SIAM J. IMAGING SCIENCES
Vol. 1, No. 3, pp. 273–293

Inpainting by Flexible Haar-Wavelet Shrinkage*

R. H. Chan†, S. Setzer‡, and G. Steidl‡

Abstract. We present novel wavelet-based inpainting algorithms. Applying ideas from anisotropic regularization and diffusion, our models can better handle degraded pixels at edges. We interpret our algorithms within the framework of forward-backward splitting methods in convex analysis and prove that the conditions for ensuring their convergence are fulfilled. Numerical examples illustrate the good performance of our algorithms.

Key words. image inpainting, anisotropic methods, forward-backward algorithm, wavelet shrinkage

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

DOI. 10.1137/070711499

1. Introduction.

The problem of inpainting occurs when part of the data in an image is missing. The task of inpainting is to recover the missing regions from the observed (sometimes noisy) incomplete data. The mathematical model for the image inpainting problem reads as follows: For convenience of notation we consider two-dimensional images $\boldsymbol{u}$ defined on $\{1,\dots,n\}\times\{1,\dots,n\}$ and reshape them columnwise into a vector $u\in\mathbb{R}^N$ with $N=n^2$. Let the nonempty set $C\subset\{1,\dots,N\}$ be the given region of the observed pixels. Then the observed incomplete image f is

$$f(j)=\begin{cases} u(j)+\varepsilon(j) & \text{if } j\in C,\\ \text{arbitrary} & \text{otherwise,}\end{cases}$$

where $\varepsilon(j)$ denotes the noise. In the following, we denote by P_C the diagonal matrix with diagonal entries 1 for indices in C and 0 otherwise.

Initiated by [3], many useful techniques have been proposed to address this problem. In this paper we are mainly interested in wavelet-based inpainting methods. Such methods were, e.g., proposed in [6, 13]. However, these methods often let degraded pixels survive at sharp directed edges. A typical example is shown in Figure 1. Here both the cubic spline interpolation and the wavelet-based method from [6] produce visible artifacts, in particular at the horizontal line. This was our motivation for considering more flexible wavelet-based methods.

We focus on the following general types of inpainting algorithms.

*Received by the editors December 21, 2007; accepted for publication (in revised form) May 5, 2008; published electronically September 4, 2008.
http://www.siam.org/journals/siims/1-3/71149.html
†Department of Mathematics, The Chinese University of Hong Kong, Shatin, Hong Kong, China (rchan@math.cuhk.edu.hk). This author's research was supported in part by HKRGC grants CUHK 400405 and CUHK DAG 2060257.
‡Department of Mathematics and Computer Science, University of Mannheim, 68131 Mannheim, Germany (ssetzer@kiwi.math.uni-mannheim.de, steidl@math.uni-mannheim.de).

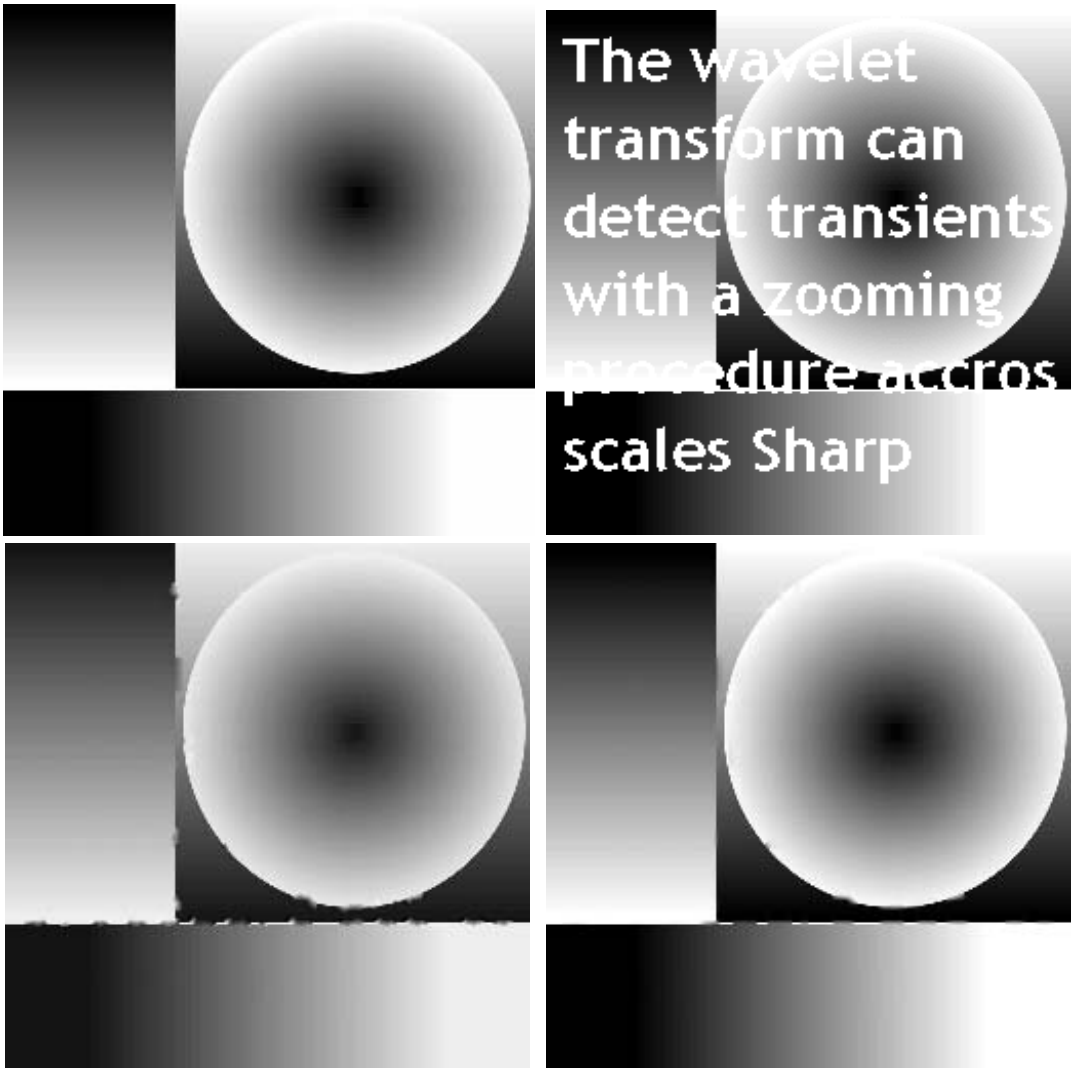

Figure 1. *Top left: original image. Top right: degraded image. Bottom left: cubic interpolation by the MATLAB routine "griddata" (peak signal-to-noise ratio (PSNR) = 29.39, $\text{err}_2 = 8.64$, $\text{err}_1 = 0.79$). Bottom right: interpolation by the algorithm in [6] with $c = 1$ and two levels ($PSNR = 33.27$, $\text{err}_2 = 5.53$, $\text{err}_1 = 0.46$). The interpolated images have artifacts, in particular at the horizontal line.*

Algorithm I (Exact data)

Initialization: u_0

For $r = 0, \ldots$ iterate until convergence

(i) Solve a restoration problem for the current image u_r to obtain $\hat{u}_{r+1}$.

(ii) Set

$$u_{r+1}(j) := \begin{cases} f(j) & \text{if } j \in C, \\ \hat{u}_{r+1}(j) & \text{otherwise.} \end{cases}$$

Output: u^*

Algorithm II (Noisy data)
Same as Algorithm I except that we have to apply step (i) to the final iterate u^* again.
Output: $u^\diamond = \hat{u}^*$

Indeed, depending on the restoration method used in step (i), many known inpainting algorithms are of this general type. In Cai, Chan, and Shen [6], the following wavelet-frame based denoising method was proposed for step (i) of Algorithm I: Let $A \in \mathbb{R}^{M,N}$, $M \geq N$, denote a frame analysis operator of a Parseval frame; i.e., any $u \in \mathbb{R}^N$ can be written as $u = A^{\mathrm{T}} d$ and $A^{\mathrm{T}} A = I$. Further, let $\Lambda := \operatorname{diag}(\lambda)$ be a diagonal matrix containing the components of the vector $\lambda := (\lambda_j)_{j=1}^M$ as diagonal entries. Then the authors suggest solving

$$d_{r+1} = \operatorname*{argmin}_{d \in \mathbb{R}^M} \left\{ \frac{1}{2} \|A u_r - d\|_2^2 + \|\Lambda\, d\|_1 \right\}.$$

Since the solution of $\operatorname{argmin}_d\{\frac{1}{2}\|c - d\|_2^2 + \|\Lambda\, d\|_1\}$ is given by $\mathcal{T}_\Lambda(c)$ with the *soft threshold operator* $\mathcal{T}_\Lambda$ defined componentwise by

$$T_{\lambda_j}(c_j) = \frac{1}{2}\left((c_j - \lambda_j) + |c_j - \lambda_j| + (c_j + \lambda_j) - |c_j + \lambda_j|\right), \qquad j = 1, \ldots, M$$

(see, e.g., [9]), the restoration step (i) becomes

$$\hat{u}_{r+1} = A^{\mathrm{T}} \mathcal{T}_\Lambda(A u_r). \tag{1}$$

They proved that for noisy input data the iterates of Algorithm II with restoration step (1) converge to $u^\diamond = A^{\mathrm{T}} \hat{d}$, where $\hat{d}$ is the solution of

$$\hat{d} = \operatorname*{argmin}_{d \in \mathbb{R}^M} \left\{ \frac{1}{2} \|P_C f - P_C A^{\mathrm{T}} d\|_2^2 + \|\Lambda\, d\|_1 + \frac{1}{2} \|(I - A A^{\mathrm{T}}) d\|_2^2 \right\}. \tag{2}$$

Indeed, this algorithm is very similar to a method proposed in [13], where Fadili and Starck solve

$$\hat{d} = \operatorname*{argmin}_{d \in \mathbb{R}^M} \left\{ \frac{1}{2} \|P_C A^{\mathrm{T}} d - P_C f\|_2^2 + \|\Lambda\, d\|_1 \right\} \tag{3}$$

by

$$d_{r+1} = \mathcal{T}_\Lambda\left(d_r + A(P_C f - P_C A^{\mathrm{T}} d_r)\right)$$

and set $u^\diamond = A^{\mathrm{T}} \hat{d}$. Obviously, for an orthogonal matrix A the wavelet-based algorithms (2) and (3) coincide. However, for various nonorthogonal frame analysis matrices A, the numerical experiments in [6] indicate that the algorithm (2) performs better.

In [11], the method (3) was generalized in order to recover both the texture and the cartoon part of an image; see also [4]. To this end, Elad et al. [11] solve

$$\operatorname*{argmin}_{d_t, d_n} \left\{ \frac{1}{2} \|P_C(A_t^{\mathrm{T}} d_t + A_n^{\mathrm{T}} d_n - f)\|_2^2 + \lambda(\|d_t\|_1 + \|d_n\|_1) + \gamma \mathrm{TV}(A_n^{\mathrm{T}} d_n) \right\},$$

where A_n denotes the discrete curvelet transform, A_t denotes the discrete cosine transform, and d_t and d_n are the texture and cartoon components, respectively.

Beyond regularization techniques, PDE-based approaches can be applied in the restoration step. In [15, 29] it was demonstrated that inpainting methods based on edge enhancing anisotropic diffusion appear to be superior to linear methods, e.g., spline interpolation methods, and nonlinear isotropic diffusion methods. Indeed, these ideas were, together with wavelet techniques, the second ingredient for our algorithms. For other PDE-based methods we refer only to [7] and the references therein.

In this paper, we focus on inpainting by combining anisotropic regularization and diffusion methods with multilevel Haar-wavelet filters. Our new methods increase the PSNR of various restored images significantly, e.g., by 3 dB for the image in Figure 1, and avoid highly visible artifacts. Following along the lines of [6], we have proved the convergence of our method by embedding it into the framework of forward-backward splitting algorithms.

This paper is organized as follows: In section 2, we briefly review anisotropic regularization and diffusion methods. Ideas from this section, in particular the application of a diffusion tensor, carry over to our wavelet setting. In section 3, we present a new anisotropic Haar-wavelet method for the inpainting problem. The convergence proof of our algorithm is given in section 4. Finally, section 5 contains numerical examples which demonstrate the excellent performance of our algorithm.

2. Anisotropic regularization and diffusion. In this section, we sketch the basic ideas from anisotropic diffusion and regularization methods that carry over to our wavelet setting. We prefer the more common continuous point of view in this section, while the rest of the paper deals with a discrete setting obtained by discretizing gradients with the help of wavelet filters. Anisotropic diffusion methods such as edge enhancing or coherence enhancing diffusion have been used for the directed denoising of images for a long time; see [27] and the references therein. Recently, anisotropic regularization methods have become popular, e.g., for the restoration of polygonal shapes [2, 12, 25] with sharp edges and corners.

Let us consider a single restoration step r of our inpainting method which computes for a given continuous image $\tilde{f} := u_r$ on a quadratic domain Ω the image $\hat{u}_{r+1}$. By $\circ$, we denote the Hadamard product (componentwise product) of matrices. From the variational point of view, one could restore the image by solving for an appropriate proper, lower semicontinuous (lsc), convex function Φ and an invertible matrix $V \in \mathbb{R}^{2,2}$ the problem

$$\operatorname*{argmin}_u \left\{ \frac{1}{2}\|\tilde{f} - u\|_{L_2}^2 + \lambda \int_\Omega \Phi((V^{\mathrm{T}}\nabla u) \circ (V^{\mathrm{T}}\nabla u))\, dx \right\}, \tag{4}$$

where the function space of u depends on the choice of Φ. For $\Phi(x^2, y^2) := \sqrt{x^2 + y^2}$ and $V := I$, the functional in (4) is the Rudin–Osher–Fatemi (ROF) functional [22], and we consider the space BV of functions of bounded variations. For $\Phi(x^2, y^2) := |x| + |y|$ and special rotation matrices V, the functional (4) was used for corner preserving denoising in [2, 25]. For $V = I$, minimization algorithms for this functional were considered, e.g., in [16]. If Φ is differentiable, then the Euler–Lagrange equation of (4) reads

$$0 = \tilde{f} - u + \lambda \nabla \cdot (D\, \nabla u) \tag{5}$$

with

$$D := V \begin{pmatrix} 2\partial_1 \Phi\big((V^{\mathrm{T}}\nabla u) \circ (V^{\mathrm{T}}\nabla u)\big) & 0 \\ 0 & 2\partial_2 \Phi\big((V^{\mathrm{T}}\nabla u) \circ (V^{\mathrm{T}}\nabla u)\big) \end{pmatrix} V^{\mathrm{T}}. \tag{6}$$

Here ∂_ν denotes the derivative with respect to the νth variable. For example, we have for $\Phi(x^2,y^2) := \sqrt{x^2+y^2+\varepsilon^2}$ that $\partial_1\Phi(x^2,y^2) = \partial_2\Phi(x^2,y^2) = 1/(2\sqrt{x^2+y^2+\varepsilon^2})$ and for $\Phi(x^2,y^2) := \sqrt{x^2+\varepsilon^2}+\sqrt{y^2+\varepsilon^2}$ that $\partial_1\Phi(x^2,y^2) = 1/(2\sqrt{x^2+\varepsilon^2})$ and $\partial_2\Phi(x^2,y^2) = 1/(2\sqrt{y^2+\varepsilon^2})$.

On the other hand, the so-called anisotropic edge enhancing diffusion (EED) acts via

$$\begin{aligned} \partial_t u &= \nabla\cdot(D\nabla u), \\ u(x,0) &= \tilde{f}(x), \end{aligned} \tag{7}$$

with appropriate boundary conditions, mainly Neumann boundary conditions in image processing, and with the *diffusion tensor*

$$D := V \begin{pmatrix} g(|\nabla u_\sigma|) & 0 \\ 0 & 1 \end{pmatrix} V^{\mathrm{T}}, \qquad V := (v \;\; v^\perp), \qquad v := \frac{\nabla u_\sigma}{|\nabla u_\sigma|}. \tag{8}$$

Here $u_\sigma = u * K_\sigma$ denotes the convolution of u with the Gaussian of standard deviation σ, and g is a decreasing nonnegative function. In applications, the function

$$g(|s|) = \begin{cases} 1 - e^{-\frac{3.31488}{(s/\alpha)^8}}, & s > 0, \\ 1, & s = 0, \end{cases}$$

introduced by Weickert in [27] has shown good performance.

A relation to regularization methods can be seen as follows: If we use instead of (8) the matrices (6), then (5) can be considered as a semidiscretization of (7) with an implicit Euler step of time step size λ. The following wavelet methods are related to explicit time discretizations so that we can achieve only approximations of the corresponding regularization method. For further investigations in this direction, see [24]. Note that according to [27] we will call a method anisotropic if the diagonal matrix in the diffusion tensor contains different nonzero diagonal entries. In this sense, the ROF method is an isotropic one.

3. Anisotropic Haar-wavelet shrinkage.

In this section, we return to our discrete setting from the beginning of the paper. Let $h_0 := \frac{1}{2}[1\;\;1]$ and $h_1 := \frac{1}{2}[1\;\;-1]$ be the filters of the Haar-wavelet. For convenience of notation, we use periodic boundary conditions and denote by $H_0 \in \mathbb{R}^{n,n}$ and $H_1 \in \mathbb{R}^{n,n}$ the corresponding circulant matrices. A remark concerning Neumann boundary conditions can be found at the end of this section. The following remark shows the link between the continuous considerations in the previous section and our discrete setting. Basically we consider discretizations of continuous images on a regular grid and approximate the partial derivatives by special differences related to our Haar-wavelet filters.

Remark 3.1. (i) Discretizing a periodic smooth function u on $[0,1)^2$ at the grid $\{(j,k)/h : j,k = 0,\dots,n-1\}$ and setting $\boldsymbol{u} := (u_{j,k})_{j,k=0}^{n-1} = \big(u(\frac{j}{h},\frac{k}{h})\big)_{j,k=0}^{n-1}$, we see by using the two-dimensional Taylor expansion that

$$\frac{u_{j+1,k+1} - u_{j,k+1} + u_{j+1,k} - u_{j,k}}{2h} = \partial_x u\left(j+\frac{h}{2}, k+\frac{h}{2}\right) + \mathcal{O}(h^2);$$

i.e., the left-hand side is a consistent discretization of $\partial_x u$. In matrix-vector notation this yields the following approximation of ∇u:

$$2\begin{pmatrix} H_1 \boldsymbol{u} H_0^{\mathrm{T}} \\ H_0 \boldsymbol{u} H_1^{\mathrm{T}} \end{pmatrix} = -h \begin{pmatrix} \left(\partial_x u\left(j+\frac{h}{2}, k+\frac{h}{2}\right)\right)_{j,k=0}^{n-1} \\ \left(\partial_y u\left(j+\frac{h}{2}, k+\frac{h}{2}\right)\right)_{j,k=0}^{n-1} \end{pmatrix} + \mathcal{O}(h^2). \tag{9}$$

Reshaping $\boldsymbol{u}$ columnwise into a vector u and using that $R\boldsymbol{u}S^{\mathrm{T}} = (S \otimes R)u$, the left-hand side of (9) becomes $2\begin{pmatrix} H_0 \otimes H_1 \\ H_1 \otimes H_0 \end{pmatrix} u$. For digital images one sets $h := 1$.

(ii) In [28], an ℓ_2-stable, conditionally consistent, so-called locally semianalytic scheme (LSAS) for the numerical solution of the EED equation (7) was developed. It involves a sophisticated spatial discretization and an explicit Euler scheme as temporal discretization. With respect to our notation the iterative LSAS scheme computes at every time step, with time step size τ based on the old iterate u_{old}, the new one u_{new} by the following steps:

1. $$\begin{pmatrix} c_{00} \\ c_{01} \\ c_{10} \\ c_{11} \end{pmatrix} := \underbrace{\begin{pmatrix} H_0 \otimes H_0 \\ H_0 \otimes H_1 \\ H_1 \otimes H_0 \\ H_1 \otimes H_1 \end{pmatrix}}_{A} u_{old},$$
2. $$\begin{pmatrix} d_{01} \\ d_{10} \end{pmatrix} := V \begin{pmatrix} e^{-4\tau\, g(|\nabla u_{old,\sigma}|)} & 0 \\ 0 & e^{-4\tau} \end{pmatrix} V^{\mathrm{T}} \begin{pmatrix} c_{01} \\ c_{10} \end{pmatrix},$$
   $$\begin{aligned} d_{00} &:= c_{00}, \\ d_{11} &:= e^{-4\tau\,(g(|\nabla u_{old,\sigma}|)+1)} c_{11}, \end{aligned}$$
3. $u_{new} := A^{\mathrm{T}} \left(d_{00}^{\mathrm{T}}, d_{01}^{\mathrm{T}}, d_{10}^{\mathrm{T}}, d_{11}^{\mathrm{T}}\right)^{\mathrm{T}}$,

where

$$\nabla u_{old,\sigma} := 2\begin{pmatrix} H_0 \otimes H_1 \\ H_1 \otimes H_0 \end{pmatrix} (u_{old} * K_\sigma)$$

and V is chosen in accordance with (8) as $V := \begin{pmatrix} c & -s \\ s & c \end{pmatrix}$ with

$$c := \operatorname{diag}\big(((H_0 \otimes H_1)u_{old,\sigma})/w\big),\ \ s := \operatorname{diag}\big(((H_1 \otimes H_0)u_{old,\sigma})/w\big),$$

$$w := \sqrt{\big((H_0 \otimes H_1)u_{old,\sigma}\big)^2 + \big((H_1 \otimes H_0)u_{old,\sigma}\big)^2},$$

and componentwise quotients $((H_0 \otimes H_1)u_{old,\sigma})/w$ and squares $((H_0 \otimes H_1)u_{old,\sigma})^2$ of vectors.

We consider the undecimated discrete Haar-wavelet transform up to level m. For $k = 1, \ldots, m$, let $H_\nu^{(k)} \in \mathbb{R}^{n,n}$, $\nu \in \{0,1\}$, be the circulant matrix corresponding to the filter $h_\nu^{(k)} = \frac{1}{2}(1, \underbrace{0, \ldots, 0}_{2^{k-1}-1}, (-1)^\nu)$ with $2^{k-1}-1$ inserted zeros between the filter coefficients. Further, we set

$$\begin{pmatrix} H_{00}^{(k)} \\ H^{(k)} \\ H_{11}^{(k)} \end{pmatrix} := \begin{pmatrix} H_{00}^{(k)} \\ H_{10}^{(k)} \\ H_{01}^{(k)} \\ H_{11}^{(k)} \end{pmatrix} = \begin{pmatrix} H_0^{(k)} \otimes H_0^{(k)} \\ H_0^{(k)} \otimes H_1^{(k)} \\ H_1^{(k)} \otimes H_0^{(k)} \\ H_1^{(k)} \otimes H_1^{(k)} \end{pmatrix} \prod_{l=1}^{k-1} (H_0^{(l)} \otimes H_0^{(l)}).$$

Then the matrix

$$A = \begin{pmatrix} H_{00}^{(m)} \\ H^{(1)} \\ \vdots \\ H^{(m)} \\ H_{11}^{(1)} \\ \vdots \\ H_{11}^{(m)} \end{pmatrix} \in \mathbb{R}^{(3m+1)N,N} \tag{10}$$

satisfies $A^{\mathrm{T}}A = I$ while $AA^{\mathrm{T}} \neq I$. Let $V^{(k)}$ be orthogonal matrices, and let

$$\Lambda^{(k)} := \operatorname{diag}\left(\lambda_j^{(k)}\right)_{j=1}^{2N}, \quad \Lambda_{11}^{(k)} := \operatorname{diag}\left(\lambda_{11,j}^{(k)}\right)_{j=1}^{N}, \quad k = 1, \dots, m,$$

be diagonal matrices with nonnegative entries. For $p \in [1, 2]$, we consider the minimization problem

$$\operatorname*{argmin}_{u \in \mathbb{R}^N} \left\{ \frac{1}{2}\|f - u\|_2^2 + \frac{1}{p}\sum_{k=1}^{m} \|\Lambda^{(k)}\big(V^{(k)}\big)^{\mathrm{T}} H^{(k)} u\|_p^p + \frac{1}{p}\sum_{k=1}^{m} \|\Lambda_{11}^{(k)} H_{11}^{(k)} u\|_p^p \right\}.$$

In our numerical examples, we will use only $p = 1$ and $p = 2$. Since $A^{\mathrm{T}}A = I$, this is equivalent to

$$\operatorname*{argmin}_{u \in \mathbb{R}^N} \left\{ \frac{1}{2}\|Af - Au\|_2^2 + \frac{1}{p}\sum_{k=1}^{m} \|\Lambda^{(k)}\big(V^{(k)}\big)^{\mathrm{T}} H^{(k)} u\|_p^p + \frac{1}{p}\sum_{k=1}^{m} \|\Lambda_{11}^{(k)} H_{11}^{(k)} u\|_p^p \right\}.$$

Using the notation

$$c := Af = \left(c_{00}^{(m)},\ c^{(1)}, \dots,\ c^{(m)},\ c_{11}^{(1)}, \dots,\ c_{11}^{(m)}\right)^{\mathrm{T}}, \qquad d := Au,$$

this can be rewritten as

$$\operatorname*{argmin}_{d \in \mathbb{R}^{(3m+1)N}} \left\{ \frac{1}{2}\|c - d\|_2^2 + \frac{1}{p}\sum_{k=1}^{m} \|\Lambda^{(k)}\big(V^{(k)}\big)^{\mathrm{T}} d^{(k)}\|_p^p + \frac{1}{p}\sum_{k=1}^{m} \|\Lambda_{11}^{(k)} d_{11}^{(k)}\|_p^p \right\} \quad \text{subject to } d \in \mathcal{R}(A).$$

Note that $d \in \mathcal{R}(A)$ is equivalent to $(I - AA^{\mathrm{T}})d = 0$; i.e., the orthogonal projection of d onto the kernel of A^{T} has to be 0. In other words, if $\hat{d}$ is a solution of this problem, then $AA^{\mathrm{T}}\hat{d}$ is just the orthogonal projection of $\hat{d}$ onto $\mathcal{R}(A)$. We will not solve this minimization problem in step (i) of our inpainting algorithm, but rather the following problem, which is obtained by neglecting the constraint:

$$\operatorname*{argmin}_{d \in \mathbb{R}^{(3m+1)N}} \left\{ \frac{1}{2}\|c - d\|_2^2 + J_{\Lambda,p}(d) \right\},$$

where

$$J_{\Lambda,p}(d) := \frac{1}{p}\sum_{k=1}^{m} \|\Lambda^{(k)}\big(V^{(k)}\big)^{\mathrm{T}} d^{(k)}\|_p^p + \frac{1}{p}\sum_{k=1}^{m} \|\Lambda_{11}^{(k)} d_{11}^{(k)}\|_p^p. \tag{11}$$

This functional can be decoupled as

$$\frac{1}{2}\|c_{00}^{(m)} - d_{00}^{(m)}\|_2^2 + \sum_{k=1}^{m}\left(\frac{1}{2}\|c^{(k)} - d^{(k)}\|_2^2 + \frac{1}{p}\|\Lambda^{(k)}\big(V^{(k)}\big)^{\mathrm{T}} d^{(k)}\|_p^p\right)$$
$$+ \sum_{k=1}^{m}\left(\frac{1}{2}\|c_{11}^{(k)} - d_{11}^{(k)}\|_2^2 + \frac{1}{p}\|\Lambda_{11}^{(k)} d_{11}^{(k)}\|_p^p\right). \tag{12}$$

Now the three parts of the functional can be minimized separately, which leads to the following solution.

Lemma 3.2. *The minimizer $\hat{d}$ of the functional (12) is given by*

$$\begin{aligned} \hat{d}_{00}^{(m)} &= c_{00}^{(m)}, \\ \hat{d}^{(k)} &= V^{(k)}\,\mathcal{T}_{\Lambda^{(k)},p}\left((V^{(k)})^{\mathrm{T}} c^{(k)}\right), \quad k = 1,\dots,m, \\ \hat{d}_{11}^{(k)} &= \mathcal{T}_{\Lambda_{11}^{(k)},p}(c_{11}^{(k)}), \quad k = 1,\dots,m, \end{aligned} \tag{13}$$

with the following shrinkage procedures $\mathcal{T}_{\cdot,\cdot}$:

(i) *the soft shrinkage $\mathcal{T}_{\Lambda,1}$ for $p = 1$,*

(ii) *$\mathcal{T}_{\Lambda,p}(y) = F_{\Lambda,p}^{-1}(y)$, where $F_{\Lambda,p}$ is the injective mapping*

$$F_{\Lambda,p}(x) = x + \Lambda^p(\mathrm{sgn}(x) \circ |x|^{p-1})$$

for $p \in (1,2)$,

(iii) *$\mathcal{T}_{\Lambda,2}(y) := (I + \Lambda^2)^{-1} y$ for $p = 2$.*

Moreover, we have for $p \in (1,2]$ that

$$|\mathcal{T}_{\Lambda,p}(y)|^p \geq (I + \Lambda^p)^{-p}|y|^p - 1. \tag{14}$$

Proof. Since the matrices $V^{(k)}$ are orthogonal, we immediately obtain assertion (i).

In the following, we restrict our attention to the central functional, i.e., to $\hat{d}^{(k)}$. For $p \in (1,2]$, the functional is differentiable, and the minimizer has to fulfill

$$\begin{aligned} 0 &= \hat{d}^{(k)} - c^{(k)} + V^{(k)}\Lambda^{(k)}(\mathrm{sgn}(\Lambda^{(k)}(V^{(k)})^{\mathrm{T}}\hat{d}^{(k)}) \circ |\Lambda^{(k)}(V^{(k)})^{\mathrm{T}}\hat{d}^{(k)}|^{p-1}), \\ (V^{(k)})^{\mathrm{T}} c^{(k)} &= (V^{(k)})^{\mathrm{T}}\hat{d}^{(k)} + (\Lambda^{(k)})^p \mathrm{sgn}((V^{(k)})^{\mathrm{T}}\hat{d}^{(k)}) \circ |(V^{(k)})^{\mathrm{T}}\hat{d}^{(k)}|^{p-1}. \end{aligned}$$

Then $x = (V^{(k)})^{\mathrm{T}}\hat{d}^{(k)}$ is the solution of $(V^{(k)})^{\mathrm{T}} c^{(k)} = F_{\Lambda^{(k)},p}(x)$ and $\hat{d}^{(k)} = V^{(k)}x$. In particular, we have for $p = 2$ that $x = (I + \Lambda^2)^{-1}(V^{(k)})^{\mathrm{T}} c^{(k)}$.

We prove the last assertion (14) componentwise. For $x, y \in \mathbb{R}$ and $\lambda \in \mathbb{R}_{\geq 0}$ the equation $y = x + \lambda^p \mathrm{sgn}(x)|x|^{p-1}$ implies that

$$|y| = |x| + \lambda^p |x|^{p-1}.$$

Then, we see for $|x| \geq 1$ and $p \in (1,2]$ that $|y| \leq |x| + \lambda^p |x|$ and, consequently, $|x| \geq (1+\lambda^p)^{-1}|y|$. For $|x| < 1$, we have that $|y| \leq |x|^{p-1} + \lambda^p |x|^{p-1}$ so that $1 > |x|^{p-1} \geq (1+\lambda^p)^{-1}|y|$. Thus, $1 > (1 + \lambda^p)^{-p}|y|^p$ and $|x|^p \geq 0 > (1 + \lambda^p)^{-p}|y|^p - 1$. ∎

Let us denote the whole shrinkage procedure by $\hat{d} = \mathcal{T}_{\Lambda,p}\, c$. Finally, we can compute the denoised image u of f by $u = A^{\mathrm{T}}\hat{d}$. With this denoising procedure our inpainting algorithm reads as follows.

Algorithm I.1 (Exact data)

Initialization: u_0

For $r = 0, \ldots$ iterate until convergence

(i) Compute $\hat{u}_{r+1} = A^{\mathrm{T}}\mathcal{T}_{\Lambda,p}(Au_r)$ with $\mathcal{T}_{\Lambda,p}$ defined by Lemma 3.2.

(ii) Set

$$u_{r+1}(j) := \begin{cases} f(j) & \text{if } j \in C, \\ \hat{u}_{r+1}(j) & \text{otherwise.} \end{cases}$$

Output: u^*

Algorithm II.1 (Noisy data)

Same as Algorithm I except that we have to apply step (i) to the final iterate u^* again.

Output: $u^\diamond = \hat{u}^*$

The set

$$\mathcal{C} := \{g \in \mathbb{R}^N : g(j) = f(j) \quad \forall j \in C\}$$

is nonempty, closed, and convex so that its indicator function $\iota_{\mathcal{C}}$ is a proper lsc convex function. Thus, step (ii) of the inpainting procedure also reads

$$u_{r+1} = \operatorname*{argmin}_{u \in \mathbb{R}^N} \left\{ \frac{1}{2}\|\hat{u}_{r+1} - u\|_2^2 + \iota_{\mathcal{C}}(u) \right\}.$$

Thus, the whole algorithm can be rewritten in the form

$$d_{r+1} = \operatorname*{argmin}_{d \in \mathbb{R}^{(3m+1)N}} \left\{ \frac{1}{2}\|Au_r - d\|_2^2 + J_{\Lambda,p}(d) \right\}, \tag{15}$$

$$u_{r+1} = \operatorname*{argmin}_{u \in \mathbb{R}^N} \left\{ \frac{1}{2}\|A^{\mathrm{T}}d_{r+1} - u\|_2^2 + \iota_{\mathcal{C}}(u) \right\}, \tag{16}$$

where $J_{\Lambda,p}(d)$ is defined in (11).

Remark 3.3 (Neumann boundary conditions). If we assume mirrored boundaries, we have to replace the circulant matrices H_ν, $\nu = 0, 1$, by the Toeplitz matrices $H_\nu \in \mathbb{R}^{n+1,n+2}$ corresponding to the filters h_ν. Let $\tilde{H}_\nu \in \mathbb{R}^{n,n+1}$ denote the matrices obtained from H_ν^{T} by canceling their first and last rows. Then we have that

$$H_0^{\mathrm{T}}H_0 + H_1^{\mathrm{T}}H_1 = \begin{pmatrix} \frac{1}{2} & & \\ & I & \\ & & \frac{1}{2} \end{pmatrix} \quad \text{and} \quad \tilde{H}_0H_0 + \tilde{H}_1H_1 = \begin{pmatrix} 0 & 1 & & & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & & & 1 & 0 \end{pmatrix}.$$

Consider one decomposition level $m = 1$. For higher levels we have to incorporate the corresponding zeros into the filters and mirror the boundaries according to the filter length. Let $\tilde{\mathbf{f}}$ denote the image obtained from $\mathbf{f}$ by mirroring the boundaries and let $A, \tilde{A}$ be defined as in

(10) but with the new Toeplitz matrices $H_\nu, \tilde{H}_\nu$, $\nu = 0, 1$. Then instead of (15) we solve the minimization problem

$$d_{r+1} = \underset{d \in \mathbb{R}^{4\tilde{N}}}{\operatorname{argmin}} \left\{ \frac{1}{2}\|A\tilde{u}_r - d\|_2^2 + \frac{1}{p}\|\Lambda\, V^{\mathrm{T}}\, d\|_p^p + \frac{1}{p}\|\Lambda_{11}\, d_{11}\|_p^p \right\},$$

where $\tilde{N} = (n+1)^2$ and $\hat{u}_{r+1} := \tilde{A} d_{r+1}$.

4. Convergence considerations. Following [6], we show the convergence of our inpainting algorithm by identifying it as a forward-backward splitting algorithm to minimize the sum of two operators. There exists a vast literature on forward-backward splitting algorithms and related fixed point iterations; see Remark 4.2 below. In this paper, we need only the following setting in the Hilbert space $\mathbb{R}^N$ with the Euclidean norm.

For any proper, convex, lsc function φ, the *proximal operator* is defined by

$$\operatorname{prox}_\varphi(x) = \underset{y \in \mathbb{R}^N}{\operatorname{argmin}} \left\{ \frac{1}{2}\|x - y\|_2^2 + \varphi(y) \right\}$$

and its *envelope* by

$${}^1\varphi(x) = \min_{y \in \mathbb{R}^N} \left\{ \frac{1}{2}\|x - y\|_2^2 + \varphi(y) \right\}.$$

By [1, Theorem 5.2], the function ${}^1\varphi$ is convex and differentiable, and its gradient is

$$\nabla \left({}^1\varphi \right)(x) = x - \operatorname{prox}_\varphi(x). \tag{17}$$

Lemma 4.1 (see [6]). *Let $F_1 : \mathbb{R}^N \to \mathbb{R} \cup \{+\infty\}$ be a proper, convex, lsc function, and let $F_2 : \mathbb{R}^N \to \mathbb{R} \cup \{+\infty\}$ be a proper, convex, differentiable function with a Lipschitz continuous gradient with Lipschitz constant < 2. Assume that*

$$\underset{u \in \mathbb{R}^N}{\operatorname{argmin}} \{F_1(u) + F_2(u)\}$$

has a solution. Then, for any initial guess u_0, the so-called proximal forward-backward splitting

$$u_{r+1} = \operatorname{prox}_{F_1}\left(u_r - \nabla F_2(u_r)\right) \tag{18}$$

converges to a minimizer of the functional $F_1 + F_2$.

The iteration (18) is a special case of a more general class of algorithms which we briefly outline in the following remark.

Remark 4.2. For subdifferentiable functions $F_1, F_2 : \mathcal{H} \to \mathbb{R} \cup \{+\infty\}$ on a Hilbert space $\mathcal{H}$ we have that

$$\hat{u} = \underset{u \in \mathcal{H}}{\operatorname{argmin}} \{F_1(u) + F_2(u)\} \quad \Leftrightarrow \quad 0 \in \partial(F_1 + F_2)(\hat{u}).$$

Under certain conditions on F_1 and F_2 this is equivalent to $0 \in \partial F_1(\hat{u}) + \partial F_2(\hat{u})$. If $\partial F_1, \partial F_2$ are maximal monotone operators, Lions and Mercier [18] and, independently, Passty [21] suggested solving the inclusion on the right-hand side by the forward-backward splitting

$$\hat{u} = (I + c\,\partial F_1)^{-1}(I - c\,\partial F_2)\hat{u}. \tag{19}$$

Under certain conditions on ∂F_2 and the step size c, it was proved that the Picard iteration of (19) converges weakly to a minimizer $\hat{u}$; see, e.g., [14, 26]. Meanwhile there exist various generalizations of this algorithm such as those in [8].

Since in our special problem F_1 is proper, convex, and lsc and F_2 is differentiable, we have that $(I + \partial F_1)^{-1} = \mathrm{prox}_{F_1}$ and $\partial F_2 = \nabla F_2$, so that (19) with $c = 1$ coincides with (18).

We now return to Algorithm I.1. For our problem, we set $F_1 := \iota_C$ and $F_2 := ({}^1J_{\Lambda,p})(A\cdot)$. Then we obtain

$$F_2(u) = \min_{d\in\mathbb{R}^{(3m+1)N}} \left\{ \frac{1}{2}\|Au - d\|_2^2 + J_{\Lambda,p}(d) \right\}$$

so that

$$\begin{aligned} F_1(u) + F_2(u) &= \iota_C(u) \; + \min_{d\in\mathbb{R}^{(3m+1)N}} \left\{ \frac{1}{2}\|Au - d\|_2^2 + J_{\Lambda,p}(d) \right\} \\ &= \iota_C(u) \; + \; \frac{1}{2}\|Au - \mathcal{T}_{\Lambda,p}(Au)\|_2^2 + J_{\Lambda,p}(\mathcal{T}_{\Lambda,p}(Au)). \end{aligned} \tag{20}$$

Further, we obtain by (17) that F_2 is differentiable with

$$\nabla F_2(u) = \nabla \left({}^1J_{\Lambda,p} \circ A \right)(u) = A^{\mathrm{T}} \left(Au - \mathrm{prox}_{J_{\Lambda,p}}(Au) \right). \tag{21}$$

Now the forward-backward splitting (18) becomes

$$\begin{aligned} u_{r+1} &= \mathrm{prox}_{F_1}\left(u_r - \nabla F_2(u_r)\right) \\ &= \operatorname*{argmin}_{u\in\mathbb{R}^N} \left\{ \frac{1}{2}\|u_r - \nabla F_2(u_r) - u\|_2^2 + \iota_C(u) \right\} \\ &= \operatorname*{argmin}_{u\in\mathbb{R}^N} \left\{ \frac{1}{2}\|u_r - A^{\mathrm{T}}\big(Au_r - \mathrm{prox}_{J_{\Lambda,p}}(Au_r)\big) - u\|_2^2 \; + \; \iota_C(u) \right\} \\ &= \operatorname*{argmin}_{u\in\mathbb{R}^N} \left\{ \frac{1}{2}\|A^{\mathrm{T}}\mathrm{prox}_{J_{\Lambda,p}}(Au_r) - u\|_2^2 \; + \; \iota_C(u) \right\}. \end{aligned} \tag{22}$$

By (15) and (16) this coincides with the sequence produced by our Algorithm I.1. Next, we show that $F_1 + F_2$ in (20) is coercive.

Lemma 4.3. *The functional $F_1 + F_2$ in (20) is coercive.*

Proof. By (20) we obtain

$$F_1(u) + F_2(u) = \iota_C(u) + \frac{1}{2}\|Au - \mathcal{T}_{\Lambda,p}(Au)\|_2^2 + J_{\Lambda,p}(\mathcal{T}_{\Lambda,p}(Au)) \geq J_{\Lambda,p}(\mathcal{T}_{\Lambda,p}(Au)).$$

Let $Au := ((Au)_{00}^{(m)}, (Au)^{(1)}, \ldots, (Au)^{(m)}, (Au)_{11}^{(1)}, \ldots, (Au)_{11}^{(m)})^{\mathrm{T}}$. Then we see by (12) and (13) that

$$\begin{aligned} J_{\Lambda,p}(\mathcal{T}_{\Lambda,p}(Au)) = \frac{1}{p}\sum_{k=1}^{m} \|\Lambda^{(k)}\mathcal{T}_{\Lambda^{(k)},p}((V^{(k)})^{\mathrm{T}}(Au)^{(k)})\|_p^p \\ + \frac{1}{p}\sum_{k=1}^{m} \|\Lambda_{11}^{(k)}\mathcal{T}_{\Lambda_{11}^{(k)},p}((Au)_{11}^{(k)})\|_p^p. \end{aligned} \tag{23}$$

Now we have by (14) and by definition of the soft shrinkage function that

$$\frac{1}{p}\left(\lambda_j^{(k)}\right)^p \left|\mathcal{T}_{\lambda_j^{(k)},p}(y)\right|^p \geq \begin{cases} \lambda_j^{(k)}|y| - \left(\lambda_j^{(k)}\right)^2 & \text{for } p = 1, \\ \frac{1}{p}\left(\frac{\lambda_j^{(k)}}{1+\left(\lambda_j^{(k)}\right)^p}\right)^p |y|^p - \frac{1}{p}\left(\lambda_j^{(k)}\right)^p & \text{for } p \in (1,2]. \end{cases}$$

Thus, setting

$$\kappa_1 := \frac{1}{p} \min_{\substack{j=1,\dots,2N \\ i=1,\dots,N \\ k=1,\dots,m}} \left\{ \left(\frac{\lambda_j^{(k)}}{1+\left(\lambda_j^{(k)}\right)^p}\right)^p, \left(\frac{\lambda_{11,i}^{(k)}}{1+\left(\lambda_{11,i}^{(k)}\right)^p}\right)^p \right\}$$

and

$$\kappa_2 := \begin{cases} \sum_{k=1}^m \left(\sum_{j=1}^{2N} \left(\lambda_j^{(k)}\right)^2 + \sum_{i=1}^N \left(\lambda_{11,j}^{(k)}\right)^2 \right) & \text{for } p = 1, \\ \frac{1}{p}\sum_{k=1}^m \left(\sum_{j=1}^{2N} \left(\lambda_j^{(k)}\right)^p + \sum_{i=1}^N \left(\lambda_{11,j}^{(k)}\right)^p \right) & \text{for } p \in (1,2] \end{cases}$$

and applying that $\|x\|_p \geq \|x\|_2$ for $p \in [1,2]$, we get

$$\begin{aligned} J_{\Lambda,p}(\mathcal{T}_{\Lambda,p}(Au)) &\geq \kappa_1 \left(\sum_{k=1}^m \|(V^{(k)})^{\mathrm{T}}(Au)^{(k)}\|_p^p + \sum_{k=1}^m \|(Au)_{11}^{(k)}\|_p^p \right) - \kappa_2 \\ &\geq \kappa_1 \left(\sum_{k=1}^m \|(Au)^{(k)}\|_2^p + \sum_{k=1}^m \|(Au)_{11}^{(k)}\|_2^p \right) - \kappa_2. \end{aligned}$$

Using the notation $A_0 := H_{00}^{(m)}$ and

$$A_1 := \left((H^{(1)})^{\mathrm{T}}, \dots, (H^{(m)})^{\mathrm{T}}, (H^{(1)})_{11}^{\mathrm{T}}, \dots, (H^{(m)})_{11}^{\mathrm{T}}\right)^{\mathrm{T}},$$

this can be rewritten as

$$J_{\Lambda,p}(\mathcal{T}_{\Lambda,p}(Au)) \geq \kappa_1 \|A_1 u\|_2^p - \kappa_2. \tag{24}$$

By Lemma 4.4 below, the matrix $A_0^{\mathrm{T}} A_0$ has the simple eigenvalue 1 with a corresponding normed eigenvector $\tilde{u} = \frac{1}{\sqrt{N}}\mathbf{1}_N$. Since $A^{\mathrm{T}}A = I$, it follows that $A_1^{\mathrm{T}} A_1$ has the simple eigenvalue 0 and that the kernel of $A_1^{\mathrm{T}} A_1$ is spanned by $\tilde{u}$. Now we obtain for the orthogonal decomposition $u = v + a\tilde{u}$ that $|a| \geq \|u\|_2 - \|v\|_2$ and

$$\|A_1 u\|_2^2 = \|A_1 v\|_2^2 \geq \eta_2 \|v\|_2^2, \tag{25}$$

where $\eta_2 > 0$ is the second smallest eigenvalue of $A_1^{\mathrm{T}} A_1$. Now we fix a constant $c \in (\frac{1}{\sqrt{N}+1}, 1)$ and consider two cases:

1. For $\|v\|_2 \geq c\,\|u\|_2$, we conclude by (24) and (25) that

$$F_1(u) + F_2(u) \geq \kappa_1 \|A_1 u\|_2^p - \kappa_2 \geq \kappa_1 \sqrt{\eta_2}^p \|v\|_2^p - \kappa_2 \geq \kappa_1 \sqrt{\eta_2}^p c^p \|u\|_2^p - \kappa_2.$$

2. For $\|v\|_2 < c\|u\|_2$ it holds that $|a| > (1-c)\|u\|_2$. Hence, we have for any $i_0 \in C$ that

$$|u_{i_0}| = |v_{i_0} + a\tilde{u}_{i_0}| \geq |a||\tilde{u}_{i_0}| - |v_{i_0}| > (1-c)\|u\|_2|\tilde{u}_{i_0}| - c\|u\|_2 = \|u\|_2 \frac{1 - c(1+\sqrt{N})}{\sqrt{N}}.$$

Thus, we see for $\|u\|_2$ large enough that $|u_{i_0}| > |f_{i_0}|$ and, consequently, $F_1(u) + F_2(u) \geq \iota_C(u) = +\infty$. ∎

Lemma 4.4. *The matrix $A_0^T A_0$ has 1 as a simple eigenvalue with corresponding eigenvector $\tilde{u} = \frac{1}{\sqrt{N}}\mathbf{1}_N$.*

Proof. Using multiplication rules for tensor products, we obtain that

$$A_0^{\mathrm{T}} A_0 = B^{\mathrm{T}}B \otimes B^{\mathrm{T}}B, \quad B := \prod_{l=1}^{m} H_0^{(l)} = \frac{1}{2^m}\mathrm{circ}\,([\underbrace{1 \ldots 1}_{2^m}\, 0 \ldots 0]).$$

By [10], the circulant matrix B has eigenvectors $\frac{1}{\sqrt{n}}(e^{-\frac{2\pi ijk}{n}})_{j=0}^{n}$ and eigenvalues $\beta_0 = 1$ and

$$|\beta_k| = \left|\frac{1}{2^m}\sum_{j=0}^{2^m-1} e^{-\frac{2\pi ijk}{n}}\right| = \frac{1}{2^m}\frac{|1 - e^{-\frac{2\pi ij2^m}{n}}|}{|1 - e^{-\frac{2\pi ij}{n}}|} = \frac{1}{2^m}\prod_{p=1}^{m}|1 + e^{-\frac{2\pi ij2^{m-p}}{n}}| < 1,$$

$k = 1, \ldots, n-1$. The last inequality holds true because $|1 + e^{-\frac{2\pi ij2^{m-p}}{n}}| \leq 2$ for $p = 1, \ldots, m$ with strict inequality for $p = m$. ∎

In summary, we obtain the following convergence result.

Theorem 4.5. *The sequence $\{u_r\}_{r=0}^{\infty}$ produced by Algorithm* I.1 *converges for any start image u_0 and $p \in [1,2]$ to a minimizer of the functional $F_1 + F_2$ in* (20).

Proof. By (22), the sequence produced by our Algorithm I.1 coincides with the sequence generated by the forward-backward splitting algorithm (18). Now the assertion follows since the functional $F_1 + F_2$ in (20) fulfills the convergence assumptions of Lemma 4.1: The functions F_1 and F_2 are proper, convex, and lsc. By Lemma 4.3 the functional $F_1 + F_2$ is coercive so that there exists at least one minimizer of the functional. Finally, since $\|A\|_2 = 1$ and $I - \mathrm{prox}_{J_{\Lambda,p}}$ is nonexpansive, it is easy to check as in [6] that F_2 has a gradient with Lipschitz constant 1. ∎

With respect to Remark 4.2, we note that for our setting $(I + \partial F_1)^{-1}(I - \partial F_2)$ is an averaged operator, i.e., the strictly convex combination of the identity operator and a nonexpansive mapping. As an alternative to Lemma 4.1 one could also use convergence results for Picard iterations of averaged operators; see [5, 17, 19, 20, 23].

Remark 4.6. Numerical experiments indicate that Algorithm I.1 converges linearly. However, we have not proved this so far. In [26, Proposition 1(d)], Tseng gives a sufficient condition for linear convergence. Unfortunately, it cannot be applied here since neither ∂F_1 nor ∇F_2 is strongly monotone.

5. Numerical examples.

Finally, we present some numerical examples; in particular, we compare our algorithm with the algorithm in [6] without thresholding of the smoothest coefficients. Since the results for noisy data with a small amount of noise are similar to those for exact data, we restrict our attention to exact input data.

All programs were written in MATLAB. We have always assumed Neumann boundary conditions and we have used the following stopping criterion for the iterations: $\|u_{r+1} - u_r\|_2/\|u_{r+1}\|_2 \leq 5 \cdot 10^{-5}$. We compare the weighted ℓ_1-error $\text{err}_1 := \|u - f\|_1/N$, the weighted ℓ_2-error $\text{err}_2 := \|u - f\|_2/\sqrt{N}$, and the PSNR $:= 20 \cdot \log_{10}(255/\text{err}_2)$. The parameters were chosen with respect to the "best" PSNR.

We compare the following algorithms:

(A) The wavelet-based algorithm in [6] with the filters $h_0 := \frac{1}{4}[1\ 2\ 1]$, $h_1 := \frac{\sqrt{2}}{4}[1\ 0\ -1]$, and $h_2 := \frac{1}{4}[-1\ 2\ -1]$ and soft shrinkage of the high-pass coefficients at level k with the thresholds $c/\sqrt{2^k}$.

(B) Algorithm I.1 with our Haar-wavelet filters, $p = 1$, $V^{(k)} = I$, and soft shrinkage with threshold $\lambda_j^{(k)} := \lambda/\sqrt{2^k}$ and $\lambda_{11,j}^{(k)} := \lambda_{11}/\sqrt{2^k}$ at level k.

(C) The same algorithm as in (B) except that we use matrices $V^{(k)}$ inspired by the LSAS explained in Remark 3.1(ii): We convolve an appropriate guess $\tilde{f}$ of the original function with the Gaussian of standard derivation σ to obtain $\tilde{f}_\sigma$. Then, at level k, we set

$$V^{(k)} := \begin{pmatrix} c^{(k)} & -s^{(k)} \\ s^{(k)} & c^{(k)} \end{pmatrix}$$

with $c^{(k)} := \text{diag}\big(H_{01}^{(k)} \tilde{f}_\sigma/w\big)$, $s^{(k)} := \text{diag}\big(H_{10}^{(k)} \tilde{f}_\sigma/w\big)$, and $w^{(k)} := \sqrt{(H_{01}^{(k)} \tilde{f}_\sigma)^2 + (H_{10}^{(k)} \tilde{f}_\sigma)^2}$; i.e., we use the same matrices $V^{(k)}$ in each iteration step r.

(D) Algorithm I.1 with $p = 2$ and the following setting inspired by the LSAS for EED in Remark 3.1(ii): We define $V^{(k)}$ as in (C). In the shrinkage step we use

$$\big(I + (\Lambda^{(k)})^2\big)^{-1} := \begin{pmatrix} \text{diag}\big(e^{-4\tau\, g(w^{(k)})}\big) & 0 \\ 0 & \text{diag}\big(e^{-4\tau}\, 1_N\big) \end{pmatrix},$$
$$\big(I + (\Lambda_{11}^{(k)})^2\big)^{-1} := \text{diag}\big(e^{-4\tau\,(g(w^{(k)})+1)}\big)$$

with the vector 1_N of N ones. We still use the same matrices $V^{(k)}, \Lambda^{(k)}$, and $\Lambda_{11}^{(k)}$ in each iteration step r.

(E) The same algorithm as in (D) except that we do not freeze $V^{(k)}$ and the shrinkage matrices at the beginning of the algorithm with respect to $\tilde{f}_\sigma$ but compute them in each iteration step r with respect to the actual iterate u_r. Note that we have not proved the convergence for this algorithm. If we were to work only with one level of Haar-wavelet decomposition $m = 1$, then the restoration step could be considered as one time step of an iterative EED scheme discretized by LSAS.

In our *first example* we start with the image at the top right of Figure 1 which we also use as initial guess u_0. Alternatively, one could use the bottom left image in Figure 1 generated by the MATLAB cubic interpolation procedure "griddata" as initial guess. This leads to qualitatively similar results but with a smaller number of iterations. However, we have used this cubic interpolation in Algorithms (C) and (D) for $\tilde{f}$. Detailed results are given in the tables below. Here "iter" denotes the number of iterations. The corresponding images for the decomposition level 2 are depicted in Figure 2 and at the bottom right of Figure 1. The algorithms described in (B)–(E) perform much better than Algorithm (A). The PSNR

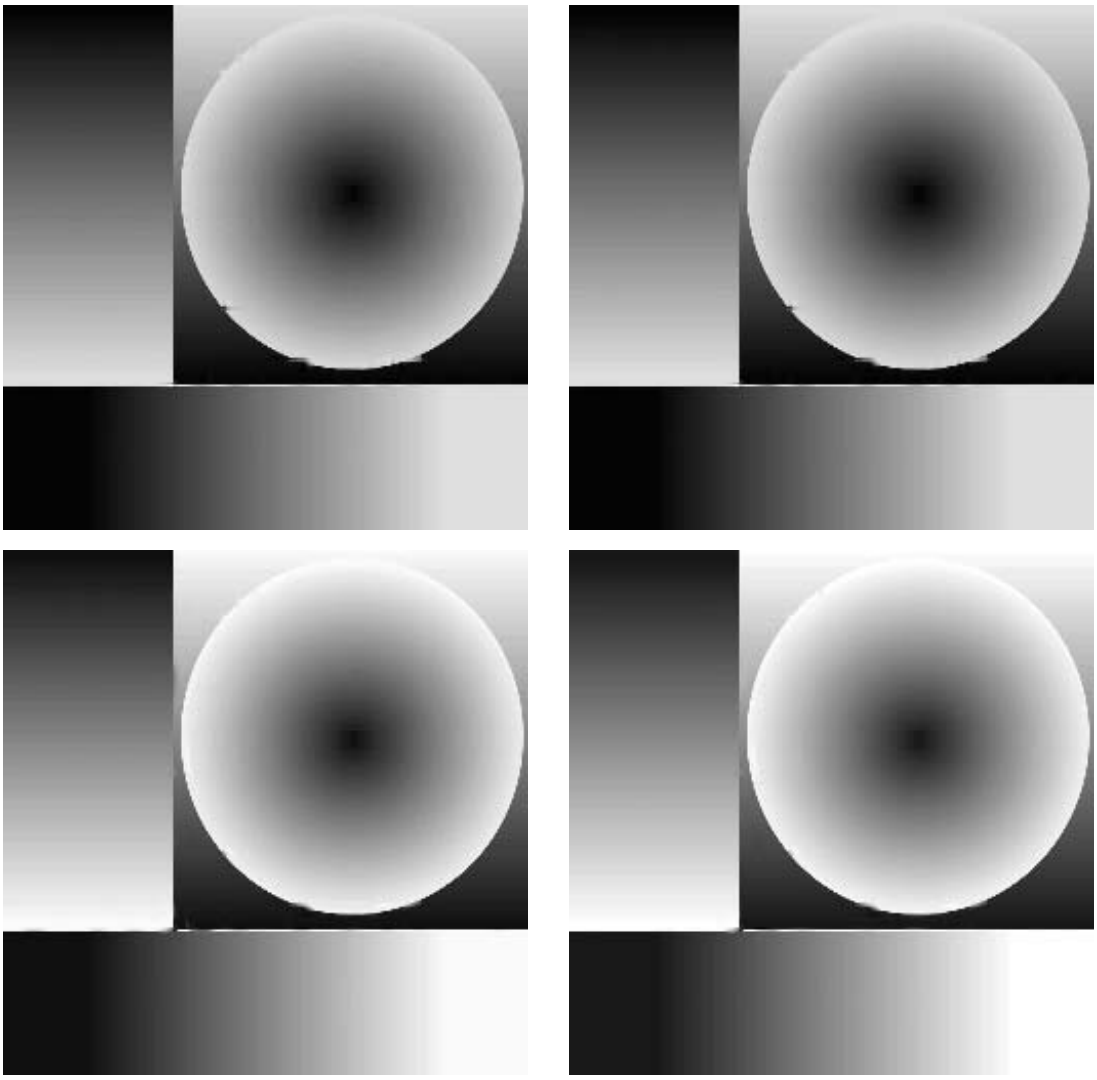

Figure 2. *Applications of Algorithms* (B)–(E) *with decomposition level* 2. *Top left: Algorithm* (B) *(*$PSNR = 36.42$, $\mathrm{err}_2 = 3.84$, $\mathrm{err}_1 = 0.27$*). Top right: Algorithm* (C) *(*$PSNR = 36.60$, $\mathrm{err}_2 = 3.36$, $\mathrm{err}_1 = 0.26$*). Bottom left: Algorithm* (D) *(*$PSNR = 36.79$, $\mathrm{err}_2 = 3.68$, $\mathrm{err}_1 = 0.29$*). Bottom right: Algorithm* (E) *(*$PSNR = 38.58$, $\mathrm{err}_2 = 2.99$, $\mathrm{err}_1 = 0.23$*). All algorithms reduce the artifacts at the straight lines. However, the images at the top contain similar errors at the boundary of the circle. The images at the bottom have the best quality.*

improves by approximately 3 dB if we use Algorithms (B)–(D) and by approximately 5 dB for Algorithm (E); compare Tables 1–5. Algorithms (B)–(E) considerably reduce the artifacts at the horizontal line. However, Algorithms (B) and (C) introduce some errors at the boundary of the circle. These artifacts do not appear if we apply Algorithms (D) and (E). In general, the PSNR cannot be substantially improved by choosing a higher decomposition level than $m = 2$.

In our *second example* we interpolate the image on the right-hand side of Figure 3. Again, we use this image as initial guess and its cubic interpolation as $\tilde{f}$ in Algorithms (C) and (D).

Table 1
Results of the inpainting Algorithm (A) *for the first example.*

Level	c	PSNR	err_2	err_1	iter
4	1.0	32.93	5.49	0.54	307
3	1.0	33.29	5.51	0.48	307
2	1.0	**33.27**	5.53	0.46	358
1	1.6	32.50	6.04	0.50	461

Table 2
Results of Algorithm (B) *for the first example.*

Level	λ	λ_{11}	PSNR	err_2	err_1	iter
4	1	8	34.84	4.61	0.36	272
3	1	10	35.52	4.27	0.29	235
2	1	100	**36.42**	3.84	0.27	278
1	1	100	35.89	4.09	0.28	814

Table 3
Results of Algorithm (C) *for the first example.*

Level	σ	λ	λ_{11}	PSNR	err_2	err_1	iter
4	4	1	8	35.43	4.31	0.36	244
3	4	1	10	35.97	4.05	0.30	223
2	4	1	100	**36.60**	3.76	0.26	269
1	4	1	100	36.03	4.02	0.26	811

Table 4
Results of the inpainting Algorithm (D) *for the first example.*

Level	σ	τ	α	PSNR	err_2	err_1	iter
4	4	1	2	35.19	4.43	0.47	73
3	4	1	2	36.08	4.00	0.36	79
2	4	1	2	**36.79**	3.68	0.29	106
1	4	1	2	36.83	3.83	0.28	208

Table 5
Results of Algorithm (E) *for the first example.*

Level	σ	τ	α	PSNR	err_2	err_1	iter
4	4	1	2	35.91	4.07	0.44	78
3	4	1	2	37.47	3.40	0.29	88
2	4	1	2	**38.58**	2.99	0.23	123
1	4	1	2	37.99	3.21	0.24	215

This cubic interpolation is depicted at the top left of Figure 4 and contains hard artifacts at the windows on the left-hand side. The results for our algorithms with two decomposition levels are as follows:

- Algorithm (A) with $c = 1.0$: PSNR $= 31.61$, $\text{err}_2 = 6.69$, $\text{err}_1 = 1.36$.
- Algorithm (B) with $\lambda = 0.5$ and $\lambda_{11} = 8$: PSNR $= 34.08$, $\text{err}_2 = 5.03$, $\text{err}_1 = 0.93$.
- Algorithm (C) with $\sigma = 0.5$, $\lambda = 0.5$, and $\lambda_{11} = 8$: PSNR $= 33.98$, $\text{err}_2 = 5.09$, $\text{err}_1 = 0.97$.

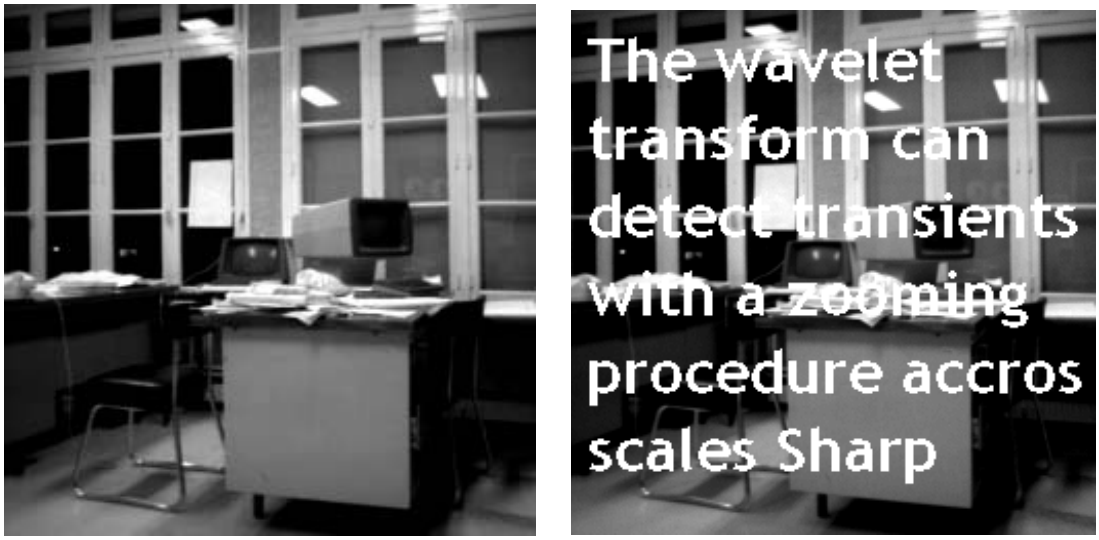

Figure 3. *Original image of the second example and its degraded version.*

- Algorithm (D) with $\sigma = 1$, $\tau = 1$, and $\alpha = 2$: PSNR $= 31.56$, $\text{err}_2 = 6.73$, $\text{err}_1 = 1.27$.
- Algorithm (E) with $\sigma = 1$, $\tau = 1$, and $\alpha = 2$: PSNR $= 31.36$, $\text{err}_2 = 6.89$, $\text{err}_1 = 1.26$.

Algorithms (B) and (C) perform best. The PSNR is approximately 2 dB higher than in the other three algorithms. While Algorithms (A), (D), and (E) produce similar artifacts, especially at the windows, these errors do not appear if we apply Algorithms (B) and (C). This is illustrated in Figure 4 and in the zoomed images in Figure 5.

In our *third example*, we consider the image at the top left of Figure 6. For this image cubic interpolation yields very good results (PSNR $= 33.62$); see top right of Figure 6. Starting with this image as an initial guess and using small parameters ($c = \lambda = \lambda_{11} = 0.05$), we can achieve a PSNR of around 33.8 by applying Algorithms (A)–(C). Visual differences from the image obtained by cubic interpolation are hard to find. For Algorithms (D) and (E) with the original image as initial guess, two decomposition levels, and parameters $\sigma = 1$, $\tau = 1$, and $\alpha = 10$, we obtain the PSNR $= 34.25$, $\text{err}_2 = 4.93$, $\text{err}_1 = 0.98$ after 86 iterations and the PSNR $= 34.21$, $\text{err}_2 = 4.96$, $\text{err}_1 = 1.00$ after 249 iterations, respectively. As shown at the bottom of Figure 6 there are visual differences at long edges.

Figure 4. *Interpolation of the image in Figure 3. Top left: cubic interpolation by the MATLAB procedure "griddata" ($PSNR = 30.18$, $\mathrm{err}_2 = 7.89$, $\mathrm{err}_1 = 1.51$). Top right: Algorithm* (A) *($PSNR = 31.61$, $\mathrm{err}_2 = 6.69$, $\mathrm{err}_1 = 1.36$). Bottom left: Algorithm* (B) *($PSNR = 34.08$, $\mathrm{err}_2 = 5.03$, $\mathrm{err}_1 = 0.93$). Bottom right: Algorithm* (C) *($PSNR = 33.98$, $\mathrm{err}_2 = 5.09$, $\mathrm{err}_1 = 0.97$). The images at the top contain artifacts, in particular at the left window side. The algorithms at the bottom show a better performance and do not introduce these artifacts.*

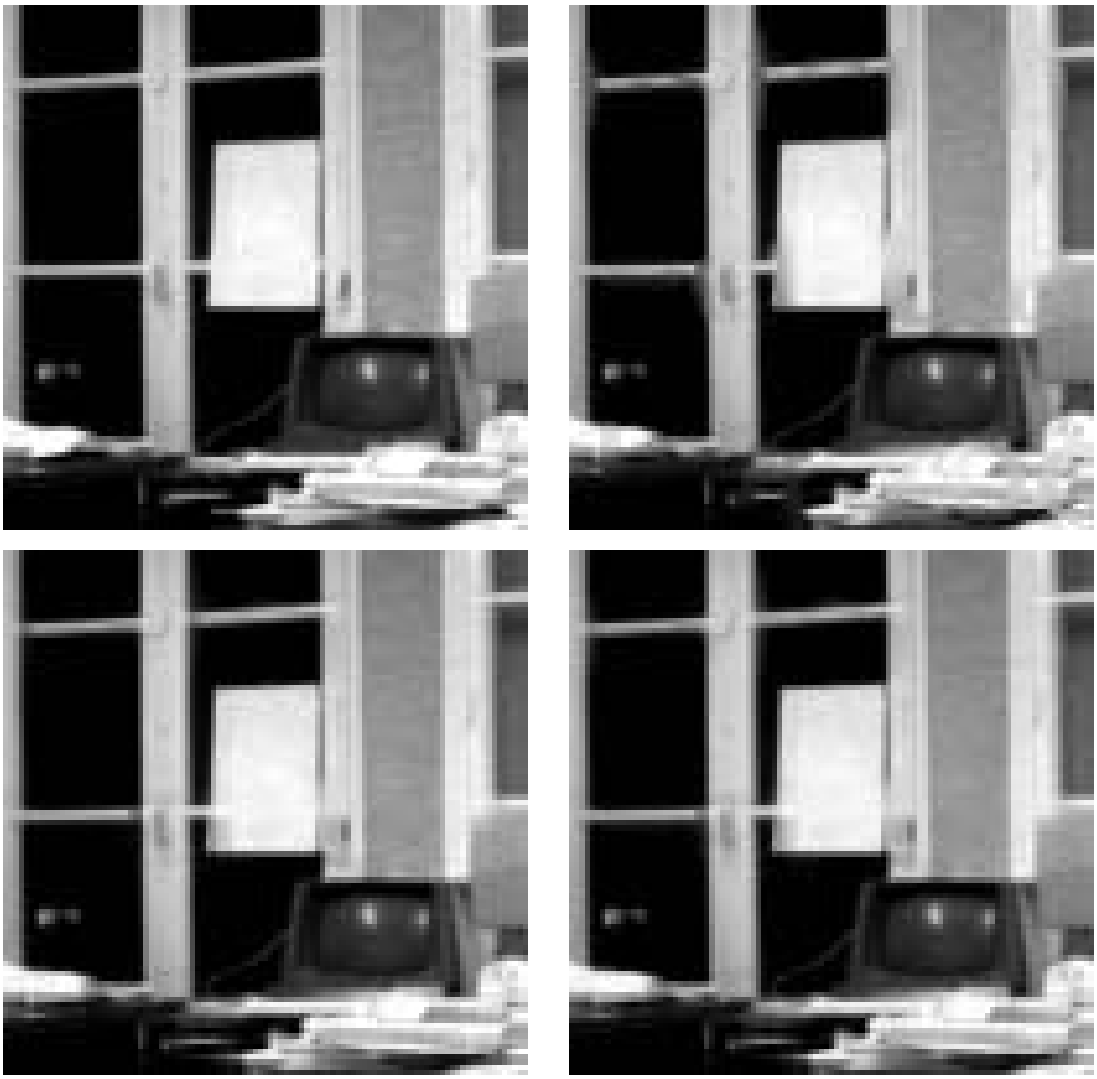

Figure 5. *Details of the interpolated images of Figure 4. Top left: original image. Top right: Algorithm* (A) *(*$PSNR = 31.61$, $\mathrm{err}_2 = 6.69$, $\mathrm{err}_1 = 1.36$*). Bottom left: Algorithm* (B) *(*$PSNR = 34.08$, $\mathrm{err}_2 = 5.03$, $\mathrm{err}_1 = 0.93$*). Bottom right: Algorithm* (C) *(*$PSNR = 33.98$, $\mathrm{err}_2 = 5.09$, $\mathrm{err}_1 = 0.97$*). In contrast to the top right image, the images at the bottom do not have high errors at the horizontal and vertical window lines.*

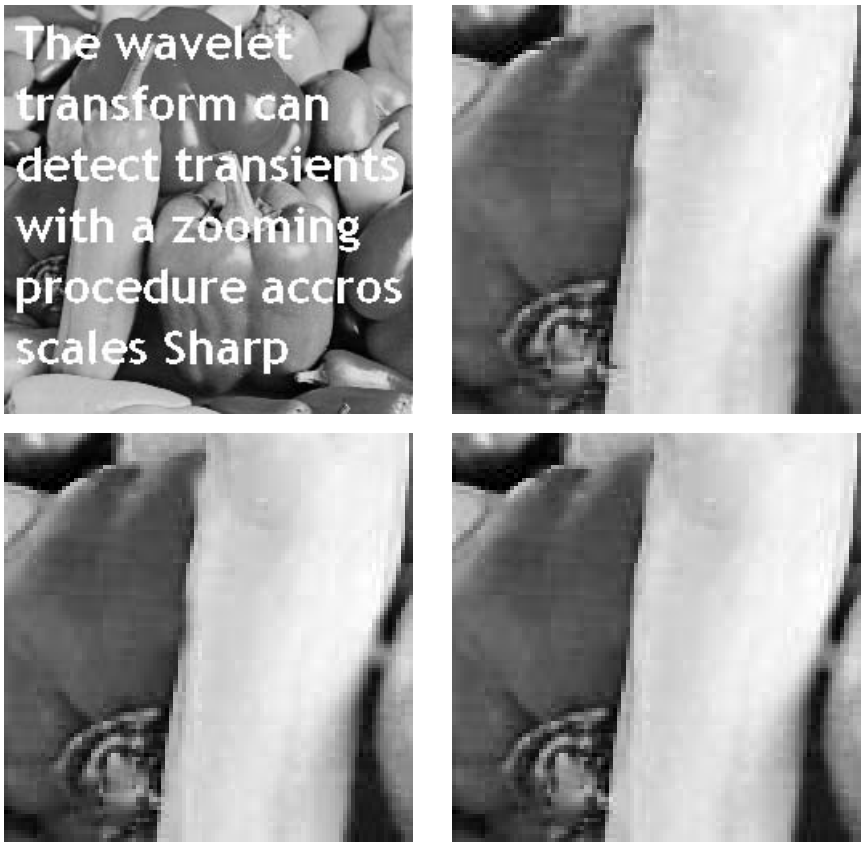

Figure 6. *Interpolation results for the "peppers" image. Top left: degraded image. Top right: cubic interpolation. Bottom left: Algorithm* (D)*. Bottom right: Algorithm* (E)*. Algorithms* (D) *and* (E) *improve the quality at long edges.*

REFERENCES

[1] J.-P. Aubin, *Optima and Equilibria: An Introduction to Nonlinear Analysis*, 2nd ed., Springer, Berlin, Heidelberg, New York, 2003.

[2] B. Berkels, M. Burger, M. Droske, O. Nemitz, and M. Rumpf, *Cartoon extraction based on anisotropic image classification*, in Vision, Modeling, and Visualization 2006, L. Kobbelt, T. Kuhlen, T. Aach, and R. Westermann, eds., IOS Press, Amsterdam, 2006, pp. 293–300.

[3] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, *Image inpainting*, in Proceedings of SIGGRAPH, 2000, pp. 417–424.

[4] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher, *Simultaneous structure and texture inpainting*, IEEE Trans. Image Process., 12 (2003), pp. 882–889.

[5] F. E. Browder and W. V. Petryshyn, *The solution by iteration of nonlinear functional equations in Banach spaces*, Bull. Amer. Math. Soc., 72 (1966), pp. 571–575.

[6] J.-F. Cai, R. H. Chan, and Z. Shen, *A framelet-based image inpainting algorithm*, Appl. Comput. Harmon. Anal., 24 (2008), pp. 131–149.

[7] 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.

[8] P. L. Combettes and V. R. Wajs, *Signal recovery by proximal forward-backward splitting*, Multiscale Model. Simul., 4 (2005), pp. 1168–1200.

[9] I. Daubechies, M. Defrise, and C. De Mol, *An iterative thresholding algorithm for linear inverse problems with a sparsity constraint*, Commun. Pure Appl. Anal., 51 (2004), pp. 1413–1541.
[10] P. Davis, *Circulant Matrices*, John Wiley and Sons, New York, 1979.
[11] M. Elad, J.-L. Starck, P. Querre, and D. L. Donoho, *Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA)*, Appl. Comput. Harmon. Anal., 19 (2005), pp. 340–358.
[12] S. Esedoglu and S. Osher, *Decomposition of images by anisotropic Rudin-Osher-Fatemi model*, Commun. Pure Appl. Anal., 57 (2004), pp. 1609–1626.
[13] M. J. Fadili and J.-L. Starck, *Sparse representations and Bayesian image inpainting*, in Proceedings of the Signal Processing with Adaptive Sparse Structured Representations Workshop (SPARS'05), Vol. 1, Rennes, France, 2005; available online from http://www.greyc.ensicaen.fr/~jfadili/Pub/spars05.pdf.gz
[14] D. Gabay, *Applications of the method of multipliers to variational inequalities*, in Augmented Lagrangian Methods: Applications to the Solution of Boundary Value Problems, M. Fortin and R. Glowinski, eds., North–Holland, Amsterdam, 1983, pp. 299–331.
[15] I. Galić, J. Weickert, M. Welk, A. Bruhn, A. Belyaev, and H.-P. Seidel, *Towards PDE-based image compression*, in Variational, Geometric, and Level Set Methods in Computer Vision, Lecture Notes in Comput. Sci. 3752, Springer, Berlin, 2005, pp. 37–48.
[16] M. Hintermüller and K. Kunisch, *Total bounded variation regularization as a bilaterally constrained optimization problem*, SIAM J. Appl. Math., 64 (2004), pp. 1311–1333.
[17] M. A. Krasnoselskii, *Two observations about the method of successive approximations*, Uspekhi Mat. Nauk, 10 (1955), pp. 123–127 (in Russian).
[18] P. L. Lions and B. Mercier, *Splitting algorithms for the sum of two linear operators*, SIAM J. Numer. Anal., 16 (1979), pp. 964–979.
[19] W. R. Mann, *Mean value methods in iteration*, Proc. Amer. Math. Soc., 4 (1953), pp. 506–510.
[20] G. J. Minty, *Monotone (nonlinear) operators in Hilbert space*, Duke Math. J., 29 (1962), pp. 341–346.
[21] G. B. Passty, *Ergodic convergence to a zero of the sum of monotone operators in Hilbert space*, J. Math. Anal. Appl., 72 (1979), pp. 383–390.
[22] L. I. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Phys. D, 60 (1992), pp. 259–268.
[23] H. Schäfer, *Über die Methode sukzessiver Approximationen*, Jahresber. Deutsch. Math.-Verein., 59 (1957), pp. 131–140.
[24] O. Scherzer and J. Weickert, *Relations between regularization and diffusion filtering*, J. Math. Imaging Vision, 12 (2000), pp. 43–63.
[25] S. Setzer, G. Steidl, and T. Teuber, *Restoration of images with rotated shapes*, Numer. Algorithms, 48 (2008), pp. 49–66.
[26] P. Tseng, *Applications of a splitting algorithm to decomposition in convex programming and variational inequalities*, SIAM J. Control Optim., 29 (1991), pp. 119–138.
[27] J. Weickert, *Anisotropic Diffusion in Image Processing*, Teubner, Stuttgart, 1998.
[28] M. Welk, G. Steidl, and J. Weickert, *Locally analytic schemes: A link between diffusion filtering and wavelet shrinkage*, Appl. Comput. Harmon. Anal., 24 (2008), pp. 195–224.
[29] M. Welk and J. Weickert, *Tensor field interpolation with PDEs*, in Visualization and Processing of Tensor Fields, J. Weickert and H. Hagen, eds., Springer, Berlin, 2006, pp. 315–325.