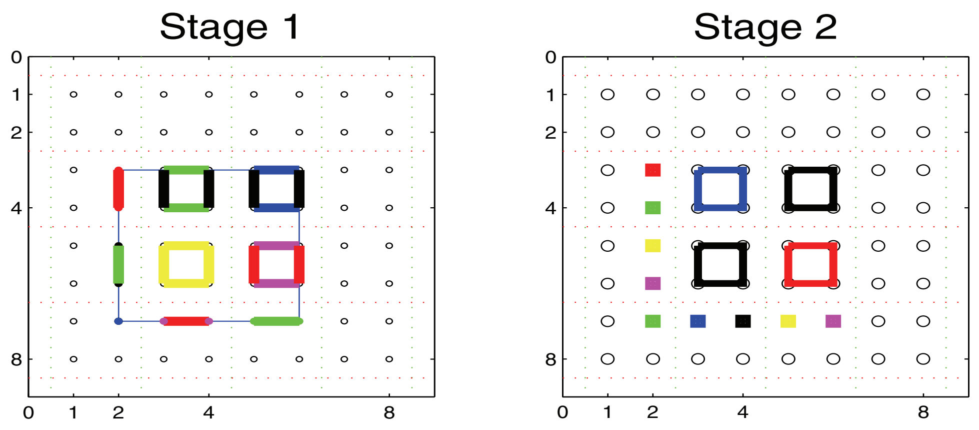

FIG. 1. *Illustration of $n = 8$ case: $\mathcal{S} = \{(\ell, m) : 3 \le \ell \le 7, 2 \le m \le 6\}$. Here the left plot shows how an interval in each coordinate direction is covered from coarse levels* 1, 2, *and the right plot shows how to convert the rectangular coverings into regular boxes.*

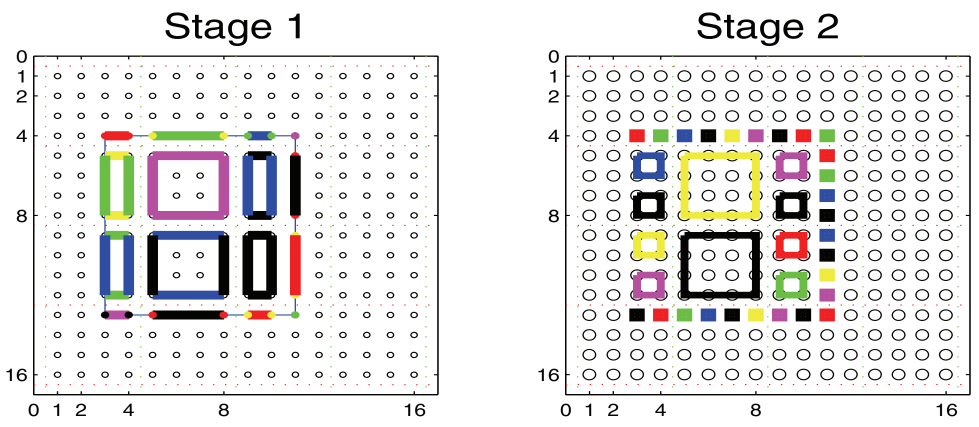

FIG. 2. *Illustration of $n = 16$ case: $\mathcal{S} = \{(\ell, m) : 5 \le \ell \le 13, 3 \le m \le 11\}$. Here the left plot shows how an interval in each coordinate direction is covered from coarse levels* 1, 2, 3, *and the right plot shows how to convert the rectangular coverings into regular boxes.*

ALGORITHM 4 (partition of a patch $\mathcal{S}$ into blocks on coarse levels). *Suppose that $\mathcal{S}$ has been defined by (25), as illustrated in Figure* 1 *where n, i_1, j_1, i_2, and j_2 are all indices on the finest level.*

Stage 1: *Level location.*

- *It suffices to discuss the x-direction $[i_1, i_2]$. Starting from the coarsest level with block size 2^L and working down to the finest level with size $2^0 = 1$, segment the interval $[i_1, i_2]$ into v subintervals of the maximum length possible falling across the levels to obtain the information vectors:*

  $Lev = [Lev_1, Lev_2, \dots, Lev_v]$ *for indicating the level information*
  $Sta = [Sta_1, Sta_2, \dots, Sta_v]$ *for the starting index information*
  $Siz = [Siz_1, Siz_2, \dots, Siz_v]$ *for indicating the block size.*

  (Here one of the vectors Lev, Siz is not necessary.)
- *Likewise, in the y-direction, the interval $[j_1, j_2]$ is split into w subintervals:*

  $Ley = [Ley_1, Ley_2, \dots, Ley_w]$ *for indicating the level information*

```
        $Sty = [Sty_1, Sty_2, \ldots, Sty_w]$ for the starting index information
        $Siy = [Siy_1, Siy_2, \ldots, Siy_w]$ for indicating the block size.
  • In the case of $\max_j Lev_j > \max_j Ley_j$, we shall resegment $[i_1, i_2]$ starting on
    the coarse level $\max_j Ley_j$.
Stage 2: Level matching.
  • Set $m = 0$.
  • for $i = 1, 2, \ldots, v$
      for $j = 1, 2, \ldots, w$
        if $Siz_i = Siy_j$, advance $m = m + 1$ and accept a mesh box with the
            starting address $(Sta_i, Sty_j)$ on level $Lev_i$.
        if $Siz_i > Siy_j$, set $r = Siz_i/Siy_j$, advance $m = m + r$, and accept $r$
            smaller boxes on level $Ley_j$ of size $Sz_m = Siy_j$.
        if $Siz_i < Siy_j$, set $r = Siy_j/Siz_i$, advance $m = m + r$, and accept $r$
            smaller boxes on level $Lev_i$ of size $Sz_m = Siz_i$.
      end
    end
On exit, the following list of vectors are obtained:
    $Le = [Le_1, Le_2, \ldots, Le_m]$ for indicating the level information
    $St = [(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)]$ for the starting indices
    $Sz = [Sz_1, Sz_2, \ldots, Sz_m]$ for indicating the block size.
```

Here with Figure 1 where $n = 8$, $i_1 = 3$, $i_2 = 7$, $j_1 = 2$, $j_2 = 6$, we obtain $m = 13$ and $Le = [1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 1, 1, 1]$, $St = [(3,2), (4,2), (3,3), (3,5), (5,2), (6,2), (5,3), (5,5), (7,2), (7,3), (7,4), (7,5), (7,6)]$, $Sz = [1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 1, 1, 1]$. Similarly with Figure 2 where $n = 16, i_1 = 5, i_2 = 13, j_1 = 3, j_2 = 11$, we obtain $m = 36$ and the vectors Le, St, Sz as shown in the right plot of Figure 2.

Once we have localized $\mathcal{S}$ onto various levels, we can use the stored information T_1 (finest), $T_2, \ldots, T_{L+1}$ (coarsest) from Algorithm 2 to work out the partial sum vector w_t (which will be used for computing $w_t^T w_t$ and $w_t^T \widetilde{z}$).

Algorithm 5 (computation of w_t using the partition from Algorithm 4).

```
Set $w_t$ to be a zero vector.
for each block $j = 1, 2, \ldots, m$
  • set the level index $k = Le_j$ and note that the block has the starting global
    indices $(x_j, y_j)$;
  • work out the local indices $(\bar{x}_j, \bar{y}_j)$ on level $k$ by the simple relations:
        $\bar{x}_j = 1 + (x_j - 1)/2^{k-1}$, $\bar{y}_j = 1 + (y_j - 1)/2^{k-1}$;
  • add column $(\bar{y}_j - 1)2^{k-1} + \bar{x}_j$ of the partial sum matrix $k$ to vector $w_t$
end
(Note $T_k$ is the root matrix of the above partial sum matrix.)
```

Finally our new multilevel method for the combined denoising and deblurring problem for solving (19) is the following algorithm.

Algorithm 6. *Given z and an initial guess $\widetilde{u} = z$, with $L + 1$ levels,*

```
  Pre-calculation.
1. Apply Algorithm 2 to compute all root matrices $T_k$ and $w_t^T w_t = \|T_k\|_F^2$ for partial
   sum matrices on level $k = 1, 2, \ldots, L + 1$.
  Multilevel Iterations.
2. Iteration starts with $u_{old} = \widetilde{u}$ ($\widetilde{u}$ contains the initial guess before the first iteration
   and the updated solution at all later iterations).
   for $\nu$ times on each level $k = 1, 2, 3, \ldots, L + 1$:
        • Compute $\widetilde{z} = z - K\widetilde{u}$ and form $K^T\widetilde{z}$ via the FFT.
        • for each block on level $k$,
              form each $w_t^T\widetilde{z}$ from $K^T\widetilde{z}$ and compute the minimizer $c$ of (23).
          end
        • add all the corrections (from all blocks on level $k$), $\widetilde{u} = \widetilde{u} + P_k c$, where $P_k$ is
          the interpolation operator distributing $c_{i,j}$ to the corresponding $b \times b$ block on
          level $k$ as illustrated in (16).
    end
3.  On level $k = 1$, check the possible patch at each position $(i, j)$ for some small $\varepsilon$

        $patch = \{(i_\ell, j_\ell) \ : \ |u_{i_\ell, j_\ell} - u_{i,j}| < \varepsilon\}$

    assumed to be a generic block of $b_1 \times b_2$.
    First use Algorithm 5 to compute the partial sum vector $w_t$.
    Then implement the piecewise constant update for each block as with step 2.
4.  If $\|\widetilde{u} - u_{old}\|_2$ is small enough, exit with $u = \widetilde{u}$ or return to step 2 and continue
    with the next multilevel cycle.
```

We remark that this algorithm looks similar to but is different from Algorithm 1 in step 2. Specifically in Algorithm 1 for denoising both the Jacobi and the Gauss–Seidel ideas may be used. But here for the combined denoising and deblurring, we can adopt only the Jacobi idea of updating because otherwise the number of updates will be too high to maintain efficiency.

4. Numerical experiments.

Here we first demonstrate the effectiveness of the proposed multilevel method (Algorithm 6) in restoring some practical images and then attempt a comparison with two other methods. Ideally it would be useful to compare with other multilevel methods for solving the same problems. However, as remarked, there do not appear to exist other nonlinear multilevel methods (either of the FAS type or of the non-FAS optimization type); for linearized problems, linear multilevel methods are not immediately applicable because of the presence of the blurring (nonlocal) operator. However, there exist several works [4, 31] that propose to approximate the combined problem by a denoising-like problem in an operator splitting (fixed point) fashion, and then apply a linear multilevel method; essentially a dense matrix is approximated by a sparse one.

Four test images (with both noise and blur) are used in the following experiments (the bridge, the cameraman, the goldhill, and a synthetic image as shown in the left plot of Figures 3–6). The two methods selected for comparison with our method are the following:

- Method 1—the well-known primal-dual method (CGM) of [27]. This method is convenient to use. It is perhaps the only approach in which the Newton method converges as Newton-type methods do not converge for the original (primal) formulation. Unfortunately, attempts to develop a multilevel version of the approach have not been successful.
- Method 2—the fixed point method with the conjugate gradient solver preconditioned by a multilevel preconditioner [65]. Although this method is not multilevel, it does not assume any sparsity of the blurring operator (unlike [4, 31]).

As usual, restoration performance is quantitatively measured by the peak signal-to-noise ratio (PSNR)

$$PSNR = PSNR(r, u) = 10 \log_{10} \frac{255^2}{\frac{1}{mn}\sum_{i,j}(r_{i,j} - u_{i,j})^2},$$

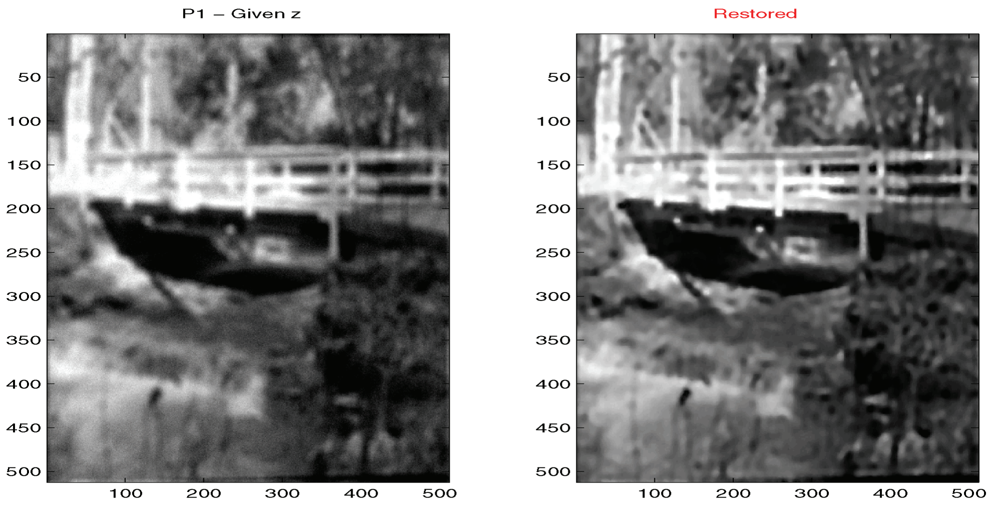

FIG. 3. *Test problem 1. Left: The noisy and blurred image. Right: The restored result from Algorithm 6.*

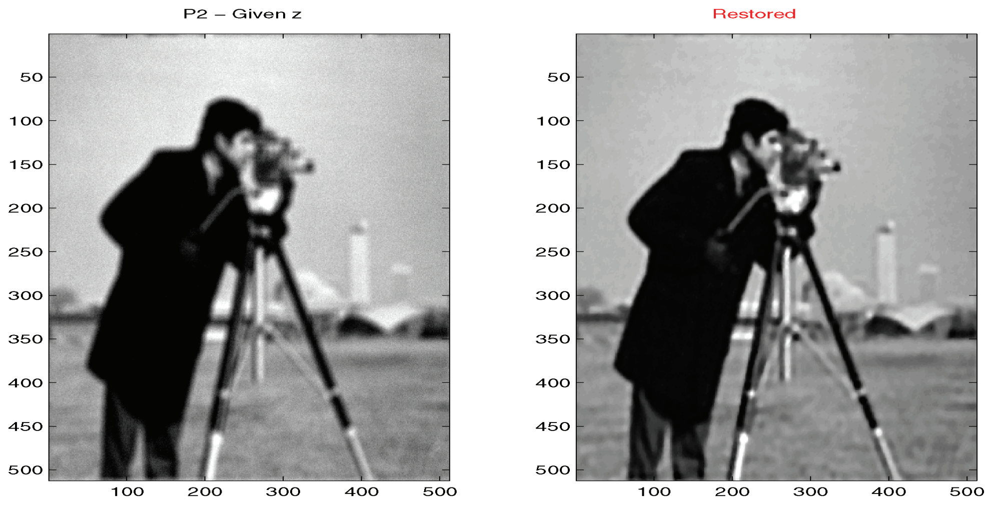

FIG. 4. *Test problem 2. Left: The noisy and blurred image. Right: The restored result from Algorithm 6.*

where $r_{i,j}$ and $u_{i,j}$ denote the pixel values of the original image and the restored image, respectively, with lexicographically ordered vectors $r, u \in \mathbb{R}^{mn}$. Here we assume $z_{i,j}, r_{i,j}, u_{i,j} \in [0, 255]$. The higher a PSNR is, the better the restoration quality is.

Quality of restoration. In the right plot of Figures 3–6, we show the test results, respectively, from our four test examples, in the resolution of $n \times n = 515 \times 512$. Clearly the restored images are visually pleasing.

Comparison with other methods and test of different resolutions. First, to compare with the above two chosen methods, our experiments are conducted in Matlab 2007b on a laptop with 4GB memory. The parameter $\alpha = 1$ is used in the first three examples and $\alpha = 2$ in the fourth example. Second, to see how our method behaves with increasing n, we also test them in several resolutions. The

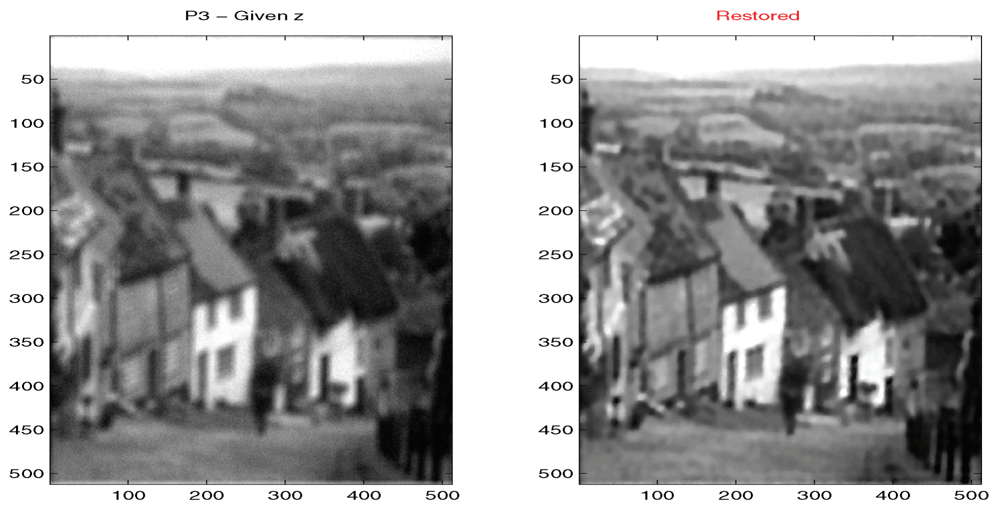

FIG. 5. *Test problem 3. Left: The noisy and blurred image. Right: The restored result from Algorithm 6.*

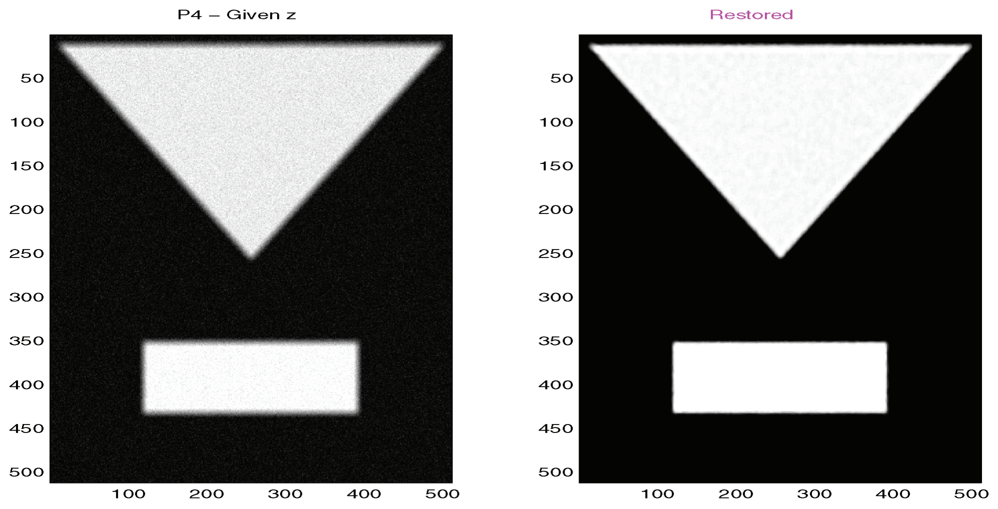

FIG. 6. *Test problem 4. Left: The noisy and blurred image. Right: The restored result from Algorithm 6.*

comparative results are displayed in Table 1, where "**" indicates that a method fails to get a result due to exhaustion of memory and "*" shows when the CPU is too excessive. Clearly we can see that (i) The CGM method [27] for smaller resolutions ($n \leq 256$) is about as fast as the fixed point method [65]. For large resolutions, it is not useful as its demand for storage (as well as for CPU) is prohibitive. (ii) The fixed point method [65] is more robust than the CGM because it performs better for larger resolutions and storage is not an issue. (iii) In obtaining a similar restoration quality, the new algorithm is definitely much faster and uses less memory than the first two methods. And more important, our algorithm displays the expected multilevel efficiency $O(N \log N) = O(n^2 \log n)$—when the number of unknowns is increased by

Table 1
Comparison of restoration quality and speed of $u_{MG} = \mathbf{u}$ with u_{CGM} and u_{FP} for various images of resolution $n \times n$ (with $N = n^2$ unknowns), where the CGM takes about 250 steps, the FP takes between 10 and 70 outer steps, and the multilevel algorithm takes about 3 cycles.

Problem	Size	CGM method [27]		FP ML method [65]		New algorithm 6	
number	n	PSNR	CPU	PSNR	CPU	PSNR	CPU
1	128	19.4	114.1	16.3	51.5	19.0	8.8
	256	20.9	644.8	17.6	460.6	20.8	35.2
	512	20.0	3326.4	18.8	352.5	19.9	145.5
	1024	**	**	21.0	12121.9	22.1	601.8
	2048	**	**	*	*	24.7	2510.1
2	128	20.5	113.2	16.5	51.3	19.9	8.6
	256	23.1	641.6	18.0	352.5	22.7	23.4
	512	21.6	3329.9	19.9	2416.9	21.7	146.4
	1024	**	**	22.7	19128.3	25.0	602.4
	2048	**	**	*	*	28.2	2524.8
3	128	21.9	112.7	18.6	83.9	21.7	8.6
	256	24.2	644.5	20.7	654.7	24.1	34.5
	512	23.1	3340.8	22.1	1759.6	23.2	147.3
	1024	**	**	24.3	12261.2	25.7	602.9
	2048	**	**	*	*	28.5	2505.2
4	128	22.2	112.8	14.3	138.6	21.8	8.7
	256	25.0	644.5	17.7	551.6	24.9	35.2
	512	22.6	3341.7	20.1	3377.5	22.8	147.1
	1024	**	**	23.9	16808.0	25.5	604.9
	2048	**	**	*	*	28.0	2505.7

a factor of 4 (from $N = n \times n = 2^L \times 2^L$ pixels to $2n \times 2n = 2^{L+1} \times 2^{L+1}$), the CPU increase is also by a factor of only slightly more than 4 or precisely by $\frac{L+1}{L}4$.

5. Conclusions.

5. Conclusions. While there exist many multilevel methods (both linear and nonlinear) to solve the image denoising model, the only multilevel method proposed for the challenging model of combined denoising and deblurring using the total variation regularization is a linear multigrid method in the fixed point framework where the blur operator is approximated by a sparse version.

In this paper, we have generalized an optimization based multilevel method previously proposed for the Gaussian denoising model to solve the combined denoising and deblurring model. The new nonlinear algorithm is found to give good restoration results in $O(N \log N)$ complexity. With a nearly optimal complexity, the proposed multilevel method offers a fast solution procedure for extremely large images.

Acknowledgments. The authors wish to thank the Chinese University of Hong Kong for the award of a science research grant CUHK DAG grant 2060257 to support this work.

REFERENCES

[1] R. Acar and C. R. Vogel, *Analysis of total variation penalty method for ill-posed problems*, Inverse Problems, 10 (1994), pp. 1217-1229.
[2] L. Alvarez, P.-L. Lions, and J.-M. Morel, *Image selective smoothing and edge detection by nonlinear diffusion II*, SIAM J. Numer. Anal., 29 (1992), pp. 845–866.
[3] H. C. Andrews and B. R. Hunt, *Digital Image Restoration*, Prentice-Hall, Englewood Cliffs, NJ, 1977.
[4] U. M. Ascher, E. Haber, and H. Huang, *On effective methods for implicit piecewise smooth surface recovery*, SIAM J. Sci. Comput., 28 (2006), pp. 339–358.
[5] D. P. Bertsekas, *Nonlinear Programming*, Athena, Belmont, MA, 1995.

[6] J. C. Bezdek and R. J. Hathaway, *Convergence of alternating optimization*, Neural Parallel Sci. Comput., 11 (2003), pp. 351–368.

[7] P. Blomgren, T. F. Chan, P. Mulet, L. Vese, and W. L. Wan, *Variational PDE models and methods for image processing*, in Research Notes in Mathematics 420, Chapman & Hall/CRC, London, 2000, pp. 43–67.

[8] A. Brandt, *Multigrid solvers and multilevel optimization strategies*, in Multiscale Optimization and VLSI/CAD, J. Cong and J. R. Shinnerl, eds., Kluwer Academic, Boston, 2000, pp. 1–68.

[9] A. Bruhn, J. Weickert, T. Kohlberger, and C. Schnörr, *A Multigrid Platform for Real-Time Motion Computation with Discontinuity-Preserving Variational Methods*, Technical report 136, Department of Mathematics, Saarland University, Saarbrücken, Germany, 2005.

[10] M. Burger, S. Osher, J. Xu, and G. Gilboa, *Nonlinear inverse scale space methods for image restoration*, Commun. Math. Sci., 4 (2006), pp. 179–212.

[11] J. F. Cai, R. H. Chan, and B. Morini, *Minimization of edge-preserving regularization functional by conjugate gradient type methods*, in Image Processing Based on Partial Differential Equations, X.-C. Tai, K.-A. Lie, T. F. Chan, and S. Osher, eds., Springer-Verlag, Berlin, 2006, pp. 109–122.

[12] D. Calvetti, B. Lewis, and L. Reichel, *Krylov subspace iterative methods for nonsymmetric discrete ill-posed problems in image restoration*, Advanced Signal Processing Algorithms, Architectures, and Implementations X, F. T. Luk, ed., in Proceedings of the Society of Photo-Optical Instrumentation Engineers (SPIE), 4116, The International Society for Optical Engineering, Bellingham, WA, 2001.

[13] J. L. Carter, *Dual Method for Total Variation-Based Image Restoration*, CAM report 02-13, UCLA, Los Angeles, CA, 2002.

[14] A. Chambolle, *An algorithm for total variation minimization and applications*, J. Math. Imaging Vision, 20 (2004), pp. 89–97.

[15] A. Chambolle and P. L. Lions, *Image recovery via total variation minimization and related problems*, Numer. Math., 76 (1997), pp. 167–188.

[16] R. H. Chan, T. F. Chan, and W. L. Wan, *Multigrid for differential convolution problems arising from image processing*, in Proceedings of the Scientific Computing Workshop, R. Chan, T. F. Chan and G. H. Golub, eds., Springer-Verlag, Berlin, 1997.

[17] R. H. Chan, T. F. Chan, and C. K. Wong, *Cosine transform based preconditioners for total variation minimization problems in image processing*, IEEE Trans. Image Process., 8 (1999), pp. 1472–1478.

[18] R. H. Chan, Q.-S. Chang, and H.-W. Sun, *Multigrid method for ill-conditioned symmetric Toeplitz systems*, SIAM J. Sci. Comput., 19 (1998), pp. 516–529.

[19] R. H. Chan and K. Chen, *Multilevel algorithm for a Poisson noise removal model with total variation regularisation*, Int. J. Comput. Math., 84 (2007), pp. 1183-1198.

[20] R. H. Chan and K. Chen, *Fast multilevel algorithm for a minimization problem in impulse noise removal*, SIAM J. Sci. Comput., 30 (2008), pp. 1474–1489.

[21] R. H. Chan and X. Q. Jin, *An Introduction to Iterative Toeplitz Solvers*, SIAM, Philadelphia, 2007.

[22] T. F. Chan and K. Chen, *On a nonlinear multigrid algorithm with primal relaxation for the image total variation minimisation*, J. Numer. Algorithms, 41 (2006), pp. 387–411.

[23] T. F. Chan and K. Chen, *An optimization-based multilevel algorithm for total variation image denoising*, Multiscale Model. Simul., 5 (2006), pp. 615–645.

[24] T. F. Chan, K. Chen, and X. C. Tai, *Nonlinear multilevel schemes for solving the total variation image minimization problem*, in Image Processing Based on Partial Differential Equations, X.-C. Tai, K.-A. Lie, T. F. Chan, and S. Osher, eds., Springer-Verlag, Berlin, 2006, pp. 265–288.

[25] T. F. Chan, K. Chen, and J. L. Carter, *On multigrid methods for the dual formulation of variational image processing*, Electron. Trans. Numer. Anal., 26 (2007), pp. 299–311.

[26] T. F. Chan and S. Esedoglu, *Aspects of total variation regularized L^1 function approximation*, SIAM J. Appl. Math., 65 (2005), pp. 1817–1837.

[27] T. F. Chan, G. H. Golub, and P. Mulet, *A nonlinear primal-dual method for total variation-based image restoration*, SIAM J. Sci. Comput., 20 (1999), pp. 1964–1977.

[28] T. F. Chan and P. Mulet, *Iterative methods for total variation restoration*, in Iterative Methods in Scientific Computing, R. H. Chan, T. F. Chan, and G. H. Golub, eds., Springer-Verlag, Berlin, 1997, pp. 359–381.

[29] T. F. Chan and J. H. Shen, *Image Processing and Analysis—Variational, PDE, Wavelet, and Stochastic Methods*, SIAM, Philadelphia, 2005.

[30] Q. S. Chang and I.-L. Chern, *Acceleration methods for total variation-based image denoising*, SIAM J. Sci. Comput., 25 (2003), pp. 982–994.

[31] Q. S. Chang, W. C. Wang, and J. Xu, *A method for total variation-based reconstruction of noisy and blurred images*, in Image Processing Based on Partial Differential Equations, X.-C. Tai, K.-A. Lie, T. F. Chan, and S. Osher, eds., Springer-Verlag, Berlin, 2007, pp. 95–108.

[32] K. Chen, *Matrix Preconditioning Techniques and Applications*, Cambridge Monographs on Applied and Computational Mathematics 19, Cambridge University Press, Cambridge, UK, 2005.

[33] K. Chen and X. C. Tai, *A nonlinear multigrid method for total variation minimization from image restoration*, J. Sci. Comput., 33 (2007), pp. 115–138.

[34] J. Darbon and M. Sigelle, *A fast and exact algorithm for total variation minimization*, in Proceedings of the 2nd Iberian Conference on Pattern Recognition and Image Analysis, J. S. Marques, N. Pérez de la Blanca, and P. Pina, eds., Lecture Notes in Computer Science 3522, Springer-Verlag, Berlin, 2005, pp. 351–359.

[35] M. Donatelli and S. Serra-Capizzano, *On the regularizing power of multigrid-type algorithms*, SIAM J. Sci. Comput., 27 (2006), pp. 2053–2076.

[36] C. Frohn-Schauf, S. Henn, and K. Witsch, *Nonlinear multigrid methods for total variation image denoising*, Comput. Vis. Sci., 7 (2004), pp. 199–206.

[37] D. Goldfarb and W. T. Yin, *Second-order cone programming methods for total variation-based image restoration*, SIAM J. Sci. Comput., 27 (2005), pp. 622–645.

[38] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Addison-Wesley, Reading, MA, 1993.

[39] S. Gratton, A. Sartenaer, and P. L. Toint, *Recursive trust-region methods for multiscale nonlinear optimization*, SIAM J. Optim., 19 (2008), pp. 414–444.

[40] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comput. Phys., 73 (1987), pp. 325–348.

[41] M. Heinkenschloss, M. B. Hribar, and M. Kokkolaras, *Acceleration of multidisciplinary analysis solvers by inexact subsystem simulations*, Paper 98-4712, in Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, MO, 1998.

[42] M. Hintermüller and K. Kunisch, *Total bounded variation regularization as a bilaterally constrained optimization problem*, SIAM J. Appl. Math., 64 (2004), pp. 1311–1333.

[43] M. Hintermüller and G. Stadler, *An infeasible primal-dual algorithm for total bounded variation–based inf-convolution-type image restoration*, SIAM J. Sci. Comput., 28 (2006), pp. 1–23.

[44] Y. M. Huang, M. K. Ng, and Y.-W. Wen, *A fast total variation minimization method for image restoration*, Multiscale Model. Simul., 7 (2008), pp. 774–795.

[45] J. Idier, *Convex half-quadratic criteria and interacting auxiliary variables for image restoration*, IEEE Trans. Image Process., 10 (2001), pp. 1001–1009.

[46] T. Kärkkäinen, K. Majava, and M. M. Mäkelä, *Comparison of Formulations and Solution Methods for Image Restoration Problems*, Series B Report B 14/2000, Department of Mathematical Information Technology, University of Jyväskylä, Jyväskylä, Finland, 2000.

[47] R. Kornhuber, *Monotone multigrid methods for variational inequalities II*, Numer. Math., 72 (1996), pp. 481–499.

[48] S. H. Lee and J. K. Seo, *Noise removal with Gauss curvature driven diffusion*, IEEE Trans. Image Process., 14 (2005), pp. 904-909.

[49] Y. Y. Li and F. Santosa, *A computational algorithm for minimizing total variation in image restoration*, IEEE Trans. Image Process., 5 (1996), pp. 987–995.

[50] A. Marquina and S. Osher, *Explicit algorithms for a new time dependent model based on level set motion for nonlinear deblurring and noise removal*, SIAM J. Sci. Comput., 22 (2000), pp. 387–405.

[51] S. Nash, *A multigrid approach to discretized optimisation problems*, J. Opt. Methods Softw., 14 (2000), pp. 99–116.

[52] M. Ng and N. K. Bose, *Mathematical analysis of super-resolution methodology*, IEEE Signal Proc. Magazine, 20 (2003), pp. 62–74.

[53] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic, New York, 1970.

[54] S. Osher and S. Esedoglu, *Decomposition of images by the anisotropic Rudin-Osher-Fatemi model*, CAM Report 03-34, UCLA, Los Angeles, CA, 2003.

[55] W. H. Press et al., *Numerical Recipes in C*, Cambridge University Press, London, 1992.

[56] E. Radmoser, O. Scherzer, and J. Schöberl, *A Cascadic Algorithm for Bounded Variation Regularization*, SFB-Report 00-23, Johannes Kepler University of Linz, Linz, Austria, 2000.

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

[58] G. Sapiro, *Geometrical Differential Equations and Image Analysis*, Cambridge University Press, London, 2001.

[59] J. Savage and K. Chen, *An improved and accelerated nonlinear multigrid method for total variation denoising*, Int. J. Comput. Math., 82 (2005), pp. 1001–1015.

[60] S. Ta'asan, *Lecture note 4 of Von-Karman Institute Lectures*, Belgium, http://www.math.cmu.edu/∼shlomo/VKI-Lectures/lecture4, 1997.

[61] X. C. Tai and J. C. Xu, *Global and uniform convergence of subspace correction methods for some convex optimization problems*, Math. Comp., 71 (2001), pp. 105–124.

[62] P. Tseng, *Convergence of a block coordinate descent method for nondifferentiable minimization*, J. Optim. Theory Appl., 109 (2001), pp. 475–494.

[63] C. R. Vogel, *A multigrid method for total variation-based image denoising*, in Computation and Control IV, K. Bowers and J. Lund, eds., Progress in Systems and Control Theory 20, Birkhäuser, Basel, Switzerland, 1995.

[64] C. R. Vogel, *Negative results for multilevel precondtioners in image deblurring*, in Scale-space theories in computer vision, M. Nielson et al., eds., Springer-Verlag, Berlin, 1999, pp. 292–304.

[65] C. R. Vogel, *Computational Methods for Inverse Problems*, SIAM, Philadelphia, 2002.

[66] C. R. Vogel and M. E. Oman, *Iterative methods for total variation denoising*, SIAM J. Sci. Comput., 17 (1996), pp. 227–238.

[67] C. R. Vogel and M. E. Oman, *Fast, robust total variation-based reconstruction of noisy, blurred images*, IEEE Trans. Imag. Process., 7 (1998), pp. 813–824.

[68] Y. L. Wang, J. F. Yang, W. T. Yin, and Y. Zhang, *A new alternating minimization algorithm for total variation image reconstruction*, SIAM J. Imag. Sci., 1 (2008), pp. 248–272.