SIAM J. IMAGING SCIENCES
Vol. 3, No. 3, pp. 363–389

Multigrid Algorithm for High Order Denoising*

Carlos Brito-Loeza† and Ke Chen‡

Abstract. Image denoising has been a research topic deeply investigated within the last two decades. Excellent results have been obtained by using such models as the total variation (TV) minimization by Rudin, Osher, and Fatemi [*Phys. D*, 60 (1992), pp. 259–268], which involves solving a second order PDE. In more recent years some effort has been made [Y.-L. You and M. Kaveh, *IEEE Trans. Image Process.*, 9 (2000), pp. 1723–1730; M. Lysaker, S. Osher, and X.-C. Tai, *IEEE Trans. Image Process.*, 13 (2004), pp. 1345–1357; M. Lysaker, A. Lundervold, and X.-C. Tai, *IEEE Trans. Image Process.*, 12 (2003), pp. 1579–1590; Y. Chen, S. Levine, and M. Rao, *SIAM J. Appl. Math.*, 66 (2006), pp. 1383–1406] in improving these results by using higher order models, particularly to avoid the staircase effect inherent to the solution of the TV model. However, the construction of stable numerical schemes for the resulting PDEs arising from the minimization of such high order models has proved to be very difficult due to high nonlinearity and stiffness. In this paper, we study a curvature-based energy minimizing model [W. Zhu and T. F. Chan, *Image Denoising Using Mean Curvature*, preprint, http://www.math.nyu.edu/~wzhu/], for which one has to solve a fourth order PDE. For this model we develop two new algorithms: a stabilized fixed point method and, based upon this, an efficient nonlinear multigrid (MG) algorithm. We will show numerical experiments to demonstrate the very good performance of our MG algorithm.

Key words. denoising, variational models, regularization, fourth order partial differential equations, multilevel methods

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

DOI. 10.1137/080737903

1. Introduction.

Image denoising is a basic but very important image processing task that has been extensively investigated for many years. Although there exist different types of noise, here we study only algorithms for removing additive, zero-mean Gaussian noise. This can be modeled as

$$z(x,y) = u(x,y) + \eta(x,y), \tag{1.1}$$

where $z = z(x,y)$ is the known noisy image, $u = u(x,y)$ is the unknown true image, and $\eta = \eta(x,y)$ is the unknown additive noise, all of which are defined on a domain $\Omega \subseteq \mathbb{R}^2$. The task of removing noise can be accomplished in traditional ways such as linear filters, which, though very simple to implement, may cause the restored image to be blurred at the edges. A

*Received by the editors October 14, 2008; accepted for publication (in revised form) May 18, 2010; published electronically August 19, 2010.
http://www.siam.org/journals/siims/3-3/73790.html
†Centre for Mathematical Imaging Techniques and Department of Mathematical Sciences, The University of Liverpool, Peach Street, Liverpool L69 7ZL, United Kingdom (cbrito@liverpool.ac.uk).
‡Corresponding author. Centre for Mathematical Imaging Techniques and Department of Mathematical Sciences, The University of Liverpool, Peach Street, Liverpool L69 7ZL, United Kingdom (k.chen@liverpool.ac.uk, http://www.liv.ac.uk/~cmchenke/cmit).

much better technique is to use nonlinear PDEs as anisotropic diffusion filters because they apply different strengths of diffusivity to different locations in the image.

Usually anisotropic filters are implemented as second order PDEs; see, for instance, [41, 39]. The main drawback of these models is that they convert smooth functions into piecewise constant functions in a phenomenon known as the *staircase effect*, which causes images to look blocky. Although some effort has been made to numerically reduce the staircase effect in second order models (see, for instance, [37, 43, 23] and the references therein), some researchers trying to avoid this problem have turned to higher order models. In this direction are, for instance, the works presented in [55, 36, 35, 56]. However, very few papers have touched on fast solvers for these high order models, and this work aims to partially fill this gap.

The outline of this paper is as follows. In section 2 we introduce the model to be solved. In section 3 we review the existing numerical methods for solving this model. In section 4 the numerical discretization of the resulting Euler–Lagrange (EL) equation is described. We use section 5 to introduce our two new algorithms for solving this EL equation: a stabilized fixed point (SFP) method and a nonlinear multigrid (MG) method. We introduce some early work on using MG algorithms for similar problems and explain the main difficulties to be overcome. We also use this section to present a detailed local Fourier analysis of the linearized problem to have an insight into the performance of the nonlinear MG algorithm. A complexity analysis is presented as well. In section 6, we present numerical evidence to show the very good and fast performance of the MG algorithm and explain how it is affected by variations on the different parameters of the model and the numerical equation. Finally, in section 7 we discuss how our algorithms can be adapted to solve similar high order problems, and we present our conclusions in section 8.

2. High order denoising model.

Additive noise in images is seen as random high frequency oscillations. Therefore energy minimization–based techniques represented as

$$\min_u \left\{ E(u) \equiv \alpha R(u) + \int_\Omega |u - z|^2 \, dxdy \right\} \tag{2.1}$$

attempt to damp such oscillations by the regularization term $R(u)$, and the key for noise removal is to select a suitable $R(u)$ capable of efficiently measuring oscillations. Different high order approaches for $R(u)$ have been proposed. For example, [55, 35, 36, 13] all use second order information (i.e., second order derivatives), so it is expected for them to be able to model noise better than those using only first order information such as the well-known total variation (TV) model [41, 18].

Curvature-based denoising model. In this paper, we study the model of [56] resulting from selecting $R(u) = \int_\Omega \Phi(\kappa) \, dxdy$ with κ the curvature of the image and Φ defined as either $\Phi(\kappa) = |\kappa|$ or $\Phi(\kappa) = \kappa^2$ or, as in [56], as a combination of both.

To minimize (2.1) one could be tempted to use optimization algorithms such as Newton's method. There is, however, a problem with this approach; after computing the first order condition, the resulting algebraic system of equations is highly nonlinear and has a reduced domain of convergence, causing Newton's method to fail since it requires a very good initial guess to guarantee convergence. This fact is not surprising since a similar problem was reported in [51, 20, 17] when solving the very similar formulation of the TV denoising model.

Multilevel optimization methods of [14], on the other hand, still need to be tested for high order problems.

Our approach, instead, is to minimize (2.1) by solving its EL equation. This method has proved to deliver quality restoration results in a wide variety of image processing applications; see [18, 3, 50] for references. The EL equation we aim to solve is

$$\alpha\nabla\cdot\left(\frac{\nabla\Phi'(\kappa)}{|\nabla u|_\beta}-\frac{\nabla u\cdot\nabla\Phi'(\kappa)}{(|\nabla u|_\beta)^3}\nabla u\right)+u-z=0 \qquad \text{in } \Omega \tag{2.2}$$

with Neumann boundary condition $\nabla u\cdot\vec{\nu}=0$ on $\partial\Omega$. Note that we already have applied regularization to avoid division by zero by replacing $|\nabla u|$ with $|\nabla u|_\beta=\sqrt{|\nabla u|^2+\beta}$. For simplicity from now on we will write the derivative of $\Phi(\kappa)$ as just Φ' instead of $\Phi'(\kappa)$.

A remark is in order here: In [56] an image is understood as a surface represented by $(x,y,u(x,y))$ where initially $u(x,y)=z$. In this representation, the curvature term κ that appears in (2.2) is therefore the curvature of the image surface and is defined by $\kappa\equiv\kappa_S=\nabla\cdot\frac{\nabla u}{\sqrt{|\nabla u|^2+1}}$. One can adopt the more common understanding of an image as a collection of level sets and still obtain the same PDE but this time with $\kappa\equiv\kappa_{LS}=\nabla\cdot\frac{\nabla u}{|\nabla u|}$ standing for the curvature at every level line or isophote of the image; see [18, 3]. Note that when κ_{LS} is regularized using a β-parameter as above, $\kappa_{LS}=\nabla\cdot\frac{\nabla u}{|\nabla u|_\beta}$ equals κ_S for $\beta=1$.

The selection of β is actually of great importance in numerical implementations since for $\beta\ll 1$ the anisotropy of (2.2) is increased, making this model less suitable for MG algorithms. Regardless of this fact, the MG algorithm we develop here is in this sense very general, allowing us to select a relatively small value for β or $\beta=1$ and still obtain very good performance. Moreover, an interesting discussion about the *correct* value of β for a similar second order problem can be found in [2]. There, the authors showed that there is a range of values of β for which the model delivers a good reconstruction; these values are not necessarily extremely small.

3. Review of numerical methods.

We start this section by remarking that to solve (2.2) only an explicit time-marching (and slow) scheme has been proposed [56]. Before introducing our numerical algorithm, we will briefly explain the main difficulties and explore possible options based upon ideas developed in [37, 36, 49]. To this end, first rewrite the EL equation as

$$\alpha\nabla\cdot\big(D_1(u)\nabla\Phi'-D_2(u)\nabla u\big)+u-z=0, \tag{3.1}$$

where

$$D_1(u)=\frac{1}{|\nabla u|_\beta} \quad\text{and}\quad D_2(u)=\frac{\nabla u\cdot\nabla\Phi'}{(|\nabla u|_\beta)^3} \tag{3.2}$$

are diffusion coefficients whose numerical values depend mainly on the values of their respective denominators. Due to noise present in u and the edges of u itself, D_1 and D_2 are usually discontinuous coefficients on Ω, causing (3.1) to be highly anisotropic.

In the literature we can find similar PDEs having only D_1-type coefficients (for example, the TV denoising PDE of [41]), and such PDE models represent a class of challenging problems

in developing fast and stable numerical algorithms; see [14, 15, 19, 20, 31, 37, 51, 42] and the references therein. The PDE (3.1) is even more challenging since it contains both D_1- and D_2-type coefficients. For example, fixing $\beta = 10^{-4}$ in a plain region of the image yields $D_1 \sim O(10^2)$ compared to $D_2 \sim O(10^6)$. Hence, depending upon the smoothness of the image and the level of noise, this phenomenon can produce a very unbalanced discrete operator.

Solving (3.1) with an explicit method as in [56] has the drawback that the time step needs to be selected extremely small for stability reasons. To implement this method, first (3.1) is transformed into the parabolic form

$$\frac{\partial u}{\partial t} = \alpha \nabla \cdot V(u) + u - z \equiv r(u), \tag{3.3}$$

with $V = (D_1(u)\nabla\Phi' - D_2(u)\nabla u)$ and initial condition $u(x, y, 0) = z(x, y)$, and then (3.3) is evolved in time until reaching steady state using the easy to implement, but very slow to converge, explicit Euler method described below:

$$u_{i,j}^{k+1} = u_{i,j}^{k} + \Delta t\, r(u)_{i,j}^{k}, \quad \text{with } k = 0, 1, \ldots \text{ and } \Delta t \text{ the time step.} \tag{3.4}$$

One way to accelerate the convergence of the explicit method could be by multiplying $r(u)$ by $|\nabla u|$. This results in the scheme $\frac{\partial u}{\partial t} = |\nabla u|_{i,j}^{k} r(u)_{i,j}^{k}$, where, again, an explicit Euler scheme can be used for the time derivative. This idea was applied to similar PDEs in [37, 16] with some success but failed to deliver considerable acceleration of the explicit method when it was applied to (3.1).

In summary, the above two explicit methods have the inconvenience of having to obey a severe restriction on the time step. Usually $\Delta t \sim O((\Delta x)^4)$ for fourth order PDEs, which in our case implies that both schemes are practically of no use for processing large images.

An alternative option could be to find a suitable change of variables, obtaining an easier-to-solve system of second order equations. This was done in [36, 49] for harmonics maps, letting the authors solve the problem indirectly. Unfortunately, this does not seem to be straightforward here. A last option based upon convexity splitting ideas [27, 28] will be studied in section 5.3.

4. Numerical implementation. We proceed to outline the discretization scheme that we use. From now on, we assume[1] a continuous domain $\Omega = [0, m]\times[0, n]$ and let (h_x, h_y) represent a vector of finite mesh sizes; then we define the infinite grid by $G_h = \{(x, y) : x = x_i = ih_x,$ $y = y_j = jh_y;\ i, j \in \mathbb{Z}\}$, $\Omega_h = \Omega \cap G_h$, and denote by $u_h = u_h(x, y) = u_h(x_i, y_j) = u_h(ih_x, jh_y)$ the discrete version of any function u defined on Ω_h. For simplicity we will assume that $m = n$, $h = h_x = h_y$, and u and z will take on scaled values in the interval $[0, 1]$. We will use $(\cdot)_\psi$ to denote the derivative with respect to any variable ψ.

To approximate $\nabla \cdot V = (V^1)_x + (V^2)_y$ for any $V = (V^1, V^2)$ at some pixel (i, j) we use central differences between ghost half-points as follows:

$$\nabla \cdot V_{i,j} = \frac{\left(V^1_{i+\frac{1}{2},j} - V^1_{i-\frac{1}{2},j}\right)}{h} + \frac{\left(V^2_{i,j+\frac{1}{2}} - V^2_{i,j-\frac{1}{2}}\right)}{h}, \tag{4.1}$$

[1]The choice of $\Omega = [0, m] \times [0, n]$ ensures that $h = 1$ on the finest grid without loss of generality.

where $h \times h$ is the size of one cell on the cell-centered grid Ω_h. When appropriate we use min-mod derivatives as defined in the following since, as noted in [41, 18], they help to recover sharp edges:

$$\text{min-mod}\,(a,b) = \left(\frac{\operatorname{sgn} a + \operatorname{sgn} \mathrm{b}}{2}\right) \min(|a|,|b|). \tag{4.2}$$

To compute $V^1_{i+\frac{1}{2},j}$ and $V^1_{i-\frac{1}{2},j}$ at the half-points, we proceed in the following way:
Curvature by

$$\kappa_{i,j} = \frac{(u_x)_{i+\frac{1}{2},j}}{|\nabla u|_{i+\frac{1}{2},j}} - \frac{(u_x)_{i-\frac{1}{2},j}}{|\nabla u|_{i-\frac{1}{2},j}} + \frac{(u_y)_{i,j+\frac{1}{2}}}{|\nabla u|_{i,j+\frac{1}{2}}} - \frac{(u_y)_{i,j-\frac{1}{2}}}{|\nabla u|_{i,j-\frac{1}{2}}}.$$

Partial derivatives in x by the central differencing of two adjacent *whole* pixels:

$$\begin{aligned}
(u_x)_{i+\frac{1}{2},j} &= (u_{i+1,j} - u_{i,j})/h,\\
(u_x)_{i-\frac{1}{2},j} &= (u_{i,j} - u_{i-1,j})/h,\\
(\Phi'_x)_{i+\frac{1}{2},j} &= \left(\Phi'_{i+1,j} - \Phi'_{i,j}\right)/h,\\
(\Phi'_x)_{i-\frac{1}{2},j} &= \left(\Phi'_{i,j} - \Phi'_{i-1,j}\right)/h,\\
|\nabla u|_{i+\frac{1}{2},j} &= \sqrt{((u_x)_{i+\frac{1}{2},j})^2 + ((u_y)_{i,j+\frac{1}{2}})^2 + \beta}.
\end{aligned}$$

Partial derivatives in y by the min-mod of $(\cdot)_y$'s at two adjacent *whole* points:

$$\begin{aligned}
(u_y)_{i+\frac{1}{2},j} &= \text{min-mod}\left(\tfrac{1}{2h}(u_{i+1,j+1} - u_{i+1,j-1}), \tfrac{1}{2h}(u_{i,j+1} - u_{i,j-1})\right),\\
(u_y)_{i-\frac{1}{2},j} &= \text{min-mod}\left(\tfrac{1}{2h}(u_{i,j+1} - u_{i,j-1}), \tfrac{1}{2h}(u_{i-1,j+1} - u_{i-1,j-1})\right),
\end{aligned}$$

$$\begin{aligned}
&(\Phi'_y)_{i+\frac{1}{2},j} = \text{min-mod}\,(\zeta,\vartheta) \quad \text{with}\\
&\zeta = \tfrac{1}{2h}\left(\Phi'_{i+1,j+1} - \Phi'_{i+1,j-1}\right) \quad \text{and} \quad \vartheta = \tfrac{1}{2h}\left(\Phi'_{i,j+1} - \Phi'_{i,j-1}\right),
\end{aligned}$$

$$\begin{aligned}
&(\Phi'_y)_{i-\frac{1}{2},j} = \text{min-mod}\,(\zeta,\vartheta) \quad \text{with}\\
&\zeta = \tfrac{1}{2}\left(\Phi'_{i,j+1} - \Phi'_{i,j-1}\right) \quad \text{and} \quad \vartheta = \tfrac{1}{2}\left(\Phi'_{i-1,j+1} - \Phi'_{i-1,j-1}\right),
\end{aligned}$$

$$|\nabla u|_{i-\frac{1}{2},j} = \sqrt{((u_x)_{i-\frac{1}{2},j})^2 + ((u_y)_{i,j-\frac{1}{2}})^2 + \beta}.$$

By a similar procedure we can obtain the approximations for $V^2_{i,j+\frac{1}{2}}$ and $V^2_{i,j-\frac{1}{2}}$. Finally, the Neumann's boundary condition on $\partial\Omega$ is treated as

$$u_{i,0} = u_{i,1}, \qquad u_{i,n+1} = u_{i,n}, \qquad u_{0,j} = u_{1,j}, \qquad u_{m+1,j} = u_{m,j}. \tag{4.3}$$

5. A nonlinear MG for a fourth order denoising model. To the best of our knowledge no MG method has been reported for the solution of a similar high order denoising problem which presents at the same time the challenges of dealing with nonlinearity, anisotropy, and high order derivatives. This situation is not surprising since the application of either standard linear or nonlinear MG with the typical known components does not converge. In section 5.3 we shall introduce a new nonstandard smoother for a full approximation scheme (FAS) MG algorithm.

5.1. Early works and numerical difficulties. Before moving to the development of such an FAS algorithm, we will briefly pause to review some early works on MG algorithms, most of them mainly in the context of image processing techniques. Our intention is to highlight the main difficulties in developing optimal MG algorithms for such problems.

We start with *nonlinear isotropic problems* where the functionals to be minimized have mostly been regularized using Tikhonov's idea with $\int_\Omega |\nabla u|^2$ and therefore their corresponding EL equations include Laplacian-like differential operators with the nonlinearity coming from the fitting term. It is well known that these strongly elliptic operators are suitable for MG algorithms, and very good performance can be obtained without very much effort. The approach used in these cases is usually to linearize the problem and apply a linear MG for the inner iterations (see, e.g., [31, 46, 25, 34]). In [45], however, a nonlinear MG was reported.

On *linear anisotropic problems*, interesting discussions about MG algorithms have been presented in [29, 19, 1, 40, 7], but we will not dig deep into this subject.

More interesting are *nonlinear anisotropic problems* where some work has been done as well. For example, the following works developed MG algorithms for image processing problems: [32] on image registration, [4, 38] on image segmentation, and [42, 43, 15, 30] on image denoising. All of the above, however, solve second order PDEs.

In particular, previous image denoising works are of interest since they give us a glimpse of the difficulties encountered in developing optimal MGs for this kind of problem. All of the image denoising works [42, 43, 15, 30] reported difficulties in obtaining optimal performance of geometric MG algorithms when the anisotropy of the problem associated with the TV regularizer $\int_\Omega |\nabla u|$ reached high levels. The anisotropy of denoising problems is mainly due to the value of the regularization parameter β, the level of noise η, and the smoothness of u itself, meaning that the stronger the edges present in u are, the more anisotropic the PDE that needs to be solved is. In [30], to overcome this problem, β was initially selected very large, and a continuation method with $\beta \to 0$ was implemented. At every step of this continuation method an MG cycle was used to solve the problem. In [42, 43, 15], a different approach was taken: A small increment on the number of smoothing steps ν of the MG algorithm was applied, and β no less than 10^{-2} was used.

For the curvature-based model we study here, the *first* issue is the level of discontinuity of the coefficients D_1 and D_2 in (3.1). That is, if they are moderately discontinuous, then the nonlinear operator can be fairly approximated on coarser grids. However, for strongly discontinuous coefficients (say $\beta \ll 1$), the performance of the MG algorithm will be strongly affected. Notice that in image denoising problems, since every image is different and because of noise, we do not have a priori information about which variables (if any) are strongly coupled and in which direction; hence standard methods such as line relaxation and semicoarsening are not useful here.

To overcome this challenge, some authors have proposed using algebraic multigrid (AMG) methods, where the coarsening is adapted to the structure of the domain itself. Here, because the domain of every image is different, this type of coarsening has to be adaptive, making it computationally very expensive; see [22, 47] and the references therein. In [54] a geometric MG algorithm with adaptive coarsening for the anisotropic Cahn–Hilliard equation was developed by the authors using the simple rule of coarsening only where discontinuities were not present. We believe this technique can be adapted to MG algorithms for image denoising by coarsening

only at plain regions of the image, and although it looks promising, some tests need to be carried out to confirm its effectiveness. In this paper we adopted the geometric MG scheme with standard coarsening due to its simplicity and focus on nonstandard smoothers.

A *second* issue for our model is the transportation of the error from one grid to another. Geometric MG algorithms assume that the error is smooth enough so it can be well approximated onto the next coarser grid by using a simple transportation operator. Designing good smoothers for anisotropic equations is, however, a very difficult task. We tackle this problem in section 5.3 by introducing a new adaptive fixed point algorithm which performs very well in homogeneous regions of the image domain, reducing the high frequency components of the error, and which can guarantee—up to some degree—its smoothness close to the inhomogeneous regions (edges).

Finally, MGs for *high order problems* other than those for the biharmonic equation [47] are difficult to find, even more so if the problem is anisotropic or nonlinear. High order brings numerical difficulties of which one needs to be aware. For example, in [48] an AMG was proposed for anisotropic second and fourth order equations. In [33], however, it was argued that discretizations of high order problems might not satisfy the M-matrix condition for MG convergence [47]; hence the authors suggested using geometric MG algorithms instead of an AMG for these problems. It [33], the authors also reported having observed that an inaccurate approximation of high order derivatives at coarse levels can produce such poor representations of positive definite operators that they are no longer positive definite on coarser grids, causing the MG algorithm to fail to converge.

From the above discussion, we see that there are many difficulties to overcome when developing MG algorithms for *nonlinear high order anisotropic problems*; the curvature-based denoising model falls into this class. Of primary importance is to guarantee the smoothness of the error. As we shall show, standard smoothers do not work for (2.2); hence we will focus our efforts on developing a good smoother for this problem and will test it within the standard framework of a nonlinear MG algorithm.

5.2. The MG algorithm. In this section, we introduce a nonlinear MG algorithm for the fast solution of the high order denoising formulation (2.2). To this end, we denote the nonlinear operator equation by

$$(Nu)_{i,j} \equiv \alpha\nabla\cdot\left(D_1(u)_{i,j}(\nabla\Phi')_{i,j} - D_2(u)_{i,j}(\nabla u)_{i,j}\right) + u_{i,j} = z_{i,j} \tag{5.1}$$

and construct a hierarchy of discretizations by approximating the operator $(Nu)_{i,j}$ at different grid levels. As is common practice, we denote by $N_h u_h = z_h$ the discrete equation defined on the finest grid Ω_h of size h, to be denoted by Ω_{h_L}, and similarly by $N_{2h}u_{2h} = z_{2h}$ the same on the coarser grid Ω_{2h} which is obtained by standard coarsening, to be denoted by $\Omega_{h_{L-1}}$. We can continue applying this process until we generate a sequence of L coarse levels $\Omega_{h_L}, \Omega_{h_{L-1}}, \ldots, \Omega_{h_0}$ with $h_\ell = 2^{L-\ell}h$. We state our MG method in Algorithm 1.

Algorithm 1 (nonlinear multigrid method).
Select an initial guess u_h on the finest grid h.
Set $k = 0$ and $err = tol + 1$.

While $err < tol$

$u_h^{k+1} \leftarrow \text{FAS}\,(u_h^k, N_h^k, z_h, \nu_0, \nu_1, \nu_2, \zeta, \alpha, \gamma)$

$err = \|E(u_h^k) - E(u_h^{k-1})\|_2, \quad k = k+1$

End

Here FAS denotes the cycle of going through all fine grids (smoothing the iterates and passing on the residual information to the next grid) to the coarsest grid, solving the equation on the coarsest grid accurately, and coming through all coarse grids (interpolating to the next grid and smoothing the iterates again) back to the finest grid, as shown below [8, 47, 21].

Algorithm 2 (FAS cycle). $u_h \leftarrow \text{FAS}\,(u_h, N_h, z_h, \nu_0, \nu_1, \nu_2, \zeta, \alpha, \gamma)$.

1. If Ω_h = coarsest grid, solve $N_h u_h = z_h$ accurately (i.e., ν_0 iterations by the SFP method) and return. Else continue with step 2.
2. Presmoothing: Do ν_1 steps of $u_h \leftarrow \text{SFP}\,(u_h, z_h, \zeta, \alpha, \gamma, \nu_1)$.
3. Restrict to the coarse grid, $u_{2h} \leftarrow R_h^{2h} u_h$.
4. Set the initial solution for the next level, $\bar{u}_{2h} \leftarrow u_{2h}$.
5. Compute the new right-hand side $z_{2h} \leftarrow R_h^{2h}(z_h - N_h u_h) + N_{2h} u_{2h}$.
6. Implement $u_{2h} \leftarrow \text{FAS}_{2h}(u_{2h}, N_{2h}, z_{2h}, \nu_0, \nu_1, \nu_2, \zeta, \alpha, \gamma)$.
7. Add the residual correction, $u_h \leftarrow u_h + I_{2h}^h(u_{2h} - \bar{u}_{2h})$.
8. Postsmoothing: Do ν_2 steps of $u_h \leftarrow \text{SFP}(u_h, z_h, \zeta, \alpha, \gamma, \nu_2)$.

In Algorithm 2 we represented by SFP the smoother operator. We will explain its construction in section 5.3, analyze its properties through local Fourier analysis (LFA) in section 5.4, and refine it by making it adaptive in section 5.5 and presenting its final form in Algorithm 6. For the restriction and interpolation operators R_h^{2h} and I_{2h}^h, respectively, full weighting (FW) and bilinear interpolation operators for cell-centered grids are used; see [11, 47] for details.

We remark again that Algorithm 1 needs a new smoother for it to converge since all simple and known smoothers do not lead to convergence.

5.3. The smoother—A new SFP method. Fixed point methods are usually fast algorithms, hence the wish to develop one for our curvature-based model (3.1). Unfortunately standard ways of implementing this kind of algorithm simply do not work. For instance, following [51, 50, 42], a possible scheme would be

$$-\alpha\nabla\cdot\left((D_2(u))_{i,j}^{k+1}(\nabla u)_{i,j}^{k+1}\right) + u_{i,j}^{k+1} = -\alpha\nabla\cdot\left((D_1(u))_{i,j}^{k}(\nabla\Phi')_{i,j}^{k}\right) + z_{i,j}. \tag{5.2}$$

This scheme is, however, neither stable nor convergent, the reason being that D_2 can easily change its sign so neither positive definiteness nor diagonal dominance can be guaranteed for schemes of the form $A(u^k)u^{k+1} = f(u^k, z)$. In [11], the same sort of problem was observed in trying to develop a fixed point method for a fourth order PDE with structure similar to that of (3.1), already giving an indication that standard fixed point methods for this type of equation do not converge!

In this section, we discuss how to develop a working fixed point algorithm for (3.1) that not only is much faster than the explicit time-marching methods reviewed in section 3, but also has proved to be always convergent as a stand-alone algorithm in all of the simulations we have carried out. Our aim, however, goes further and is oriented to developing an even faster

MG algorithm; with this in mind we will not use our fixed point algorithm as a stand-alone method but as a smoother for such a nonlinear MG algorithm.

To develop a working fixed point algorithm for (3.1), we will first analyze unconditionally stable time-marching schemes based on convexity-splitting ideas developed in [27, 28] (the idea was also adapted for a different imaging problem by Bertozzi, Esedoḡlu, and Gillet [6]). The resulting semi-implicit scheme (5.3) improves the stability of time-marching schemes in such a way that stability of (5.3) is guaranteed for all possible time steps. To put it simply, we solve

$$\frac{u_{i,j}^{k+1}-u_{i,j}^{k}}{\Delta t}=r(u)_{i,j}^{k}+\gamma\mathcal{N}_{i,j}^{\,k}-\gamma\mathcal{N}_{i,j}^{\,k+1}, \tag{5.3}$$

where $r(u)$ is as in (3.3), $\gamma>0$ is an appropriate constant whose value depends on the selection of $\mathcal{N}$ and needs to be sufficiently large to bring the required stability to the new algorithm, and $\mathcal{N}=\mathcal{N}(u)$ is the differential operator arising from the minimization of a convex functional such as $\int_\Omega|\nabla u|$ or $\int_\Omega|\nabla u|^2$. If $r(u)$ can be split into two parts (convex and nonconvex), then the convex part is treated implicitly and the nonconvex part explicitly. For more insight into convexity-splitting schemes and their implementation, we refer the reader to [6, 52, 44, 11, 27, 28].

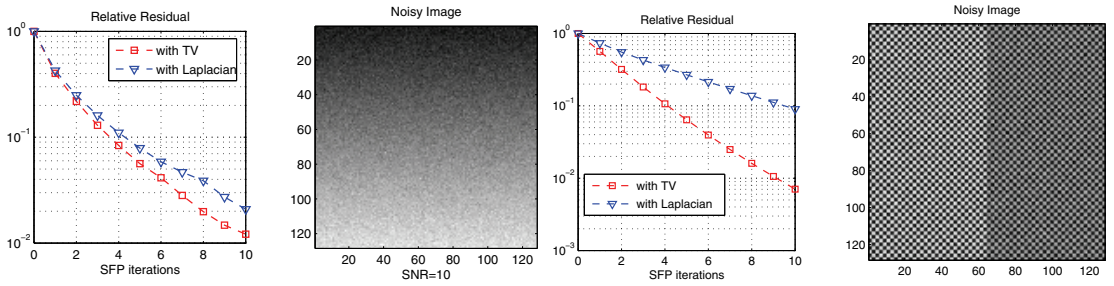
Based on the convexity-splitting scheme (5.3), we will refine the fixed point method (5.2) to make it stable. To start we introduce the stabilizing terms $\gamma\mathcal{N}^{\,k+1}$ and $\gamma\mathcal{N}^{\,k}$ and add them to both sides of (5.2), respectively. Thus, our new proposed SFP algorithm for a general $\mathcal{N}$ takes the form

$$-\gamma\mathcal{N}_{i,j}^{\,k+1}-\alpha\nabla\cdot\left((D_2(u))_{i,j}^{k+1}(\nabla u)_{i,j}^{k+1}\right)+u_{i,j}^{k+1}=-\gamma\mathcal{N}_{i,j}^{\,k}-\alpha\nabla\cdot\left((D_1(u))_{i,j}^{k}(\nabla\Phi')_{i,j}^{k}\right)+z_{i,j}. \tag{5.4}$$

Now we address the selection of $\mathcal{N}$ since it plays an important role in the performance of this new SFP scheme. We consider three possible options:

$$\mathcal{N}=\begin{cases} u, \\ \triangle u, \\ \mathcal{TV}(u)=\nabla\cdot\frac{\nabla u}{|\nabla u|}. \end{cases} \tag{5.5}$$

In our experiments the first option, $\mathcal{N}=u$, delivered very poor performance, because a very large γ needs to be selected, resulting in very slow convergence. The second option, $\mathcal{N}=\triangle u$, has been preferred by some authors for time-evolution schemes such as (5.3) in other contexts; see, for instance, [6, 44]. The third option, $\mathcal{N}=\mathcal{TV}(u)$, is our recommended choice, as we illustrate the advantages of this selection using the two examples in Figure 1. First, Figure 1(a) shows the performance for the first 10 iterations of our SFP algorithm for denoising a smoothly varying image (with no visible jumps in it); for this problem, option 2 is almost the same as option 3 in performance. However, Figure 1(b) shows that option 3 is clearly better for denoising an image with a lot of jumps (edges). We also tried other refinements of option 3 with $\mathcal{TV}_M(u)=\nabla\cdot\frac{\nabla u}{|\nabla u|^M}$ for $M=2,3$. Then we found that the resulting SFP method is more sensitive to the selection of γ.



(a) (b)

Figure 1. *Illustration of the importance of the selection of the right* $\mathcal{N}$. (a) *For smooth images, both* $\mathcal{N} = \Delta u$ *and* $\mathcal{N} = \mathcal{TV}(u)$ *work fine.* (b) *For very jumpy images (with many strong edges),* $\mathcal{N} = \mathcal{TV}(u)$ *works much better.*

Once the best $\mathcal{N}$ has been selected—in this case option 3—it is useful to rewrite (5.4) the following way:

$$-\nabla \cdot \left(C_{i,j}^{k+1}(u)(\nabla u)_{i,j}^{k+1} \right) + g_{i,j}^{k}(u) + u_{i,j}^{k+1} = z_{i,j}, \tag{5.6}$$

where the diffusion coefficient is defined as $C_{i,j}^{k+1} \equiv \gamma(D_1(u))_{i,j}^{k+1} - \alpha(D_2(u))_{i,j}^{k+1}$ and the nonlinear term $g_{i,j}^{k}(u) \equiv \nabla \cdot \left(\gamma \frac{(\nabla u)_{i,j}^{k}}{|\nabla u|_{i,j}^{k}} + \alpha \frac{(\nabla \Phi')_{i,j}^{k}}{|\nabla u|_{i,j}^{k}} \right)$. To solve (5.6) we considered the following two different methods:

1. The first is a nonlinear Gauss–Seidel–Picard (GSP) method. That is, we represent the system of algebraic equations (5.6) as $N_\ell(u_{1,1}, \ldots, u_{m,n}) = 0$ with $\ell = 1, \ldots, nm$ and unknowns $u_{1,1}, \ldots, u_{m,n}$ and apply nonlinear iterations. For instance, a nonlinear Gauss–Seidel iteration to solve the unknown at grid point (i, j) from the ℓth equation reads as

$$N_\ell(u_{1,1}^{k+1}, \ldots, u_{i-1,j}^{k+1}, u_{i,j}^{k+1}, u_{i+1,j}^{k}, \ldots, u_{n,m}^{k}) = 0, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n. \tag{5.7}$$

Then, at every kth step and (i, j) grid point, a nonlinear equation needs to be solved. To do this, we apply a number of local iterations over a linearized system using Picard's method; i.e., to compute the value of $u_{i,j}^{q+1}$ (where q is the superscript for the local iterations), the nonlinear terms $C_{\cdot,\cdot}$ and $g_{i,j}$ are evaluated using values of u from the previous qth step, leaving an easy-to-solve linear system in $u_{i,j}^{q+1}$ (see Algorithm 3).

Algorithm 3 (SFP1). $u_h \leftarrow \mathrm{SFP1}\,(u_h, z_h, \nu, \zeta, \alpha, \gamma)$.

For $k = 1$ to ν,

For $i = 1$ to m; For $j = 1$ to n,

For $q = 1$ to ζ,

$$u_{i,j}^{q+1} = \frac{u_{i+1,j}^{k} C_{i+\frac{1}{2},j}^{q} + u_{i-1,j}^{k+1} C_{i-\frac{1}{2},j}^{q} + u_{i,j+1}^{k} C_{i,j+\frac{1}{2}}^{q} + u_{i,j-1}^{k+1} C_{i,j-\frac{1}{2}}^{q} + z_{i,j} - g_{i,j}^{q}}{1 + C_{i+\frac{1}{2},j}^{q} + C_{i-\frac{1}{2},j}^{q} + C_{i,j+\frac{1}{2}}^{q} + C_{i,j-\frac{1}{2}}^{q}} \tag{5.8}$$

End

Set $u_{i,j}^{k+1} = u_{i,j}^{\zeta+1}$
End; End
End

We will name this SFP algorithm solved by GSP the SFP1 method. We remark that for (5.8), due to the costly updating of the nonlinear terms, the total cost of this method will be very large if more than one inner iteration is selected, i.e., if $q > 1$.

2. The second method is a *global* linearization of the fixed point method (5.6) by freezing all $C_{\cdot,\cdot}$'s and $g_{i,j}$ at the kth step to obtain

$$u_{i,j}^{k+1}\mathbb{S}_{i,j}^{k} - u_{i+1,j}^{k+1}C_{i+\frac{1}{2},j}^{k} - u_{i-1,j}^{k+1}C_{i-\frac{1}{2},j}^{k} - u_{i,j+1}^{k+1}C_{i,j+\frac{1}{2}}^{k} - u_{i,j-1}^{k+1}C_{i,j-\frac{1}{2}}^{k} = f_{i,j}^{k}, \tag{5.9}$$

where $C_{i+\frac{1}{2},j}^{k} = \gamma(D_1(u))_{i+\frac{1}{2},j}^{k} + \alpha(D_2(u))_{i+\frac{1}{2},j}^{k}$ and the other coefficients are computed in a similar way, $\mathbb{S}_{i,j}^{k} = 1 + C_{i+\frac{1}{2},j}^{k} + C_{i-\frac{1}{2},j}^{k} + C_{i,j+\frac{1}{2}}^{k} + C_{i,j-\frac{1}{2}}^{k}$, and $f_{i,j}^{k} = z_{i,j} - g_{i,j}^{k}$. Denote the resulting system by $A(u^k)u^{k+1} = f_{i,j}^{k}$, and note that A is positive definite and diagonally dominant. If SFP is intended to be used as a stand-alone algorithm, then at each kth iteration, the system can be solved using, for instance, PCG or linear MG. However, in the context of a nonlinear MG with SFP used as smoother, we found that partially solving the system with a few Gauss–Seidel or SOR iterations works better; we name this SFP algorithm, solved up to some accuracy by any of the above linear solvers, the SFP2 method. Algorithm 4 describes the particular case when lexicographic Gauss–Seidel (GSLEX) is used as the inner solver.

Algorithm 4 (SFP2). $u_h \leftarrow \mathrm{SFP2}\,(u_h, z_h, \nu, \zeta, \alpha, \gamma)$.
For $k = 1$ to ν,
Compute $C_{i+\frac{1}{2},j}^{k}, C_{i-\frac{1}{2},j}^{k}, C_{i,j+\frac{1}{2}}^{k}, C_{i,j-\frac{1}{2}}^{k}$
For $q = 1$ to ζ,
For $i = 1$ to m; For $j = 1$ to n,

$$u_{i,j}^{q+1} = \frac{u_{i+1,j}^{q}C_{i+\frac{1}{2},j}^{k} + u_{i-1,j}^{q+1}C_{i-\frac{1}{2},j}^{k} + u_{i,j+1}^{q}C_{i,j+\frac{1}{2}}^{k} + u_{i,j-1}^{q+1}C_{i,j-\frac{1}{2}}^{k} + f_{i,j}^{k}}{1 + C_{i+\frac{1}{2},j}^{k} + C_{i-\frac{1}{2},j}^{k} + C_{i,j+\frac{1}{2}}^{k} + C_{i,j-\frac{1}{2}}^{k}} \tag{5.10}$$

End; End.
End
End

5.4. Local Fourier analysis (LFA). As stated in [47], LFA is a very useful tool for the quantitative analysis of MG methods. Theoretically LFA is designed to study linear problems with constant coefficients on an infinite grid. Regardless of this strong limitation, LFA is still a recommended tool [8, 34, 1, 53] for the analysis of discrete nonlinear operators like (5.1). To this end, the first step is to neglect boundary conditions and to extend the discrete operator to an infinite grid; the second step assumes that any discrete nonlinear operator can be linearized locally (by freezing coefficients) and can be replaced locally by an operator with constant coefficients [47]. This method has been successfully applied to obtain a better understanding of MG algorithms applied to nonlinear problems or linear problems with discontinuous coefficients [1, 34, 15, 10, 4].

LFA for the SFP1 method. The local iterations of the SFP1 method allow the stencil representation

$$\frac{1}{h^2}\begin{bmatrix} 0 & 0 & 0 \\ -C^q_{i-\frac{1}{2},j} & \mathbb{S}^q_{i,j} & 0 \\ 0 & -C^q_{i,j+\frac{1}{2}} & 0 \end{bmatrix} u^{q+1} = z_{i,j} - g^q_{i,j} - \frac{1}{h^2}\begin{bmatrix} 0 & -C^q_{i,j-\frac{1}{2}} & 0 \\ 0 & 0 & -C^q_{i+\frac{1}{2},j} \\ 0 & 0 & 0 \end{bmatrix} u^q, \tag{5.11}$$

with $C^q_{\cdot,\cdot}, \mathbb{S}_{i,j}$, and $g_{i,j}$ defined as before. To apply LFA to (5.11) we start by defining $\bar{u}_{i,j}$ as the true solution of (5.6) at the grid point (i,j) and the error functions $\xi^{q+1}_{i,j}$ and $\xi^q_{i,j}$, as is common practice, by $\xi^{q+1}_{i,j} = \bar{u}_{i,j} - u^{q+1}_{i,j}$ and $\xi^q_{i,j} = \bar{u}_{i,j} - u^q_{i,j}$. Then, we expand them in Fourier components as

$$\xi^{q+1}_{i,j} = \sum_{\phi_1,\phi_2=-m/2}^{m/2} \psi^{q+1}_{\phi_1,\phi_2} e^{\boldsymbol{i}\theta_1 x/h} e^{\boldsymbol{i}\theta_2 y/h} \quad \text{and} \quad \xi^q_{i,j} = \sum_{\phi_1,\phi_2=-m/2}^{m/2} \psi^q_{\phi_1,\phi_2} e^{\boldsymbol{i}\theta_1 x/h} e^{\boldsymbol{i}\theta_2 y/h}, \tag{5.12}$$

where $\boldsymbol{\theta} = (\theta_1, \theta_2) \in \Theta = (-\pi, \pi]^2$, $\theta_1 = 2\pi\phi_1/m$, $\theta_2 = 2\pi\phi_2/m$, and $\boldsymbol{i} = \sqrt{-1}$. The next step is to approximate $g^q_{i,j}$ using a first order Taylor expansion as follows:

$$g_{i,j}(u^q) \approx g_{i,j}(\bar{u}) + c_{i,j}(u^q_{i,j} - \bar{u}_{i,j}) = g_{i,j}(\bar{u}) - c_{i,j}\xi^q_{i,j} \tag{5.13}$$

with $c_{i,j} \equiv \frac{\partial g}{\partial u}(\bar{u}_{i,j})$, which is reasonable when $u^q_{i,j}$ is sufficiently close to $\bar{u}_{i,j}$, i.e., when the algorithm has almost converged to the true solution.

Now we face the problem of approximating the $C^q_{\cdot,\cdot}$ coefficients. Here using a Taylor expansion does not help much, so we decided to freeze the coefficients (make them constant) and look at the predicted rates for the worst possible scenario, i.e., when the coefficients are very anisotropic. We also note that, as remarked before, the present analysis is valid when the algorithm has almost reached convergence. At this stage, the change in the values of the coefficients is almost null (they remain almost constant), and the edges in the image have reached their sharpest form; i.e., the coefficients are very anisotropic in these regions. These two observations back up our decision to use constant coefficients in the LFA analysis.

Based on these assumptions, we can substitute (5.12)–(5.13) into (5.8) to obtain the error equation

$$-\mathbb{S}^k_{i,j}\xi^{q+1}_{i,j} + C^k_{i+\frac{1}{2},j}\,\xi^q_{i+1,j} + C^k_{i-\frac{1}{2},j}\,\xi^{q+1}_{i-1,j} + C^k_{i,j+\frac{1}{2}}\,\xi^q_{i,j+1} + C^k_{i,j-\frac{1}{2}}\,\xi^{q+1}_{i,j-1} + c_{i,j}\xi^q_{i,j} = 0, \tag{5.14}$$

where the superscript k on the coefficients is just to indicate that they were computed (frozen) at the kth outer iteration. Then, the local amplification factor is given by

$$\tilde{S}_h(\boldsymbol{\theta})_{i,j}{}^{(\mathrm{SFP1})} = \frac{\left| C^k_{i+\frac{1}{2},j}\, e^{\boldsymbol{i}\theta_1} + C^k_{i,j+\frac{1}{2}}\, e^{\boldsymbol{i}\theta_2} + c_{i,j}h^2 \right|}{\left| \mathbb{S}^k_{i,j} - C^k_{i-\frac{1}{2},j}\, e^{-\boldsymbol{i}\theta_1} - C^k_{i,j-\frac{1}{2}}\, e^{-\boldsymbol{i}\theta_2} \right|}, \tag{5.15}$$

and the smoothing factor at each (i,j) grid point of the image is given as $\mu_{i,j}{}^{(\mathrm{SFP1})} = \sup\{|\tilde{S}_h(\boldsymbol{\theta})_{i,j}{}^{(\mathrm{SFP1})}| : \boldsymbol{\theta} \in \Theta^{high} = [-\pi,\pi)^2 \backslash [-\frac{\pi}{2},\frac{\pi}{2})^2\}$.

A close examination of (5.15) reveals that $\mu_{i,j}$ will increase in inhomogeneous regions of the image due to the values of the $C_{\cdot,\cdot}$'s there but, more importantly, that the SFP1 scheme might not be a good smoother if $c_{i,j}h^2$ is large enough, a situation which is likely to happen in an MG algorithm at the coarse levels. Fortunately, a number of experiments showed that, provided enough noise has been removed, the value of $g_{i,j}(u)$ barely changes across iterations, indicating that $c_{i,j}$ is also very small and suggesting the need for a good initial guess for this method. This, however, does not represent much of a problem since a simple convolution step or a full MG is enough to provide the required good initial guess.

Thus we see that $c_{i,j}h^2$ has a negligible effect on the smoothing properties of the SFP1 method if the above condition is satisfied. In contrast, the coefficients $C_{\cdot,\cdot}$ as well as parameters α, β, γ will affect the performance of this smoother.

LFA for the SFP2 method. From (5.8)–(5.10) is not difficult to see the strong similarity between SFP1 and SFP2. Other than the continuous updating of $g^q_{i,j}$ in SFP1, they differ only when the coefficients are updated. Thus we can represent the local (inner) iterations of SFP2 as

$$\frac{1}{h^2}\begin{bmatrix} 0 & 0 & 0 \\ -C^k_{i-\frac{1}{2},j} & \mathbb{S}^k_{i,j} & 0 \\ 0 & -C^k_{i,j+\frac{1}{2}} & 0 \end{bmatrix} u^{q+1} = f^k_{i,j} - \frac{1}{h^2}\begin{bmatrix} 0 & -C^k_{i,j-\frac{1}{2}} & 0 \\ 0 & 0 & -C^k_{i+\frac{1}{2},j} \\ 0 & 0 & 0 \end{bmatrix} u^q. \tag{5.16}$$

Hence using the same definitions for $\xi^{q+1}_{i,j}$ and $\xi^q_{i,j}$ as before, the SFP2 error equation is given by

$$-\mathbb{S}^k_{i,j}\xi^{q+1}_{i,j} + C^k_{i+\frac{1}{2},j}\,\xi^q_{i+1,j} + C^k_{i-\frac{1}{2},j}\,\xi^{q+1}_{i-1,j} + C^k_{i,j+\frac{1}{2}}\,\xi^q_{i,j+1} + C^k_{i,j-\frac{1}{2}}\,\xi^{q+1}_{i,j-1} = 0, \tag{5.17}$$

and the SFP2 local amplification factor by

$$\tilde{S}_h(\boldsymbol{\theta})_{i,j}{}^{(\mathrm{SFP2})} = \frac{\left|C^k_{i+\frac{1}{2},j}\,e^{\boldsymbol{i}\theta_1} + C^k_{i,j+\frac{1}{2}}\,e^{\boldsymbol{i}\theta_2}\right|}{\left|\mathbb{S}^k_{i,j} - C^k_{i-\frac{1}{2},j}\,e^{-\boldsymbol{i}\theta_1} - C^k_{i,j-\frac{1}{2}}\,e^{-\boldsymbol{i}\theta_2}\right|}, \tag{5.18}$$

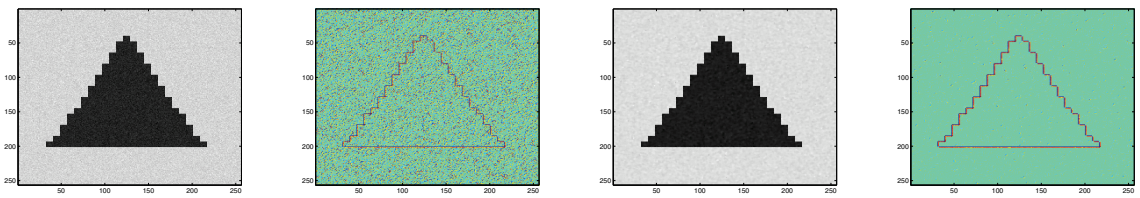
with $\mu_{i,j}{}^{(\mathrm{SFP2})} = \sup\{|\tilde{S}_h(\boldsymbol{\theta})_{i,j}{}^{(\mathrm{SFP2})}| : \boldsymbol{\theta} \in \Theta^{high}\}$.

LFA experiments for both methods. Numerical simulations carried out over different images with fixed α, β and different noise levels yielded results very much related to noise levels. In Table 1, we present the worst values across Ω_h, i.e., $\bar{\mu} = \sup\{\mu_{i,j} : (i,j) \in \Omega_h\}$ for two different noise levels in the problem of Figure 3(a).

From analyzing Table 1, some interesting conclusions may be established. For instance, $\bar{\mu}$ is smaller for less noisy images, but, more importantly, the values of $\mu_{i,j}$ are unevenly distributed within Ω. That is, by defining Ω_E as the set embracing all regions with sharp gradients and similarly $\Omega_S = \Omega \setminus \Omega_E$ as the set for smooth regions, we see from Table 1 that using the same number of relaxation steps for Ω_S and Ω_E may not be the best strategy since $\bar{\mu}$ is very different in both regions. In other words, Ω_E demands more relaxation steps, although in most images it usually represents only a small portion of Ω; see Figure 2. Hence, we can use LFA to guide us in constructing a more effective smoother than SFP1 and SFP2.

Table 1
The values of $\bar{\mu}$ for Ω_E and Ω_S for two different levels of noise.

Noise level	Image status	SFP1		SFP2	
		Ω_E	Ω_S	Ω_E	Ω_S
$SNR = 25$	Noisy	0.7431	0.7120	0.7166	0.6953
	Noise removed	0.6245	0.4879	0.6176	0.4765
$SNR = 3.5$	Noisy	0.9214	0.8821	0.8996	0.8589
	Noise removed	0.7818	0.5309	0.7779	0.5243



(a) (b) (c) (d)

Figure 2. *Example of the behavior of the smoothing amplification factor $\mu_{i,j}$.* (a) *Noisy image.* (b) *Smoothing rates computed for* (a) *presented as an image. Large values of $\mu_{i,j}$ (red points) are everywhere.* (c) *Partially denoised image after some iterations of the* SFP1 *method.* (d) *Computed values of $\mu_{i,j}$ for* (c). *Clearly the larger values of $\mu_{i,j}$, around* 0.625 *for this example, group along edges.*

5.5. An adaptive smoother (A-SFP). Here we propose an improved smoother which has an adaptive number of local smoothing steps depending on the structure of the image. This A-SFP smoother will be based on SFP2 since it is less costly than and has smoothing properties similar to those of SFP1. The LFA from section 5.4 shows that if we use the same number of smoothing steps for Ω_E and Ω_S, there is the risk of the residual not being smooth enough in Ω_E. This was confirmed in our experiments, where we noticed that the intergrid transfer operators were not working properly very close to the edges due to this fact.

To solve this problem two methods are usually considered: (1) Construct adaptive high order intergrid transfer operators as in [1]; (2) apply extra smoothing steps only in Ω_E as in [5, 9] in an adaptive way.

We selected the second option, the reason being that, on the one hand, our method needs only a little modification and the overall increment on the computational cost of the MG algorithm is very small, and, on the other hand, constructing a successful adaptive interpolation operator is not an easy task and many different ways to do it need to be tested (see [1, 24] for comments on this respect). Further, these methods can be computationally costly, and their storage requirements are high [24].

To select the regions where extra relaxation will be applied, we propose approximating Ω_E on each Ω_{h_ℓ} by an index set Q_{h_ℓ} whose entries point to pixels requiring additional smoothing steps. An easy way to construct such an index set for grid Ω_{h_ℓ} is shown in Algorithm 5. Other more accurate methods using edge-detection techniques may be used as well. Basically this algorithm is used in the first leg of the very first FAS cycle (after step 2 in Algorithm 2). The scalar entry δ, in Algorithm 5, indicates the percentage of the domain to be overrelaxed, i.e.,

edges. Since edges usually represent a small portion of the whole image, we suggest using $\delta < 0.2$.

Algorithm 5 (computation of the index set Q_{h_ℓ} for Ω_E on grid Ω_{h_ℓ}).

1. Set $[m\ n] = size(u_{h_\ell})$, and let the number of required pixels s be the integer part of δmn.
2. For $i = 1$ to m, For $j = 1$ to n,
   $|\nabla u_{h_\ell}|_{i,j} = \sqrt{(u_{h_\ell i+1,j} - u_{h_\ell i-1,j})^2 + (u_{h_\ell i,j+1} - u_{h_\ell i,j-1})^2}/2h$;
   End, End.
3. Set the set Q_{h_ℓ} as the index set of the largest s entries of matrix $|\nabla u_{h_\ell}|$.

Because the size of each Q_{h_ℓ} is relatively small, the extra storage added to the FAS algorithm is practically negligible. Once the indicator vector has been constructed, we can use it to implement our adaptive smoother, as shown in Algorithm 6.

Algorithm 6 (A-SFP). $u_{h_\ell} \leftarrow$ A-SFP $(u_{h_\ell}, z_{h_\ell}, \zeta, h_\ell, \alpha, \gamma, \nu, \delta, \omega)$.

1. Obtain Q_{h_ℓ} by using Algorithm 5 with input δ and u_{h_ℓ}.
2. Apply ν smoothing steps of Algorithm 4, and name the outcome $\bar{u}_{h_\ell}$, i.e.,
   $\bar{u}_{h_\ell} \leftarrow$ SFP2 $(u_{h_\ell}, z_{h_\ell}, \nu, \zeta, \alpha, \gamma)$.
3. If Q_{h_ℓ} is nonempty, i.e., $size(Q_{h_\ell}) \neq \emptyset$,
   Using $\bar{u}_{h_\ell}$ as input to Algorithm 4, apply $\omega\nu$ relaxation steps over those (i, j) grid points from the set Q_{h_ℓ}; i.e., solve (5.10) only if $(i, j) \in Q_{h_\ell}$.
   Take the updated $\bar{u}_{h_\ell}$ as the new output u_{h_ℓ}.
   Else
   Take $u_{h_\ell} = \bar{u}_{h_\ell}$ as the output.
   End

5.6. Two-grid analysis. The two-grid analysis is a tool which helps us to understand the convergence properties of an MG algorithm. Before proceeding to use such a tool for our Algorithm 2, we will start by explaining its basic principles; a more detailed explanation can be found in [47, 53]. The notation we will use throughout this section is the following: L_h and L_{2h} will represent the linearized operator (5.9) on grids Ω_h and Ω_{2h} of size h and $2h$, respectively; an important assumption is that L_{2h}^{-1} exists. Similarly, we will represent by S_h the smoother operator, i.e., the SFP1 algorithm. In this way, the iteration operator for the $(h, 2h)$ two-grid cycle is given by

$$M_h^{2h} = S_h^{\nu_2} K_h^{2h} S_h^{\nu_1} \quad \text{with} \quad K_h^{2h} = I_h - I_{2h}^h L_{2h}^{-1} R_h^{2h} L_h. \tag{5.19}$$

It is important to note that M_h^{2h} above needs to be computed at each (i, j)-location, but in trying to keep notation simple we have not expressed this dependence explicitly. To calculate convergence factors for M_h^{2h} one needs to analyze how the operators I_{2h}^h, L_{2h}^{-1}, R_h^{2h}, and L_h act on the Fourier components $B_h(\boldsymbol{\theta}^{(0,0)}, \cdot) = e^{\boldsymbol{i}\theta_1 x/h} e^{\boldsymbol{i}\theta_2 y/h}$ with $\boldsymbol{\theta}^{(0,0)} = (\theta_1, \theta_2)$; to this end we use the fact that quadruples of $B_h(\boldsymbol{\theta}^{(0,0)}, \cdot)$ coincide in Ω_{2h} with the respective grid function $B_{2h}(2\boldsymbol{\theta}^{(0,0)}, \cdot)$. Then, for any low frequency $\boldsymbol{\theta} \in \Theta^{low} = [-\frac{\pi}{2}, \frac{\pi}{2})^2$ we consider the frequencies

$$\boldsymbol{\theta}^{(0,0)} = (\theta_1, \theta_2), \quad \boldsymbol{\theta}^{(1,1)} = (\bar{\theta}_1, \bar{\theta}_2), \quad \boldsymbol{\theta}^{(1,0)} = (\bar{\theta}_1, \theta_2), \quad \boldsymbol{\theta}^{(0,1)} = (\theta_1, \bar{\theta}_2), \tag{5.20}$$

where

$$\bar{\theta}_i = \begin{cases} \theta_i + \pi & \text{if } \theta_i < 0, \\ \theta_i - \pi & \text{if } \theta_i \geq 0. \end{cases} \tag{5.21}$$

After defining $\boldsymbol{\alpha} = (\alpha_1, \alpha_2)$, the corresponding four components $B(\boldsymbol{\theta}^{\boldsymbol{\alpha}}, \cdot)$ are called harmonics of each other, and for $\boldsymbol{\theta} = \boldsymbol{\theta}^{(0,0)} \in \Theta^{low}$ they generate the four-dimensional space of harmonics $E_h^{\boldsymbol{\theta}} = \text{span}[B(\boldsymbol{\theta}^{\boldsymbol{\alpha}}, \cdot) : \boldsymbol{\alpha} \in \{(0,0),(1,1),(1,0),(0,1)\}]$. Hence, assuming that I_{2h}^h, L_{2h}^{-1}, R_h^{2h}, and L_h can be approximated on Ω_h and Ω_{2h} and $E_h^{\boldsymbol{\theta}}$ remains invariant under S_h, the two-grid operator M_h^{2h} can be represented on $E_h^{\boldsymbol{\theta}}$ for each $\boldsymbol{\theta} \in \Theta^{low}$ by the 4×4 matrix

$$\hat{M}_h^{2h}(\boldsymbol{\theta}) = \hat{S}_h(\boldsymbol{\theta})^{\nu_2} \hat{K}_h^{2h}(\boldsymbol{\theta}) \hat{S}_h(\boldsymbol{\theta})^{\nu_1} \text{ with } \hat{K}_h^{2h}(\boldsymbol{\theta}) = \hat{I}_h - \hat{I}_{2h}^h(\boldsymbol{\theta}) \hat{L}_{2h}^{-1}(\boldsymbol{\theta}) \hat{R}_h^{2h}(\boldsymbol{\theta}) \hat{L}_h(\boldsymbol{\theta}), \tag{5.22}$$

where the hat notation, as in $\hat{I}_{2h}^h$, stands for the approximation of each matrix in $E_h^{\boldsymbol{\theta}}$, with each defined by

$$\begin{aligned}
\hat{I}_h &= \text{diag}\{1,1,1,1\} \in \mathbb{C}^{4\times 4}, \\
\hat{L}_{2h}(\boldsymbol{\theta}) &= \tilde{L}_{2h}(2\boldsymbol{\theta}^{(0,0)}) \in \mathbb{C}^{1\times 1}, \\
\hat{L}_h(\boldsymbol{\theta}) &= \text{diag}\{\tilde{L}_h(\boldsymbol{\theta}^{(0,0)}),\ \tilde{L}_h(\boldsymbol{\theta}^{(1,1)}),\ \tilde{L}_h(\boldsymbol{\theta}^{(1,0)}),\ \tilde{L}_h(\boldsymbol{\theta}^{(0,1)})\} \in \mathbb{C}^{4\times 4}, \\
\hat{R}_h^{2h}(\boldsymbol{\theta}) &= [\tilde{R}_h^{2h}(\boldsymbol{\theta}^{(0,0)})\ \ \tilde{R}_h^{2h}(\boldsymbol{\theta}^{(1,1)})\ \ \tilde{R}_h^{2h}(\boldsymbol{\theta}^{(1,0)})\ \ \tilde{R}_h^{2h}(\boldsymbol{\theta}^{(0,1)})] \in \mathbb{C}^{1\times 4}, \\
\hat{I}_{2h}^h(\boldsymbol{\theta}) &= \frac{1}{4}[\tilde{I}_{2h}^h(\boldsymbol{\theta}^{(0,0)})\ \ \tilde{I}_{2h}^h(\boldsymbol{\theta}^{(1,1)})\ \ \tilde{I}_{2h}^h(\boldsymbol{\theta}^{(1,0)})\ \ \tilde{I}_{2h}^h(\boldsymbol{\theta}^{(0,1)})]^T \in \mathbb{C}^{4\times 1},
\end{aligned} \tag{5.23}$$

respectively, where the tilde notation, as in $\tilde{L}_{2h}(2\boldsymbol{\theta}^{(0,0)})$, represents the corresponding *symbol* [47, 53] of each matrix. Based on the above, we can calculate the asymptotic convergence factor of M_h^{2h} as follows:

$$\rho_{loc}(M_h^{2h})_{i,j} = \sup\{\rho_{loc}(\hat{M}_h^{2h}) : \boldsymbol{\theta} \in \Theta^{low}, \boldsymbol{\theta} \notin \Lambda\}, \tag{5.24}$$

where $\Lambda = \{\boldsymbol{\theta} \in \Theta^{low} : \tilde{L}_h(\boldsymbol{\theta}) = 0 \text{ or } \tilde{L}_{2h}(\boldsymbol{\theta}) = 0\}$. Again we will have different values for $\rho_{loc}(M_h^{2h})_{i,j}$ depending on the (i,j)-location, so we define $\bar{\rho}_{loc}$ as the the maximum of $\rho_{loc}(M_h^{2h})$ over all (i,j) and take this value as the asymptotic convergence factor of M_h^{2h}.

In Algorithm 2 the transfer operators we use are standard full weighting and bilinear interpolation, and their symbols [47, 53] are defined by

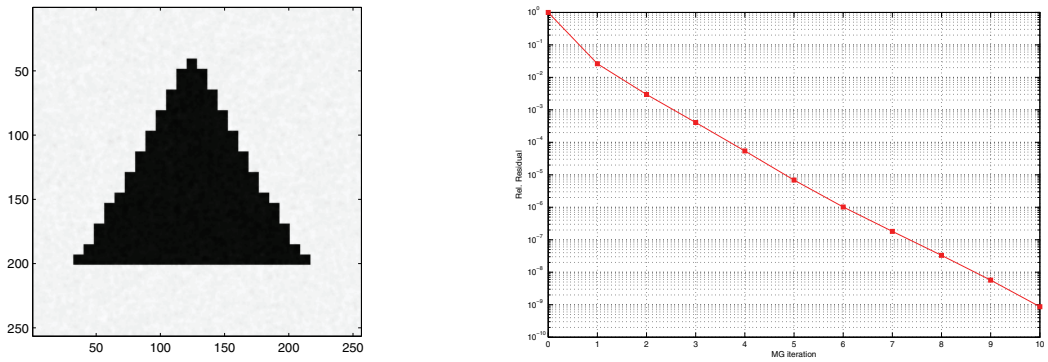
$$\tilde{R}_h^{2h}(\boldsymbol{\theta}^{\boldsymbol{\alpha}}) = \frac{1}{4}(1+\cos(\theta_1^{\boldsymbol{\alpha}}))(1+\cos(\theta_2^{\boldsymbol{\alpha}})) \quad \text{and} \quad \tilde{I}_{2h}^h(\boldsymbol{\theta}^{\boldsymbol{\alpha}}) = (1+\cos(\theta_1^{\boldsymbol{\alpha}}))(1+\cos(\theta_2^{\boldsymbol{\alpha}})).$$

The symbols for the linearized operators, on the other hand, are obtained from (5.9), and they are defined by

$$\begin{aligned}
\tilde{L}_h(\boldsymbol{\theta}^{\boldsymbol{\alpha}}) &= -\mathbb{S}_{i,j}^k + C_{i+\frac{1}{2},j}^k\, e^{\boldsymbol{i}\theta_1^{\boldsymbol{\alpha}}} + C_{i-\frac{1}{2},j}^k\, e^{-\boldsymbol{i}\theta_1^{\boldsymbol{\alpha}}} + C_{i,j+\frac{1}{2}}^k\, e^{\boldsymbol{i}\theta_2^{\boldsymbol{\alpha}}} + C_{i,j-\frac{1}{2}}^k\, e^{-\boldsymbol{i}\theta_2^{\boldsymbol{\alpha}}}, \\
\tilde{L}_{2h}(\boldsymbol{\theta}^{\boldsymbol{\alpha}}) &= -(\mathbb{S}_{i,j}^k)_{2h} + (C_{i+\frac{1}{2},j}^k)_{2h}\, e^{\boldsymbol{i}2\theta_1^{\boldsymbol{\alpha}}} + (C_{i-\frac{1}{2},j}^k)_{2h}\, e^{-\boldsymbol{i}2\theta_1^{\boldsymbol{\alpha}}} + (C_{i,j+\frac{1}{2}}^k)_{2h}\, e^{\boldsymbol{i}2\theta_2^{\boldsymbol{\alpha}}} \\
&\quad + (C_{i,j-\frac{1}{2}}^k)_{2h}\, e^{-\boldsymbol{i}2\theta_2^{\boldsymbol{\alpha}}},
\end{aligned}$$

where in the last equation $(\mathbb{S}_{i,j}^k)_{2h}$ and $(C_{\cdot,\cdot}^k)_{2h}$ are the frozen coefficients at coarse grid Ω_{2h}.

Due to the high nonlinearity of (3.1), analyzing the behavior of $\rho_{loc}(\hat{M}_h^{2h})_{i,j}$ for this equation is challenging. Consequently, we consider freezing nonlinear coefficients in (3.1) to linearize it first (hence our SFP1), and therefore $\hat{M}_h^{2h}$ is derived from a linear PDE. From (5.22),



(a) (b)

Figure 3. (a) *Denoised image.* (b) *MG iteration history with* $\nu_1 = \nu_2 = 3$ *and the GSP method as smoother.*

we see that the final value of $\rho_{loc}(\hat{M}_h^{2h})_{i,j}$ will depend on the product of three different factors: the presmoothing steps $\hat{S}_h(\boldsymbol{\theta})^{\nu_1}$, the postsmoothing steps $\hat{S}_h(\boldsymbol{\theta})^{\nu_2}$, and the coarse grid operator $\hat{K}_h^{2h}(\boldsymbol{\theta})$. In particular, the operator $\hat{K}_h^{2h}(\boldsymbol{\theta})$ will be affected by the level of noise, the value of the parameters α, β, γ, and how well the two operators $\hat{L}_h(\boldsymbol{\theta})$ and $\hat{L}_{2h}^{-1}(\boldsymbol{\theta})$ are approximated at coarse levels. It is also assumed that the residual and errors are smooth enough so the restriction and interpolation operators can do their job properly. The pre- and postsmoothing steps are also influenced by the same factors, and for them we already know that their value will increase if one of the following situations happens: noise increases, the value of either α or γ increases, or β decreases.

We are now ready to apply two-grid analysis to a real example, and to this end we select the problem of Figure 3(a) to carry out our tests. Numerical simulations using $\alpha = 1/250, \beta = 10^{-2}, \gamma = 150, 15\%$ of noise added to the image, $\nu_1 = \nu_2 = \zeta = 3$, and SFP1 as smoother yielded the value $\bar{\rho}_{loc} = 0.1289$ for this problem. This result is perfectly in accordance with the performance of our MG algorithm in solving this problem since it predicts a relative residual of $0.1289^{10} = 1.26 \times 10^{-9}$, which is congruent with the residual of 5.7×10^{-9} that we obtained experimentally after 10 FAS cycles; see Figure 3(b).

Note that the problem of Figure 3 was designed (by keeping the noise level low and few edges) to satisfy the requirements of LFA analysis, that is, moderately discontinuous coefficients. Although we do not expect this analysis to be accurate for large levels of noise, we believe we have gathered enough information from LFA to expect a good performance of our MG algorithm.

For the same problem with the same parameters as above but this time using SFP2 as smoother, $\bar{\rho}_{loc} = 0.1256$. However, in our experiment, a relative residual of 4.9×10^{-11} was obtained after the same number (10) of FAS cycles, i.e., an extra reduction of two orders of magnitude of the residual value! This experiment confirms that SFP2 has smoothing properties that are similar to or perhaps slightly better than those of SFP1.

5.7. Complexity analysis. The main cost of our MG method is the cost of the smoothing steps. Let $z \in \mathbb{R}^{m \times n}$. The cost of each outer iteration of the smoother consists of the cost

of each inner step, which is equal to 13 flops per grid point if GSLEX is being used (or 15 flops if PCG is used instead), plus the cost of the discretization, which equals 195 flops per grid point. For instance, if ζ steps of GSLEX are used, this makes the cost of every outer iteration equal to $(195+13\zeta)N$, where $N=nm$ is the total number of grid points. The other costs to be considered are *rhs*, which stands for the cost associated with the construction of the right-hand side of the residual equation at the next coarser level (steps 3, 4, and 5 of Algorithm 2), and *irc*, representing the interpolation plus residual correction procedure (step 7 of Algorithm 2). On Ω_{2h} there are 1/4 the number of grid points that there are on Ω_h, and in general if $p=2^l$, there are $p^{-2}=(1/4)^l$ as many grid points on Ω_{ph} as there are on Ω_h. Hence an upper bound on the cost of one FAS cycle is therefore

$$\begin{aligned}\lim_{L\to\infty}\left(\underbrace{(1+\delta\omega)(\nu_1+\nu_2)(195+13\zeta)N}_{\text{pre- + postsmoothing}}+\underbrace{306N}_{rhs}+\underbrace{4N}_{irc}\right)\sum_{k=0}^{L}(1/4)^k\\ =(1+\delta\omega)(\nu_1+\nu_2)(195+13\zeta)\frac{N}{0.75}+413.3N.\end{aligned}\tag{5.25}$$

Clearly our MG algorithm has the complexity of $O(N)$.

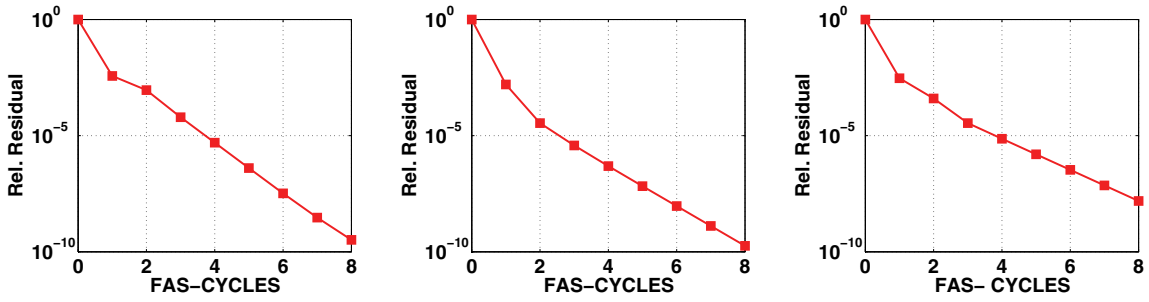
Notice that this bound is strongly dominated by the cost of *smoothing*, while the contribution from *rhs* and *irc* is relatively negligible. For instance, selecting $\delta=0.2$, $\omega=2$, $\nu_1=\nu_2=10$, and $\zeta=2$, we see that the cost of *smoothing* is 20 times that of the latter or, put precisely, the cost of $8250.6N\sum_{k=0}^{L}(1/4)^k$ versus $413.3N\sum_{k=0}^{L}(1/4)^k$.

6. Numerical results and experiments.

In this section we present some results obtained with our MG algorithm to show that its convergence properties are good, that the quality of restoration by the high order model is good as well, and that it is much faster than previous explicit methods. We also present some analysis to show how our MG algorithm is affected by changes in the values of parameters α,γ,h,β and the level of noise. In our tests, $\Phi(\kappa)=\kappa^2$ was selected since visually it gives more pleasant results. For the selection of the initial guess u_h^0, there is no restriction whatsoever; we tested with $u_h^0=z$, $u_h^0=0$, and $u_h^0=G\star z$ with G a Gaussian convolution operator. For all these options our MG algorithm converged, but convergence was slightly faster if the convolved initial guess was used; furthermore, the cost of convolution is very low.

Convergence tests. To illustrate the convergence performance of our MG algorithm, we present in Figure 4 the residual (R) and relative residual (RR) iteration results after 10 FAS cycles when solving the three test problems of Figure 10 with signal-to-noise ratio (SNR) of 3.5. Here the residual $R=\|Nu^k-z\|_2$, and the relative residual $RR=\|Nu^k-z\|_2/\|Nu^0-z\|_2$ will be tested shortly. The MG algorithm was run with the following parameters: $\alpha=1/200$, $\beta=10^{-2}$, $\gamma=100$, $\nu_1=\nu_2=10$, $\zeta=2$, $\delta=0.2$, $\omega=2$. Clearly our MG algorithm shows very fast convergence in both cases. At the coarsest level we use $\nu_0=250$ or stop when the residual is less than 1×10^{-6}; all tests in this section follow this approach.

Computational cost and speed comparison. From section 3, the only available methods for (2.2) are the explicit time-marching method and its improved version; here we hope to compare our algorithm with them.



(a) (b) (c)

Figure 4. (a) *Lena problem.* (b) *Peppers problem.* (c) *Brain problem.*

Table 2

Performance comparison of our smoothers as solvers with the corresponding MG algorithms. Clearly the MGs are faster. Here "steps" indicates outer steps, and "cost" is the overall cost in terms of WUs.

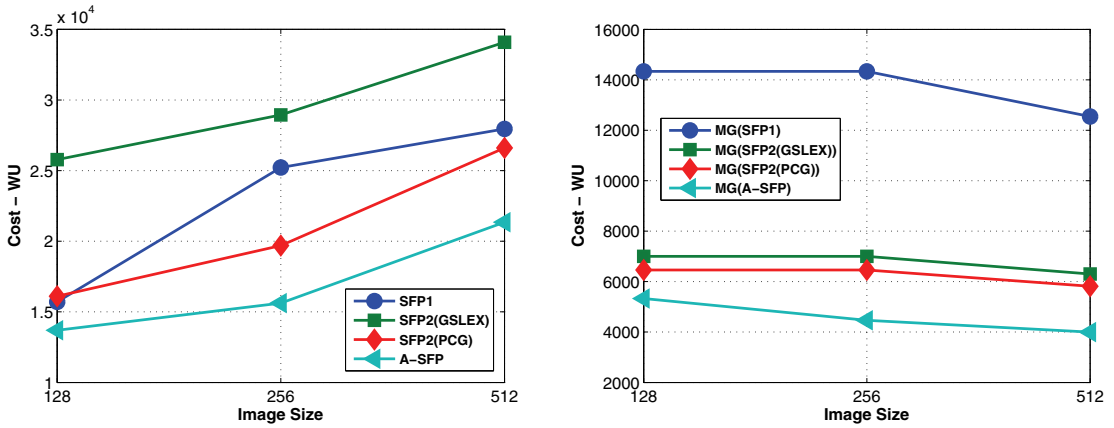
Size	Type	SFP1	MG(SFP1)	SFP2	MG(SFP2)	A-SFP	MG(A-SFP)
128^2	Steps	982	8	517	10	361	8
	Cost	25776	14335	16106	6460	13694	5530
256^2	Steps	1576	8	632	10	412	7
	Cost	25216	14335	19689	6460	15623	4464
512^2	Steps	1747	7	854	9	563	6
	Cost	27952	12543	26605	5815	21347	3998

To proceed, we introduce our work unit definition and test problem: If we define a work unit (WU) as the cost of the Gauss–Seidel updating, i.e., 1 WU = 13 flops, then the cost of discretization, which is the same for every method, is 195/13 = 15 WU, and updating with PCG is 15/13 ≈ 1.15 WU. On the other hand, our test problem here consisted of denoising the problem of Figure 10(a) with $\alpha = 1/200$, $\beta = 10^{-2}$, $\gamma = 100$, $SNR = 3.5$ until reaching a relative residual below 10^{-9}.

First, we compare to the Euler explicit time-marching method. Updating the unknowns with this method has a cost of only 5 flops; however, at every step all derivatives, coefficients, etc. need to be updated, increasing the total cost for each explicit step to $195 + 5$ flops $= 15.4$ WU. Being that the explicit method takes tens of thousands of iterations to converge, our MG algorithm is roughly two orders of magnitude less costly and faster than this method.

Second, the potentially competitive methods compared to our MG algorithms are the proposed smoothers used as solvers. Below we shall compare our smoothers SFP1, SFP2, and A-SFP when used as standalone methods with the corresponding MG(SFP1), MG(SFP2), and MG(A-SFP), which are three different MGs with the framework of Algorithm 2 but with a different selection of smoothers. We show the test results in Table 2.

There, the number of inner and outer iterations of each algorithm was optimized to deliver the best cost-effective result. In that way, SFP1 was solved using only one inner step, SFP2 using 14 PCG steps to drop the tolerance to 10^{-6}, and A-SFP with two GSLEX steps. Similarly, MG(SFP1) used 40/1 (outer/inner steps), MG(SFP2) used 10/14, and MG(A-SFP) used 10/2. In both A-SFP and MG(A-SFP), additional iterations over Ω_E with $\delta = 0.2$ and $\omega = 2$ were used.



(a) (b)

Figure 5. (a) *Reduction in the relative residual per WU for each method.* (b) *Increase in the cost per size of the image in pixels*^{2} *for each method.*

Table 3
Test results from checking stopping criteria.

FAS cycles	PSNR	Energy	Residual	RR
0	49.38	223.4	20.655	1
1	62.257	190.92	0.02419	10^{-3}
2	62.259	190.91	0.00386	10^{-4}
3	62.259	190.91	0.00074	10^{-5}
4	62.259	190.90	0.00014	10^{-6}
5	62.259	190.90	0.00002	10^{-7}

From Table 2, we can confirm that MG(A-SFP) is the most cost-efficient algorithm with its WUs 5–7 times less than the unilevel smoother related methods.

Further, in Figure 5, we show the good scaling with respect to increasing problem sizes with MG(A-SFP) (which is a usual advantage of an MG method). There we compare the cost per pixel point of MG(A-SFP) with those of the SFP1, SFP2, and A-SFP methods for image sizes of $n = 128,\ 256,\ 512$ (with n^2 pixels in each case). Clearly one observes that MG(A-SFP) is scalable with respect to WUs, while the unilevel methods are not.

Stopping criteria. We already have shown the convergence properties of our MG algorithm, and now we clarify the stopping criteria. Although R and RR defined above can be used to stop the algorithm, we found it more useful to use the energy values to execute this task. Our observations indicate that the peak SNR (PSNR) stops increasing when the change in the energy between two consecutive iterations is below 10^{-1}. Therefore, in practice, we suggest stopping our MG algorithm when $\|E(u^k) - E(u^{k-1})\|_2 < tol$ with $tol = 10^{-1}$. This is roughly equivalent to stopping the algorithm when $RR < 10^{-4}$. To help the reader understand our motives we show in Table 3 the data obtained from solving the benchmark problem with our MG algorithm using the following parameters : $\alpha = 1/200$, $\beta = 10^{-2}$, $\gamma = 100$, $\nu_1 = \nu_2 = 10$, $\zeta = 2$, $SNR = 3.5$, and $size = 256^2$.

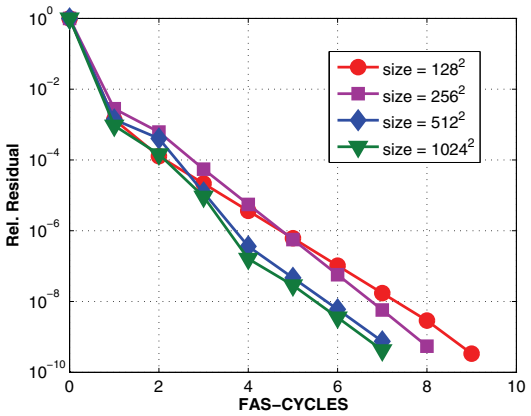


Figure 6. *Mesh size h-dependence test.*

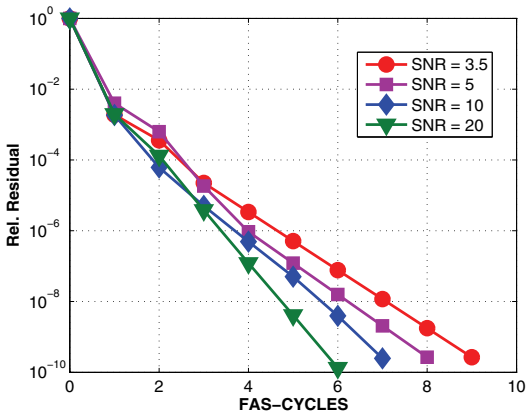


Figure 7. *SNR-dependence test.*

Computational analysis. Here we analyze how the performance of our MG algorithm is affected when the value of any of α, β, γ, $size$, SNR is changed while values of the rest are kept fixed.

h-dependence test. In Figure 6 we illustrate the h-dependence of our MG algorithm, i.e., its performance with respect to different sizes of the image. The MG algorithm was run with the parameters $\alpha = 1/200$, $\beta = 10^{-2}$, $\gamma = 100$, $\nu_1 = \nu_2 = 10$, $\zeta = 2$, $SNR = 3.5$ for all tests. Clearly the performance of our MG algorithm not only is not decimated but gets better as the size of the image increases. One possible explanation for this unusual behavior is that SFP1 and SFP2 are h-dependent due to the $C_{\cdot,\cdot}$'s depending on h. This means that when h increases, the values of the coefficients decrease, making $\mathbb{S}_{i,j}$ the dominant term, hence obtaining better smoothing at coarser levels; see Table 1. In other words, the bigger the size of the image, the more h grows at coarse levels and hence the better the efficiency is.

Noise level test. In Figure 7 the RR history against MG iterations (FAS cycles) is presented for different noise levels. The MG algorithm was run with the parameters $\alpha = 1/200$, $\beta = 10^{-2}$, $\gamma = 100$, $\nu_1 = \nu_2 = 10$, $\zeta = 2$, and $size = 256^2$ for all tests. Although convergence is slower for noisier images in general, the number of FAS cycles does not increment very much.

γ-dependence test. In Figure 8, dependence on the stabilization parameter γ is presented. The MG algorithm was run with the fixed parameters $\alpha = 1/200$, $\beta = 10^{-2}$, $\nu_1 = \nu_2 = 10$, $\zeta = 2$, $SNR = 3.5$, and $size = 256^2$ for all tests, and γ was varied from 100 to 160. As can be seen, a bad selection of γ tends to reduce the performance of the MG algorithm. The right value can be selected by the L-curve approach [50] and is dependent upon the level of noise and the values of α and β.

α-dependence test. In Figure 9 we show how the MG algorithm is affected by selecting different values for the regularization parameter α. The MG algorithm was run with the fixed parameters $\beta = 10^{-2}$, $\gamma = 100$, $\nu_1 = \nu_2 = 10$, $\zeta = 2$, $SNR = 3.5$, and $size = 256^2$ for all tests, and α was varied from 0.001 to 0.1. Usually, the value of α needs to be increased as the level of noise gets higher. This has the effect of making the $D(u)$-coefficients more discontinuous, affecting, as we have already mentioned, both the approximation of the nonlinear operator on

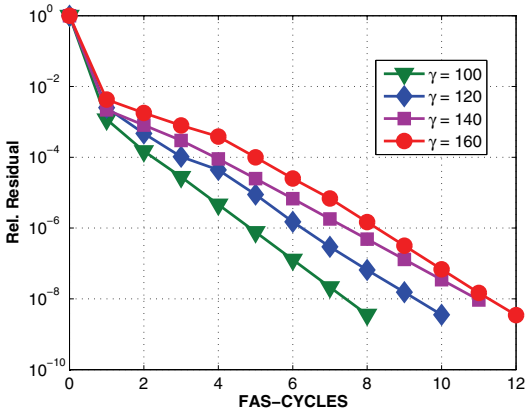


Figure 8. *γ-dependence test.*

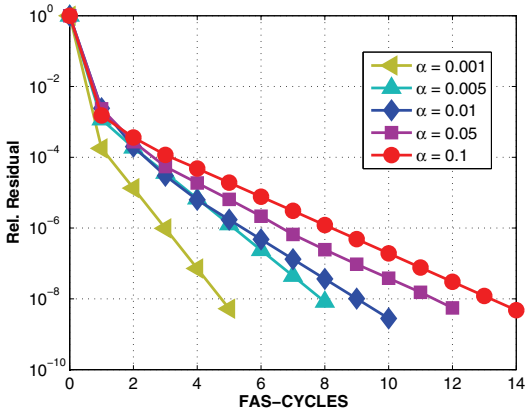


Figure 9. *α-dependence test.*

Table 4
Test results from varying α.

α	FAS cycles	PSNR
0.001	5	56.8
0.005	8	62.3
0.01	10	62.1
0.05	12	57
0.1	14	52

coarse grids and the smoothing factor. Nonetheless, the performance of our MG algorithm is still quite good, obtaining very low residuals with few MG cycles.

The results shown in the Figure 9 need to be carefully interpreted. Although they show that the performance of the MG algorithm is worse for large α, it is also true that such large values are not used in practice. The purpose of α is to select the amount of noise to be removed, so there is no point in choosing either a very large or very small α. In Table 4 we illustrate the effect that different values of α have on the PSNR and the number of FAS cycles used by our algorithm to solve the problem. The maximum PSNR was for small $\alpha = 0.005$, and only 8 FAS cycles were needed.

β-dependence test. The most critical parameter affecting the performance of our MG algorithm is definitively β. On the benchmark problem we ran the MG algorithm with the fixed parameters $\alpha = 1/200$, $\gamma = 100$, $\nu_1 = \nu_2 = 10$, $\zeta = 2$, $SNR = 3.5$, and $size = 256^2$ for all tests, while β was varied from 1 to 10^{-2}.

The results are presented in Table 5 and show a much better performance of our MG for large β, which is not surprising since β strongly influences the values of the $D(u)$-coefficients. Theoretically when κ is the curvature of level sets, β should be as small as possible; however, in practice this is not only not necessary but also not recommended from a practical point of view. This has been discussed, for instance, in [2] for the TV denoising model, where it was shown that a smaller β does not necessarily lead to a better reconstruction. For the curvature-based model a similar argument applies, and $\beta = 10^{-2}$ is a fair selection which

Table 5
Results of a fixed α and varying β.

β	α	FAS cycles	PSNR
1	1/200	3	48
10^{-1}	1/200	4	55.9
10^{-2}	1/200	10	60.5
10^{-3}	1/200	25	60.6

Table 6
Fixing β first and then optimizing α.

β	α	FAS cycles	PSNR
1	1	3	59.3
10^{-1}	1/40	5	59.9
10^{-2}	1/200	10	60.5
10^{-3}	1/200	25	60.6

provides a good balance of quality of reconstruction and our MG algorithm performance.

Table 5 shows that decreasing β from 10^{-2} to 10^{-3} for fixed α does not improve the PSNR in a meaningful way, but there is a huge difference in the number of FAS cycles used by our algorithm to solve both problems.

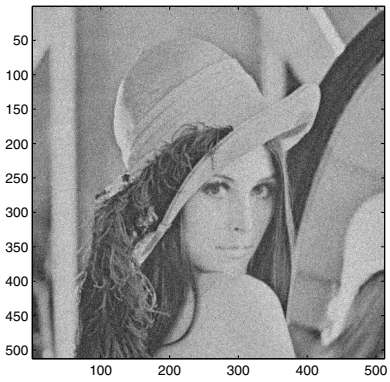
Table 6 presents the PSNR values obtained using different values of β—with α selected to deliver the best possible PSNR for each β value—and the number of FAS cycles used by our MG algorithm to converge.

Quality of restoration test. We show some qualitative results in Figure 10. The added Gaussian noise is large so that all of the images in the left column have $SNR = 3.5$. As can be appreciated, the resulting denoised images do not show the undesirable staircase effect, as expected, and noise is mostly removed.

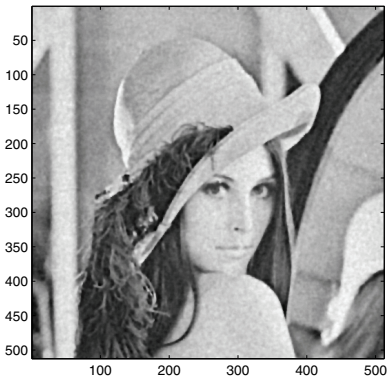
7. Generalization. Fast algorithms for solving high order PDEs are in high demand. In the image processing community researchers have realized that using high order models yields better results than using second order models. Two good examples are the curvature-based denoising model we studied here and the elastica inpainting model [16]. For the resulting PDEs from these high order models only explicit or semi-implicit time-marching methods have been reported in the literature. One of the advantages of our SFP1 and SFP2 algorithms is that they can be very easily generalized to solve other PDEs similar to (2.2). That is, we can implement fast fixed point algorithms for other models by splitting their differential operators and adding up suitable stabilizing terms. Then these fixed point algorithms can be used as smoothers in an MG context. For instance, we already have successfully applied this idea to solving the fourth order PDE of the Euler's elastica digital inpainting model; see [11] for reported results. We also have encouraging results from using this method for solving a three-dimensional denoising problem also known as surface fairing that has been studied by Elsey and Esedoḡlu [26]. For this we have implemented only the two-dimensional case or curve denoising.

8. Conclusions. In this paper, we introduced two algorithms (SFP and MG) for solving the curvature-based denoising model [56], which is high order and capable of effective noise removal. The resulting EL equation is of fourth order, anisotropic, and highly nonlinear, so conventional algorithms struggle to find the solution quickly and efficiently. In contrast, our MG algorithm is shown to be fast and robust up to some degree to changes in the noise level and parameters.

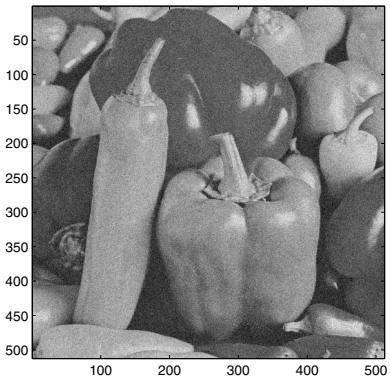
We explained that a fixed point method using the Vogel and Oman idea [51] is unstable and simply does not work for this curvature-based formulation. We then showed how to stabilize this fixed point method and developed a stabilized fixed point method, giving evidence through



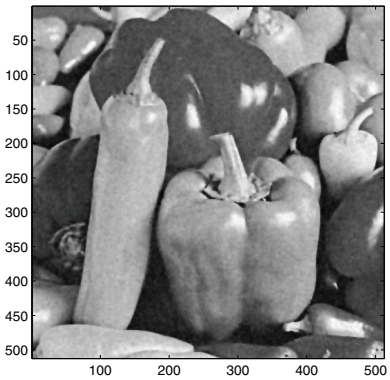
(a)



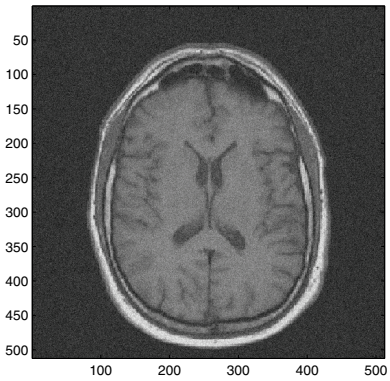
(b)



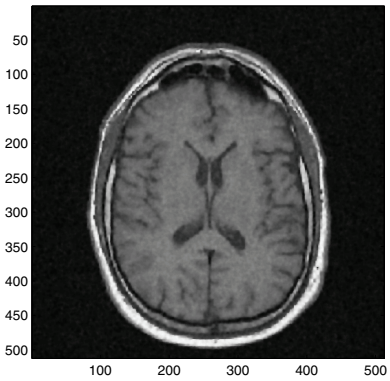
(c)



(d)



(e)



(f)

Figure 10. *Image denoising examples obtained with our MG algorithm.* (a), (c), *and* (e) *are noisy images.* (b), (d), *and* (f) *are denoised images.*

local Fourier analysis of its smoothing properties. Based on this, we developed a fast nonlinear MG method. Finally, a generalization of our algorithms to similar problems was discussed. A recent generalization to color images was done in [12].

REFERENCES

[1] R. E. Alcouffe, A. Brandt, J. E. Dendy, Jr., and J. W. Painter, *The multi-grid method for the diffusion equation with strongly discontinuous coefficients*, SIAM J. Sci. Statist. Comput., 2 (1981), pp. 430–454.

[2] 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.

[3] G. Aubert and P. Kornprobst, *Mathematical Problems in Image Processing. Partial Differential Equations and the Calculus of Variations*, Appl. Math. Sci. 147, Springer-Verlag, New York, 2002.

[4] N. Badshah and K. Chen, *Multigrid method for the Chan-Vese model in variational segmentation*, Commun. Comput. Phys., 4 (2008), pp. 294–316.

[5] D. Bai and A. Brandt, *Local mesh refinement multilevel techniques*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 109–134.

[6] A. L. Bertozzi, S. Esedoḡlu, and A. Gillet, *Inpainting of binary images using the Cahn-Hilliard equation*, IEEE Trans. Image Process., 16 (2007), pp. 285–291.

[7] J. H. Bramble and X. J. Zhang, *Uniform convergence of the multigrid V-cycle for an anisotropic problem*, Math. Comp., 70 (2001), pp. 453–470.

[8] A. Brandt, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp., 31 (1977), pp. 333–390.

[9] A. Brandt, *Multigrid Techniques:* 1984 *Guide with Applications to Fluid Dynamics*, Gesellschaft für Mathematik und Datenverarbeitung (GMD), St. Augustin, Germany, 1984.

[10] C. Brito-Loeza and K. Chen, *Multigrid method for a modified curvature driven diffusion model for image inpainting*, J. Comput. Math., 26 (2008), pp. 856–875.

[11] C. Brito-Loeza and K. Chen, *Fast numerical algorithms for Euler's elastica inpainting model*, Int. J. Mod. Math., 5 (2010), pp. 157–182.

[12] C. Brito-Loeza and K. Chen, *On high-order denoising models and fast algorithms for vector-valued images*, IEEE Trans. Image Process., 19 (2010), pp. 1518–1527.

[13] T. Chan, A. Marquina, and P. Mulet, *High-order total variation-based image restoration*, SIAM J. Sci. Comput., 22 (2000), pp. 503–516.

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

[15] T. F. Chan, K. Chen, and J. L. Carter, *Iterative methods for solving the dual formulation arising from image restoration*, Electron. Trans. Numer. Anal., 26 (2007), pp. 299–311.

[16] T. F. Chan, S. H. Kang, and J. Shen, *Euler's elastica and curvature-based inpainting*, SIAM J. Appl. Math., 63 (2002), pp. 564–592.

[17] T. F. Chan and P. Mulet, *On the convergence of the lagged diffusivity fixed point method in total variation image restoration*, SIAM J. Numer. Anal., 36 (1999), pp. 354–367.

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

[19] T. F. Chan and W. L. Wan, *Robust multigrid methods for nonsmooth coefficient elliptic linear systems*, J. Comput. Appl. Math., 123 (2000), pp. 323–352.

[20] T. F. Chan, H. M. Zhou, and R. H. Chan, *Continuation method for total variation denoising problems*, in Proceedings of the SPIE Symposium on Advanced Signal Processing Algorithms, Vol. 2563, SPIE, Bellingham, WA, 1995, pp. 314–325.

[21] K. Chen, *Matrix Preconditioning Techniques and Applications*, Cambridge Monogr. Appl. Comput. Math. 19, Cambridge University Press, Cambridge, UK, 2005.

[22] K. Chen and J. Savage, *An accelerated algebraic multigrid algorithm for total variation denoising*, BIT, 47 (2007), pp. 277–296.

[23] Y. Chen, S. Levine, and M. Rao, *Variable exponent, linear growth functionals in image restoration*, SIAM J. Appl. Math., 66 (2006), pp. 1383–1406.
[24] J. E. Dendy, Jr., *Black box multigrid*, J. Comput. Phys., 48 (1998), pp. 366–386.
[25] M. Donatelli, *A multigrid for image deblurring with Tikhonov regularization*, Numer. Linear Algebra Appl., 12 (2005), pp. 715–729.
[26] M. Elsey and S. Esedoḡlu, *Analogue of the total variation denoising model in the context of geometry processing*, Multiscale Model. Simul., 7 (2009), pp. 1549–1573.
[27] D. J. Eyre, *Unconditionally gradient stable time marching the Cahn-Hilliard equation*, in Computational and Mathematical Models of Microstructural Evolution, J. W. Bullard et al., eds., Materials Research Society, Warrendale, PA, 1998, pp. 39–46.
[28] D. J. Eyre, *An Unconditionally Stable One-Step Scheme for Gradient Systems*, manuscript, 1998; available online from http://www.math.utah.edu/~eyre/research/methods/stable.ps.
[29] R. Fischer and T. Huckle, *Multigrid solution techniques for anisotropic structured linear systems*, Appl. Numer. Math., 58 (2008), pp. 407–421.
[30] C. Frohn-Schauf, S. Henn, and K. Witsch, *Nonlinear multigrid methods for total variation image denoising*, Comput. Vis. Sci., 7 (2004), pp. 199–206.
[31] S. Henn and K. Witsch, *A multigrid approach for minimizing a nonlinear functional for digital image matching*, Computing, 64 (2000), pp. 339–348.
[32] L. Homke, *A multigrid method for anisotropic PDEs in elastic image registration*, Numer. Linear Algebra Appl., 13 (2006), pp. 215–229.
[33] S. L. Keeling and G. Haase, *Geometric multigrid for high-order regularizations of early vision problems*, Appl. Math. Comput., 184 (2007), pp. 536–556.
[34] H. Kostler, K. Ruhnau, and R. Wienands, *Multigrid solution of the optical flow system using a combined diffusion- and curvature-based regularizer*, Numer. Linear Algebra Appl., 15 (2008), pp. 201–218.
[35] M. Lysaker, A. Lundervold, and X.-C. Tai, *Noise removal using fourth-order partial differential equation with applications to medical magnetic resonance images in space and time*, IEEE Trans. Image Process., 12 (2003), pp. 1579–1590.
[36] M. Lysaker, S. Osher, and X.-C. Tai, *Noise removal using smoothed normals and surface fitting*, IEEE Trans. Image Process., 13 (2004), pp. 1345–1357.
[37] 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.
[38] G. Papandreou and P. Maragos, *Multigrid geometric active contour models*, IEEE Trans. Image Process., 16 (2007), pp. 229–240.
[39] P. Perona and J. Malik, *Scale-space and edge detection using anisotropic diffusion*, IEEE Trans. Pattern Anal. Mach. Intell., 12 (1990), pp. 629–639.
[40] A. Reusken and M. Soemers, *On the robustness of a multigrid method for anisotropic reaction-diffusion problems*, Computing, 80 (2007), pp. 299–317.
[41] L. I. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Phys. D, 60 (1992), pp. 259–268.
[42] J. Savage and K. Chen, *An improved and accelerated non-linear multigrid method for total-variation denoising*, Int. J. Comput. Math., 82 (2005), pp. 1001–1015.
[43] J. Savage and K. Chen, *On multigrids for solving a class of improved total variation based staircasing reduction models*, 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. 69–94.
[44] P. Smereka, *Semi-implicit level set methods for curvature and surface diffusion motion*, J. Sci. Comput., 19 (2003), pp. 439–456.
[45] R. S. Spitareli, R. March, and D. Arena, *A multigrid finite-difference method for the solution of Euler equations of the variational image segmentation*, Appl. Numer. Math., 39 (2001), pp. 181–189.
[46] M. Sturmer, H. Kostler, and U. Rude, *A fast full multigrid solver for applications in image processing*, Numer. Linear Algebra Appl., 15 (2008), pp. 187–200.
[47] U. Trottenberg, C. W. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, San Diego, 2001.
[48] P. Vanek, J. Mandel, and M. Brezina, *Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems*, Computing, 56 (1996), pp. 179–196.

[49] L. A. Vese and S. J. Osher, *Numerical methods for p-harmonic flows and applications to image processing*, SIAM J. Numer. Anal., 40 (2002), pp. 2085–2104.
[50] C. Vogel, *Computational Methods for Inverse Problems*, Frontiers Appl. Math. 23, SIAM, Philadelphia, 2002.
[51] C. R. Vogel and M. E. Oman, *Iterative methods for total variation denoising*, SIAM J. Sci. Comput., 17 (1996), pp. 227–238.
[52] B. P. Vollmayr-Lee and A. D. Rutenberg, *Fast and accurate coarsening simulation with an unconditionally stable time step*, Phys. Rev. E, 68 (2003), 066703.
[53] R. Wienands and W. Joppich, *Practical Fourier Analysis for Multigrid Methods*, Chapman & Hall/CRC, Boca Raton, FL, 2005.
[54] S. Wise, J. Kim, and J. Lowengrub, *Solving the regularized, strongly anisotropic Cahn–Hilliard equation by an adaptive nonlinear multigrid method*, J. Comput. Phys., 226 (2007), pp. 414–446.
[55] Y.-L. You and M. Kaveh, *Fourth-order partial differential equations for noise removal*, IEEE Trans. Image Process., 9 (2000), pp. 1723–1730.
[56] W. Zhu and T. F. Chan, *Image Denoising Using Mean Curvature*, preprint, http://www.math.nyu.edu/~wzhu/.