Comparison of formulations and solution methods for image restoration problems

INSTITUTE OF PHYSICS PUBLISHING INVERSE PROBLEMS

Inverse Problems **17** (2001) 1977–1995 PII: S0266-5611(01)21336-7

Comparison of formulations and solution methods for image restoration problems

Tommi Kärkkäinen, Kirsi Majava and Marko M Mäkelä

University of Jyväskylä, Department of Mathematical Information Technology, PO Box 35 (Agora), FIN-40351 Jyväskylä, Finland

E-mail: tka@mit.jyu.fi, majkir@mit.jyu.fi and makela@mit.jyu.fi

Received 25 January 2001, in final form 21 September 2001
Published 15 November 2001
Online at stacks.iop.org/IP/17/1977

Abstract
Different formulations for image restoration problems are presented, analysed and compared. Two of the three formulations considered are smooth enough to satisfy the assumptions for convergence of ordinary gradient-based optimization methods, such as the conjugate gradient method. For solving the third problem, two general methods of nonsmooth optimization are applied: a (first-order) proximal bundle method and a (second-order) bundle-Newton method. Moreover, a new generalization of active-set methods that have earlier shown high efficiency for BV-regularized image restoration problems is proposed and analysed. Comparison of methods as well as different formulations is completed with numerical experiments.

(Some figures in this article are in colour only in the electronic version)

1. Introduction

The aim of this paper is to present, analyse and compare different formulations and their solution methods for image restoration (denoising) problems. Such formulations form basis in many application areas of inverse problems [1, 2] and computer vision, including, for example, medicine [3]. The central requirement for a good formulation is its restoration capability: the restoration process should preserve certain properties of the true image which may, however, depend on the application. For example, one can either consider the overall quality of the restored image or be only interested in the location of edges for further segmentation purposes. From the solution point of view, important mathematical requirements for a good formulation are convexity, unique solvability and smoothness of a formulation, which make it easier to use and develop efficient solution methods. Finally, a good formulation should be simple: it should have as few unknowns as possible, and the amount of free parameters to be fixed should be minimal.

In this paper, we discuss simple formulations for image restoration problems that are strictly convex. A proper formulation for recovering edges of the true image is nowadays well known and based on bounded variational (BV) regularization [4, 5]. Here we study whether, using generalizations of basic smoothing approaches for the BV-seminorm, one can enhance the recovery of smooth (sub)surfaces contained in the true image. Two of the three formulations considered are smooth enough so that ordinary gradient-based optimization methods, such as the conjugate gradient method, can be applied as a solver. For solving the third problem, different solution methods are compared: first of all, we propose and analyse a generalization of the active-set methods discussed in [6–8]. Moreover, we apply two general methods of nonsmooth, nonconvex optimization: the proximal bundle method [9] and the bundle-Newton (BN) method [10].

After a numerical study of the formulations in section 6, it becomes obvious that the computational results are not optimal by means of the restoration properties. Hence, some form of adaptivity is needed and attempts to this direction have been made, e.g., [4, 11–13]. However, as soon as adaptivity appears, formulations tend to become much more complicated. Adaptive formulations are often nonconvex or nonsmooth (or even both), and in many cases the number of unknowns in these formulations is increased. Therefore, we think that formulations lying between the BV-formulation and existing adaptive formulations deserve further, more comprehensive studies. They must be thoroughly understood in order to be able to develop new, presumably better and more adaptive formulations for image restoration problems, with the aforementioned mathematical properties.

The rest of this paper is organized in the following way. In section 2, three different formulations for the image restoration problem are introduced, and in section 3 the discretized formulations are given. An active-set method is described and its convergence is considered in section 4. Section 5 is devoted to other optimization methods used in numerical experiments. Finally, in section 6, different optimization methods and different formulations are compared through numerical experiments.

2. Description of regularization methods

Let us consider the two-dimensional image restoration problem in a general form:

$$\min_{u \in H_0^1(\Omega)} \varphi(u) + g \int_\Omega \psi(\nabla u)\,\mathrm{d}x, \tag{1}$$

where $\Omega \subset \mathcal{R}^2$ is a rectangle, $\varphi(u) = \frac{1}{2}\int_\Omega |u - z|^2\,\mathrm{d}x + \frac{\mu}{2}\int_\Omega |\nabla u|^2\,\mathrm{d}x$, and z is the noisy data. Further, $g > 0$ is a regularization parameter and $\psi(u)$ is a convex regularization function. Here the term $\frac{\mu}{2}\int_\Omega |\nabla u|^2\,\mathrm{d}x$ for $0 < \mu \ll g$ ensures coercivity of $\varphi(u)$ in $H_0^1(\Omega)$ and yields the unique solvability of problem (1) in this space [6–8]. In the discrete case also $\mu = 0$ is admissible.

Reconstructing a smoothed image u from the noisy data with sharp edges remaining can be done by using the nonsmooth regularization function [4, 5]

$$\psi_{BV}(t) = |t|. \tag{2}$$

Hence, the term $\int_\Omega |\nabla u|\,\mathrm{d}x$ denotes the BV seminorm [14] which does not penalize discontinuities in u, thus allowing one to recover sharp edges of the original image. However, as the theoretical analysis in [15, 16] and a substantial amount of numerical experiments (e.g., [4, 6, 8, 12]) show, the result due to the BV-formulation consists of a staircase-like structure, which is not optimal for images with smooth subsurfaces. On the other hand, the purely quadratic, smooth regularization function

$$\psi_{\text{smooth}}(t) = \tfrac{1}{2}t^2, \tag{3}$$

Table 1. Smoothness of different regularization functions.

Function	Smoothness of derivative	Lips. constant
$\psi_s, s = 1$	Discontinuous	
$\psi_s, 1 < s < 2$	Hölder cont.	
$\psi_s, s = 2$	Lips. cont.	1
ψ_δ	Lips. cont.	$1/\delta$
ψ_ε	Lips. cont.	$1/\sqrt{\varepsilon}$

recovers smooth surfaces well, but smears out sharp edges of the image.

In order to generalize the above approaches, we first study the following regularization function [2, 11]:

$$\psi_s(t) = \frac{1}{s}|t|^s, \qquad 1 \leqslant s \leqslant 2. \tag{4}$$

It is clear that by using a value $s > 1$, it is theoretically impossible (regularity of the solution) to reconstruct edges [4]. Another way to try to combine the restoration properties of regularizations (2) and (3) is to consider the strictly convex and differentiable regularization function based on inf-convolution [4]

$$\psi_\delta(t) = \begin{cases} |t| - \dfrac{\delta}{2}, & \text{for} \quad |t| > \delta, \\ \dfrac{1}{2\delta}t^2, & \text{for} \quad |t| \leqslant \delta, \end{cases} \tag{5}$$

for $\delta > 0$. This approach can be considered a multiobjective optimization problem with objective functions (2) and (3), which are combined using the weighting method [17]. The third formulation considered comes from the fact that in many occasions the nonsmooth BV-term $\int_\Omega |\nabla u|\,\mathrm{d}x$ is replaced with a term $\int_\Omega \sqrt{|\nabla u|^2 + \varepsilon}\,\mathrm{d}x$, for small positive ε, to artificially smooth the problem [5, 12, 18]. However, if we allow larger values of ε, the strictly convex and differentiable regularization function [19]

$$\psi_\varepsilon(t) = \sqrt{t^2 + \varepsilon} \tag{6}$$

has indeed the same kind of behaviour as $\psi_\delta(t)$. Notice that in [20], $\psi_\varepsilon(t)$ with $\varepsilon = 1$ was used in regularization.

The close resemblance of the regularization functions ψ_s and ψ_δ is illustrated in figure 1. On the left, the function $\psi_s'(t) = \mathrm{sign}(t)\,|t|^{s-1}$ is plotted in $[-3, 3]$, for different values of s. On the right, the function

$$\psi_\delta'(t) = \begin{cases} \mathrm{sign}(t), & \text{for} \quad |t| > \delta, \\ t/\delta, & \text{for} \quad |t| \leqslant \delta, \end{cases}$$

is plotted in $[-7, 7]$, for $\delta = 1, \ldots, 6$. Figure 1 shows that there exists a transient region around zero where the s- and δ-regularization are close to each other (by means of scaling). In figure 2, the function $\psi_\varepsilon'(t) = t/\sqrt{t^2 + \varepsilon}$ is plotted in $[-7, 7]$, for different values of ε. Figures 1 and 2 show that $\psi_\delta'(t)$ is actually a piece-wise linear approximation of $\psi_\varepsilon'(t)$. Table 1 shows the smoothness of regularization functions. In the second column of the table, smoothness of the derivative of the regularization function is given. In the last column, the Lipschitz constant (at point $u = 0$) is given.

The restoration properties of the proposed formulations are compared through numerical experiments in section 6.

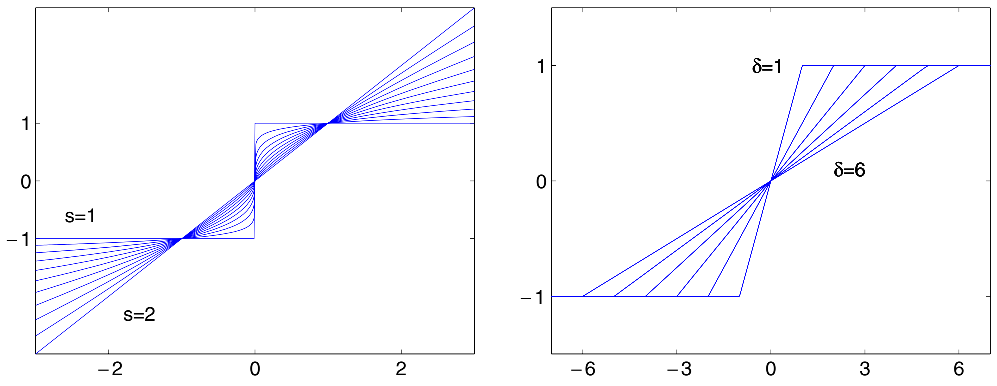

Figure 1. Left: $\psi_s'(t)$ for $1 \leqslant s \leqslant 2$, right: $\psi_\delta'(t)$ for $\delta = 1, \ldots, 6$.

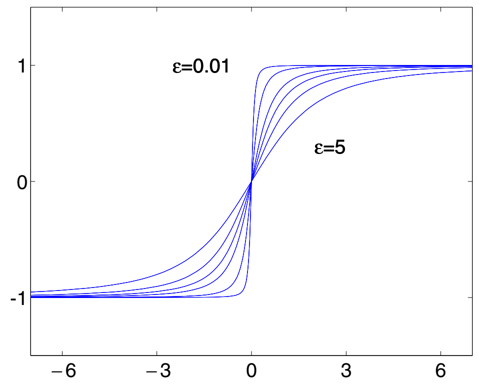

Figure 2. $\psi_\varepsilon'(t)$ for $0.01 \leqslant \varepsilon \leqslant 5$.

3. Discretized formulations and sensitivity analysis

Consider the discretization of problem (1) with the regularization functionals ψ_s, ψ_δ and ψ_ε via the finite-difference method. Then, n_1 and n_2 are the numbers of discretization points in the x- and the y-directions, $H_0^1(\Omega) = \mathcal{R}^{n_1 \times n_2}$ supplied with the Euclidean norm, and $u, z, \lambda_1, \lambda_2$ denote vectors in $\mathcal{R}^{n_1 \times n_2}$, whose components correspond to the values of the unknown functions in the equidistant discretization points in Ω. Notations of u, z are not changed, because only discretized problems are considered in the rest of the paper. We denote by $|\cdot|$ the Euclidean norm of a scalar or vector and define the set of indices as $N = \{1, 2, \ldots, n_1 n_2\}$.

The discrete approximation of $\varphi(u)$ is then defined as

$$\varphi_N(u) = \frac{1}{2}(u - z)^T(u - z) + \frac{\mu}{2} u^T K u,$$

where K is the central difference approximation (five-point scheme) of $-\triangle$. Moreover, $\nabla \varphi_N(u) = u - z + \mu K u$. Let $\vec{D} = \begin{bmatrix} D_1 \\ D_2 \end{bmatrix}$ and $\vec{\lambda} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix}$, where D_1, D_2 are the backward difference approximations of the x- and y-directional derivatives. This choice for discretizing the gradient is, firstly, based on strict locality for representing especially the jumps, and, secondly, on invertibility of the operator in the one-dimensional case (cf the convergence proof of AS in section 4.2).

The discrete approximation of problem (1) with ψ_s is then defined as

$$\min_{u\in\mathcal{R}^{n_1\times n_2}} \mathcal{J}_s(u) = \varphi_N(u) + \frac{g}{s}|\vec{D}u|_s, \tag{7}$$

where

$$|\vec{D}u|_s = \sum_{i=1}^{n_1 n_2} |(\vec{D}u)_i|^s.$$

For all $\mu \geqslant 0$, $\mathcal{J}_s(u)$ is strictly convex, and, for $s > 1$, it is also differentiable in $\mathcal{R}^{n_1\times n_2}$. In the latter case, the two identities

$$\nabla\varphi_N(u^*) + g\vec{D}^T\vec{\lambda}_s^* = 0, \tag{8}$$

$$(\lambda_{s,l}^*)_i = \begin{cases} \dfrac{(D_l u^*)_i}{|(\vec{D}u^*)_i|^{2-s}}, & \text{for} \quad (\vec{D}u^*)_i \neq 0, \\ 0, & \text{for} \quad (\vec{D}u^*)_i = 0, \end{cases} \tag{9}$$

for $l = 1, 2$, constitute the necessary and sufficient optimality condition for the solution u^* of problem (7). For $s = 1$, $\mathcal{J}_s(u)$ is nondifferentiable, and a subgradient $\xi \in \partial\mathcal{J}_s(u)$ is defined by

$$\xi = \nabla\varphi_N(u) + g\vec{D}^T\vec{\lambda}_s,$$

where $|(\vec{\lambda}_s)_i| \leqslant 1$ for $(\vec{D}u)_i = 0$. Now $0 \in \partial\mathcal{J}_s(u)$ is the necessary and sufficient optimality condition for problem (7) for $s = 1$.

The finite difference approximation of problem (1) with ψ_δ is defined as

$$\min_{u\in R^{n_1\times n_2}} \mathcal{J}_\delta(u) = \varphi_N(u) + g\left(\sum_{j\in J}\left(|(\vec{D}u)_j| - \frac{\delta}{2}\right) + \frac{1}{2\delta}\sum_{i\in I}|(\vec{D}u)_i|^2\right), \tag{10}$$

where the index sets are

$$J = \{j \in N : |(\vec{D}u)_j| > \delta\}, \qquad I = \{i \in N : |(\vec{D}u)_i| \leqslant \delta\}.$$

The necessary and sufficient optimality condition for problem (10) is given by (8), where $\vec{\lambda}_s^*$ is replaced with

$$\vec{\lambda}_\delta^* = \begin{bmatrix} P_J\lambda_1^* + \frac{1}{\delta}P_I\gamma_1^* \\ P_J\lambda_2^* + \frac{1}{\delta}P_I\gamma_2^* \end{bmatrix}, \quad \text{where} \quad (\lambda_l^*)_i = \frac{(D_l u^*)_i}{|(\vec{D}u^*)_i|}, \quad (\gamma_l^*)_i = (D_l u^*)_i,$$

for $l = 1, 2$, and P_S denotes projection onto an arbitrary index set S

$$(P_S u)_i = \begin{cases} u_i, & \text{for} \quad i \in S, \\ 0, & \text{for} \quad i \in N \setminus S. \end{cases}$$

Finally, the finite difference approximation of problem (1) with ψ_ε is defined as

$$\min_{u\in R^{n_1\times n_2}} \mathcal{J}_\varepsilon(u) = \varphi_N(u) + g\sum_{i=1}^{n_1 n_2}\left((\vec{D}u)_i^2 + \varepsilon\right)^{1/2}, \tag{11}$$

and the necessary and sufficient optimality condition for the problem is also given by (8), where $\vec{\lambda}_s^*$ is substituted by

$$(\lambda_{\varepsilon,l}^*)_i = \left((\vec{D}u^*)_i^2 + \varepsilon\right)^{-1/2}(D_l u^*)_i,$$

for $l = 1, 2$. In what follows, we refer to the three discretized formulations, (7), (10), and (11), as s-, δ- and ε-formulations, respectively.

Remark 3.1. In report [21], we considered ψ_δ in a slightly different form

$$\psi_\delta(t) = \begin{cases} |t|, & \text{for} \quad |t| > \delta, \\ \dfrac{1}{\delta}t^2, & \text{for} \quad |t| \leqslant \delta, \end{cases} \tag{12}$$

which is nonconvex and nondifferentiable. Numerical results obtained using the two regularizations, (5) and (12), are, however, practically the same.

4. Active-set method: description and convergence analysis

In this section, we describe an active-set method for solving the s-defined problem (7). The use of active-set methods for solving the BV-regularized image restoration problem (7) for $s = 1$ was originally proposed in [6]. In our previous work [7, 8], we developed efficient active-set algorithms for solving the BV-regularized problem in one- and two-dimensional cases. The main idea in these algorithms is to transform the original, nonsmooth and unconstrained optimization problem, using a Lagrange smoothing [22, 23] and an active-set approach [22], into a sequence of constrained problems of the form

$$\min_{u \in R^{n_1 \times n_2}} \varphi_N(u) + \frac{g}{s}|\vec{D}_{I^k} u|_s \qquad \text{subject to} \quad \vec{D}_{J^k} u = 0, \tag{13}$$

where I^k and J^k are disjoint index sets such that $N = I^k \cup J^k$ and $\vec{D}_S := P_S \vec{D}$. Problem (13) is smoother than the original problem (7) because the essential part concerning regularity of the functional appears in the constraint $\vec{D}_J u = 0$. Hence, since for $s > 1$ problem (7) is differentiable, the constraint $\vec{D}_J u = 0$ compensates the 'nearly nonsmooth' components. Next, we describe a generalization of the BV-based active-set method for $1 < s \leqslant 2$.

4.1. Description of the algorithm

Let $N = I \cup J$ be a disjoint composition. Consider the following constrained optimization problem:

$$\min_{u \in R^{n_1 \times n_2}} \mathcal{J}_N(u) = f_N(u) + \frac{g}{s}|\vec{D}_I u|_s \qquad \text{subject to} \quad \vec{D}_J u = 0, \tag{14}$$

where $f_N : \mathcal{R}^{n_1 \times n_2} \to \mathcal{R}$ is a convex, continuously differentiable and coercive functional satisfying

$$\left(\nabla f_N(u) - \nabla f_N(u_0)\right)^T (u - u_0) \geqslant \sigma (u - u_0)^T (u - u_0) \tag{15}$$

for all $u, u_0 \in \mathcal{R}^{n_1 \times n_2}$ and $\sigma > 0$. In the noise reduction problem, we have $f_N(u) = \varphi_N(u)$, which clearly satisfies (15). Now, the following result is obvious.

Proposition 4.1. *For all* $1 < s \leqslant 2$, *the minimization problem (14) has a unique solution* $\underline{u}$ *and there exists* $\underline{\vec{\lambda}}$ *satisfying*

$$(\underline{\vec{\lambda}})_i^T (\vec{D}\underline{u})_i = |(\vec{D}\underline{u})_i|^s \qquad \text{and} \qquad |(\underline{\vec{\lambda}})_i| \leqslant |(\vec{D}\underline{u})_i|^{s-1}, \qquad \textit{for all} \quad i \in I, \tag{16}$$

and

$$\nabla \varphi_N(\underline{u}) + g\vec{D}^T \underline{\vec{\lambda}} = \nabla \varphi_N(\underline{u}) + g\vec{D}^T (P_I \underline{\vec{\lambda}} + P_J \underline{\vec{\lambda}}) = 0. \tag{17}$$

Notice that on $J\underline{\vec{\lambda}}$ is the Lagrange multiplier for the equality constraint in (14). The basic active-set algorithm for solving problem (7) reads as follows:

Algorithm 4.1 (Basic active-set algorithm).

Step 1. Choose $1 < s \leqslant 2$. Initialize $u^0 \in \mathcal{R}^{n_1 \times n_2}$, $\vec{\lambda}^0 \in \mathcal{R}^{n_1 \times n_2 \times 2}$, choose $c > 0$ and set $k = 0$.

Step 2. Determine the index sets
$J = J^k = \{j \in N : |(\vec{\lambda}^k + c\vec{D}u^k)_j| \leqslant |(\vec{D}u^k)_j|^{s-1}\}$,
$I = I^k = \{i \in N : |(\vec{\lambda}^k + c\vec{D}u^k)_i| > |(\vec{D}u^k)_i|^{s-1}\}$.
If $k > 1$ and $J^k = J^{k-1}$, then STOP; the solution is u^k.

Step 3. Let $(\underline{u}, \underline{\vec{\lambda}})$ be the solution for (14) as defined in proposition 4.1.

Step 4. Set $u^{k+1} = \underline{u}$, $\vec{\lambda}^{k+1} = \underline{\vec{\lambda}}$, and $k = k + 1$. Go to step 2.

4.2. Convergence analysis

Let us next consider the convergence of algorithm 4.1. The first step towards a convergence analysis of an active-set method is to study the behaviour of the index sets J^k and I^k between two iteration steps, k and $k+1$, in the algorithm [7]. To do this, we split the set J^k as follows:

$$J^k = \{j \in N : |(\vec{\lambda}^k + c\vec{D}u^k)_j| \leqslant |(\vec{D}u^k)_j|^{s-1}\} = J^k_{(1)} \cup J^k_{(2)},$$

where

$$J^k_{(1)} = \{j \in J^k : |\vec{\lambda}^{k+1}_j| \leqslant |(\vec{D}u^{k+1})_j|^{s-1}\} \quad \text{and} \quad J^k_{(2)} = \{j \in J^k : |\vec{\lambda}^{k+1}_j| > |(\vec{D}u^{k+1})_j|^{s-1}\}.$$

Similarly, the set I^k can be split as

$$I^k = \{i \in N : |(\vec{\lambda}^k + c\vec{D}u^k)_i| > |(\vec{D}u^k)_i|^{s-1}\} = I^k_{(1)} \cup I^k_{(2)},$$

where

$$I^k_{(1)} = \{i \in I^k : (\vec{D}u^{k+1})_i = 0\} \qquad \text{and} \qquad I^k_{(2)} = \{i \in I^k : (\vec{D}u^{k+1})_i \neq 0\}.$$

By the definition of I^{k+1} and J^{k+1}, a straightforward calculation shows that we obtain

$$J^{k+1} = J^k_{(1)} \cup I^k_{(1)} \qquad \text{and} \qquad I^{k+1} = I^k_{(2)} \cup J^k_{(2)}. \tag{18}$$

Notice that since for all $j \in J^k$, $(Du^{k+1})_j = 0$ due to (14), we actually have

$$J^k_{(1)} = \{j \in J^k : \lambda^{k+1}_j = 0\} \qquad \text{and} \qquad J^k_{(2)} = \{j \in J^k : \lambda^{k+1}_j \neq 0\}$$

for $s > 1$.

Lemma 4.1. *If $u^k \neq u^{k+1}$, then $\mathcal{J}_N(u^k) > \mathcal{J}_N(u^{k+1})$ for all $k \geqslant 1$.*

Proof. Using (18), we know that $J^k = J^{k-1}_{(1)} \cup I^{k-1}_{(1)}$ and on both sets $(\vec{D}u^k)_j = 0$, so that $u^k \in \mathrm{Ker}(\vec{D}_{J^k})$. From the optimality condition for (14), it follows that

$$\left(\nabla\varphi_N(u^{k+1})\right)^T\left(u^k - u^{k+1}\right) + \frac{g}{s}|\vec{D}_{I^k}u^k|_s - \frac{g}{s}|\vec{D}_{I^k}u^{k+1}|_s \geqslant 0. \tag{19}$$

Further, it follows from assumption (15) that

$$\varphi_N(u) - \varphi_N(u_0) - (\nabla\varphi_N(u_0))^T(u - u_0) \geqslant \frac{\sigma}{2}|u - u_0|^2. \tag{20}$$

Using equality $|\vec{D}_{I^{k-1}}u^k|_s = |\vec{D}_{I^k}u^k|_s$, the coercivity condition (20), and (19), we obtain

$$\begin{aligned}
\mathcal{J}_N(u^k) - \mathcal{J}_N(u^{k+1}) &= \varphi_N(u^k) + \frac{g}{s}|\vec{D}_{I^{k-1}}u^k|_s - \varphi_N(u^{k+1}) - \frac{g}{s}|\vec{D}_{I^k}u^{k+1}|_s \\
&= \varphi_N(u^k) - \varphi_N(u^{k+1}) - \left(\nabla\varphi_N(u^{k+1})\right)^T\left(u^k - u^{k+1}\right) \\
&\quad + \left(\nabla\varphi_N(u^{k+1})\right)^T\left(u^k - u^{k+1}\right) + \frac{g}{s}|\vec{D}_{I^{k-1}}u^k|_s - \frac{g}{s}|\vec{D}_{I^k}u^{k+1}|_s \\
&= \varphi_N(u^k) - \varphi_N(u^{k+1}) - \left(\nabla\varphi_N(u^{k+1})\right)^T\left(u^k - u^{k+1}\right) \\
&\quad + \left(\nabla\varphi_N(u^{k+1})\right)^T\left(u^k - u^{k+1}\right) + \frac{g}{s}|\vec{D}_{I^k}u^k|_s - \frac{g}{s}|\vec{D}_{I^k}u^{k+1}|_s \\
&\geqslant \frac{\sigma}{2}|u^k - u^{k+1}|^2 > 0, \qquad \text{when} \quad u^k \neq u^{k+1}.
\end{aligned}$$

□

Theorem 4.1. *Assume that $J^k \neq J^{k+1}$ implies $u^k \neq u^{k+1}$. Then algorithm 4.1 converges in finitely many steps.*

Proof. From (18) we see that, if $J^k = J^{k+1}$, for some $k \geqslant 1$, then the pair $(u^{k+1}, \vec{\lambda}^{k+1})$ satisfies the optimality condition (8), (9). Thus, $u^{k+1} = u^*$, where u^* is the unique solution for (7). Moreover, lemma 4.1, together with the given assumption, shows that $\mathcal{J}_N(u^k)$ is decreasing as long as $J^k \neq J^{k+1}$. Because there exists only a finite number of possible active sets J^k, we must arrive at $J^k = J^{k+1}$ after a finite number of steps. □

The assumption in theorem 4.1 is valid, at least in the one-dimensional case. This follows from the uniqueness of λ as a function of u in (16), (17), due to the full rank of $D_J u$ on J. This condition is still an open problem in the two-dimensional case, but such a consideration is, however, not needed when using monotone algorithms, as proposed in [7, 8], for which the convergence also for $1 < s \leqslant 2$ directly follows from the monotonicity $J^k \subset J^{k+1}$.

4.3. Implementation of the algorithm in one dimension

Because convergence analysis for the basic (nonmonotone) active-set algorithm is well established only for the one-dimensional case, we next give the implemented active-set algorithm in this setting. Then, n denotes the number of discretization points in $\Omega \subset \mathcal{R}$, D the backward difference approximation of u', and K the central difference approximation of $-u''$. Moreover, the set of indices is given as $N = \{1, 2, \ldots, n\}$.

Convergence analysis in section 4.2 was based on the assumption that problem (14) is solved exactly in each iteration step of an active-set algorithm. However, earlier numerical tests in [6, 8] have shown that it is enough to decrease the cost function by taking only one projected Newton-like step with a line search. Notice that then we also change the stopping criterion of algorithm 4.1. This is due to the fact that the index sets do not necessarily behave as suggested in (18) when the inner iterative problem is not solved exactly. The implemented algorithm reads as follows:

Algorithm 4.2 (AS).

Step 1. Choose $1 < s \leqslant 2$, $c > 0$, $\tau > 0$, tol > 0, and $(u^0, \lambda^0) \in \mathcal{R}^n \times \mathcal{R}^n$. Set $k = 0$.

Step 2. Determine the index sets as

$$J = J^k = \{j \in N : |(\lambda^k + cDu^k)_j| \leqslant |(Du^k)_j|^{s-1}\},$$
$$I = I^k = \{i \in N : |(\lambda^k + cDu^k)_i| > |(Du^k)_i|^{s-1}\}.$$

Step 3. Set $\lambda_i^{k+1} = (Du^k)_i / \max\{\tau, |(Du^k)_i|^{2-s}\}$, for all $i \in I$, and solve u using the projected Newton-like step of [8] from

$$(Id + \mu K)u = z - gD_I^T \lambda^{k+1}, \qquad \text{subject to} \quad D_J u = 0.$$

Step 4. Take $d^k = u - u^k$ and update $u^{k+1} = u^k + td^k$, where $t \in [0, 1]$ is obtained using a line search.

Step 5. If $\left(\mathcal{J}_s(u^{k+1}) - \mathcal{J}_s(u^k)\right) / \max\left(\tau, \mathcal{J}_s(u^k)\right) <$ tol, then STOP; the solution is u^{k+1}. Else, update λ^{k+1} on J according to the linear optimality condition

$$u^{k+1} - z + \mu K u^{k+1} + gD^T(P_I\lambda^{k+1} + P_J\lambda^{k+1}) = 0,$$

set $k = k + 1$, and go to step 2.

The step size t is searched using the Armijo rule [22]. Moreover, in the actual realization we use the two-step process of [8]: first, we execute algorithm 4.2 using the initial value of c to get the solution u_1. Then, we increase the value of c by taking $c = 10c$ and execute the algorithm again using u_1 as the initial guess. In our experiments, we used the initial value $c = 1$ and the stopping criterion tol $= 10^{-7}$.

5. Other solution methods

In this section, we shortly introduce the optimization methods that will be compared to AS in the numerical experiments in section 6. The stopping criteria for different algorithms are chosen so that the accuracy of the solutions is approximately the same (see [21]).

5.1. Bundle methods

Bundle methods can be used as black-box methods for solving general nonsmooth optimization problems. This makes them very useful in, for example, testing and comparing such formulations for image restoration problems. The origin of the methods is in the classical cutting plane method [24], where a piecewise linear approximation of the objective function is formed. In bundle methods, a stabilizing quadratic term is added to the polyhedral approximation in order to accumulate second-order information about the curvature of the objective function. In the BN method, the piecewise linear model is replaced by a quadratic model.

The methods assume that at each point, the function value and an arbitrary subgradient of a locally Lipschitz-continuous cost function can be evaluated. For the BN method, it is also assumed that a symmetric approximation of the Hessian matrix can be evaluated. For the convergence of the methods in the nonconvex case, $\mathcal{J}$ is further assumed to be upper semismooth [10, 25].

Proximal bundle method (PB) [9, 25]. The tested proximal bundle algorithm is from the software package NSOLIB (author M M Mäkelä). The code utilizes the subgradient aggregation strategy of [26] to keep the storage requirements bounded, and the safeguarded quadratic interpolation algorithm of [25] to control the size of the search region. For solving the quadratic subproblem, the code employs the quadratic solver QPDF4, which is based on the dual active-set method described in [27].

In our experiments we used such stopping criteria that the desired accuracy of the final solution was five digits for smaller problems and four digits for larger ones.

Bundle-Newton method (BN) [10]. The tested BN algorithm is from the software package UFO (authors L Lukšan and J Vlček). This code also utilizes the subgradient aggregation strategy of [26] and employs the solver ULQDF1 implementing the dual projected gradient method proposed in [28], for solving the quadratic subproblem.

The same stopping criteria were used as in PB. The Hessian matrix was calculated numerically using difference approximations.

5.2. Conjugate gradient method (CG)

We implemented the conjugate gradient algorithm by Fletcher and Reeves with restarts after every n iterations and inexact line search using the strict Wolfe condition [29]. The same algorithm was applied to problem (7) also in the nonsmooth case $s = 1$. The search direction was then formed using a representative ($\lambda_i = 0$ for $(Du)_i = 0$) of the subdifferential. As a stopping criterion, we used $|u^{k+1} - u^k|_\infty < 10^{-8}$.

Note that for the convergence of the CG method, it is assumed that the cost functional is continuously differentiable and its gradient is Lipschitz continuous. Hence, for $s < 2$, problem (7) does not satisfy these requirements. However, numerical experiments in section 6 show that by using a more exact line search, CG converges also for $1 < s < 2$. This is due to the well known fact that discretization regularizes: smoothness problems tend to be more visible only when the discretization becomes fine enough. For solving problem (7), we used the golden section minimization such that the line search becomes more accurate along iterations until the length of the search interval is less than 10^{-10}.

6. Numerical experiments

In this section, a numerical comparison is made of the proposed optimization methods for solving the s-defined image restoration problem (7) in the one-dimensional case. Moreover, the three image restoration formulations, (7), (10), and (11), are compared. A more thorough description of the numerical experiments with other one-dimensional examples can be found in [21]. Here, we consider the following two examples:

Example 6.1. True and noisy signals are given in figure 3. The noisy signal z is generated by adding uniformly distributed random noise from the interval $[-0.2, 0.2]$ to the true signal.

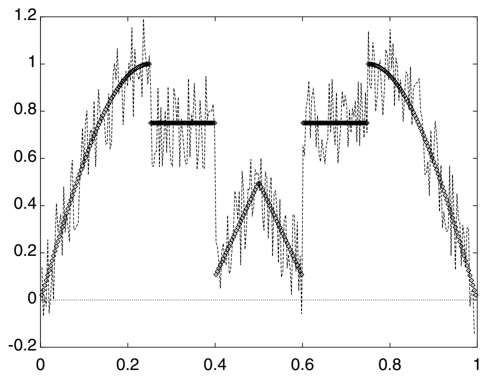

Figure 3. True (thick curve) and noisy (dashed curve) signals for $n = 300$ in example 6.1.

Example 6.2. True and noisy images are given in figure 4. The noisy image z is generated by adding normally distributed random noise from $\mathcal{N}(0, 1)$, scaled with the noise level η, to the true image.

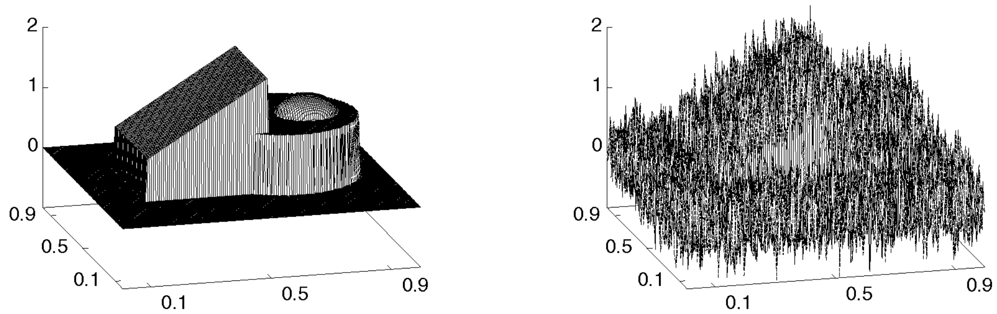

Figure 4. True (left) and noisy (right) images for $n = 300 \times 300$ in example 6.2 for $\eta = 0.3$.

All experiments were performed on an HP9000/J280 workstation (180 MHz, PA8000 CPU). The tested algorithms were implemented using Fortran 77 with double-precision arithmetic. The one-dimensional examples were computed in the standard interval $\Omega = (0, 1)$ and the two-dimensional ones in $\Omega = (0, 1) \times (0, 1)$. The values of the regularization parameters were chosen as $\mu = 10^{-10}$ and $g = 0.001$ in example 6.1, $g = 0.002$ in example 6.2. As the initial guess we used $u^0 = z$, and for λ^0 in AS we chose $\lambda_i^0 = (Du^0)_i / \max(\tau, |(Du^0)_i|^{2-s})$, for all $i = 1, \ldots, n$.

Remark 6.1. A homogeneous Dirichlet boundary condition in the given formulations was chosen for simplicity. To obtain more practical algorithms less sensitive to this assumption, we modify the basic regularizations by leaving out those backward differences which contain boundary points in their local stencil, so that [8]

$$(D_1 u)_{i,j} = \frac{u_{i,j} - u_{i-1,j}}{h_x}, \qquad \text{for} \quad i = 2, \ldots, n_1, \quad j = 1, \ldots, n_2,$$

and

$$(D_2 u)_{i,j} = \frac{u_{i,j} - u_{i,j-1}}{h_y}, \qquad \text{for} \quad i = 1, \ldots, n_1, \quad j = 2, \ldots, n_2,$$

where $h_x = \frac{m_x(\Omega)}{n_1+1}$ and $h_y = \frac{m_y(\Omega)}{n_2+1}$ and $m_x(\Omega)$, $m_y(\Omega)$ denote the length of Ω in the x- and the y-directions.

Comparison of different algorithms for solving the s-defined problem (7) in the one-dimensional case is given in tables 2 and 3. In table 2, the size of the problem and the value of s are given in the first column. Then, the results of different algorithms are given in their own blocks as follows: in the first column, the final value of the cost functional $\mathcal{J}(u^*) = \mathcal{J}_s(u^*)$ is given. In the second column, we have the elapsed CPU time (in seconds), and in the third column the number of iterations it. In the last column, we give the average error between the obtained result u^* and the true signal z^* defined as

$$e(u^*) := e(u^*, z^*) := \sqrt{\frac{1}{n} \sum_{i=1}^{n} (u^* - z^*)_i^2}.$$

For clarity, the error $e(u^*)$ is multiplied by 100 in all tables. Notice that in tables 4 and 5, the average error between the results of formulations (10) and (11) is also given. Finally, in table 3, a new column $|J^*|$ denoting the number of indices in the final active set J^* is added. The results of CG for solving the δ-formulation (10) and ε-formulation (11) are given in tables 4 and 5.

Table 2. Results of bundle methods and CG in example 6.1 for $g = 0.001$.

Method	PB				BN				CG			
s	$\mathcal{J}(u^*)$	CPU	it	$e(u^*)$	$\mathcal{J}(u^*)$	CPU	it	$e(u^*)$	$\mathcal{J}(u^*)$	CPU	it	$e(u^*)$
$n = 100$												
1.0	0.7975	44.11	530	6.86	0.7975	192.23	587	6.86	0.7979	0.76	551	6.83
1.1	0.8856	38.50	567	6.59	0.8856	14.14	73	6.59	0.8858	2.07	636	6.59
1.3	1.0830	4.92	182	6.56	1.0830	2.50	26	6.56	1.0830	1.16	416	6.56
1.5	1.2734	0.53	61	7.37	1.2734	0.93	12	7.37	1.2734	0.26	122	7.37
1.7	1.4388	0.11	36	8.34	1.4387	0.62	8	8.35	1.4387	0.13	66	8.35
2.0	1.6582	0.11	35	9.63	1.6582	0.14	2	9.63	1.6582	0.06	54	9.63
$n = 300$												
1.0	2.7580	613.72	2240	5.02	2.7580	3575.40	1046	5.02	2.7641	10.27	1849	4.89
1.1	3.0581	401.97	1818	5.04	3.0581	339.27	120	5.03	3.0590	55.86	4018	5.01
1.3	3.6007	28.15	263	5.89	3.6007	31.44	22	5.90	3.6007	22.81	1777	5.90
1.5	4.0508	5.39	133	7.05	4.0507	15.91	13	7.09	4.0507	3.48	435	7.09
1.7	4.4431	4.29	115	8.14	4.4430	10.88	9	8.16	4.4430	1.54	230	8.16
2.0	4.9914	2.35	91	9.43	4.9912	2.15	2	9.47	4.9912	0.48	146	9.47

Table 3. Results of AS in example 6.1 for $g = 0.001$.

	$n = 100$					$n = 300$					$n = 1000$				
s	$\mathcal{J}(u^*)$	CPU	it	$e(u^*)$	$\|J^*\|$	$\mathcal{J}(u^*)$	CPU	it	$e(u^*)$	$\|J^*\|$	$\mathcal{J}(u^*)$	CPU	it	$e(u^*)$	$\|J^*\|$
1.0	0.7981	0.01	14	6.87	55	2.7586	0.09	65	5.03	217	9.8950	2.81	610	3.23	894
1.1	0.8857	0.05	35	6.59	15	3.0592	0.78	239	5.03	92	10.9150	16.39	1320	3.53	265
1.3	1.0830	0.06	54	6.56	2	3.6007	1.27	400	5.88	14	12.2288	35.74	3023	5.58	47
1.5	1.2734	0.07	69	7.37	1	4.0507	0.97	337	7.07	2	13.4197	28.35	2586	7.01	18
1.7	1.4387	0.07	65	8.35	1	4.4430	1.13	369	8.15	0	14.5355	28.79	2513	8.18	0
2.0	1.6582	0.04	54	9.63	0	4.9912	1.06	687	9.48	0	16.1892	16.79	3475	9.56	0

Table 4. Results of CG for formulations (10) and (11) in example 6.1 for $g = 0.001$.

Formulation	(10)				(11)				
	δ	CPU	it	$e(u^*)$	ε	CPU	it	$e(u^*)$	$e(u^*_\delta, u^*_\varepsilon)$
$n = 100$	1	0.04	179	6.83	0.5	0.03	178	6.83	1.83×10^{-3}
	5	0.01	55	6.82	10	0.01	67	6.84	3.41×10^{-3}
	10	0.01	26	6.96	60	0.01	25	7.10	5.03×10^{-3}
	15	0.01	25	7.28	100	0.01	26	7.28	7.30×10^{-3}
$n = 300$	1	0.38	678	4.92	0.5	0.40	636	4.90	1.41×10^{-3}
	5	0.17	348	4.59	10	0.22	380	4.62	2.66×10^{-3}
	10	0.10	212	4.46	60	0.12	214	4.52	3.44×10^{-3}
	15	0.05	133	4.52	100	0.09	168	4.58	4.53×10^{-3}

Plots of the results are given in figures 5–12, where the true (one-dimensional) signal is plotted with a dashed curve. Conclusions drawn from the numerical experiments are given in the next five sections.

Table 5. Results of CG for formulations (10) and (11) in example 6.2 for $\eta = 0.3$ and $g = 0.002$.

Formulation	(10)				(11)				
	δ	CPU	it	$e(u^*)$	ε	CPU	it	$e(u^*)$	$e(u^*_\delta, u^*_\varepsilon)$
$n = 100 \times 100$	1	58.02	1 385	7.70	0.2	66.38	1 672	7.72	1.28×10^{-3}
	5	5.18	184	8.13	5	6.40	239	8.15	4.53×10^{-3}
	10	1.70	74	8.96	30	2.00	91	9.02	6.20×10^{-3}
	15	0.92	51	9.84	77	1.55	61	9.84	7.41×10^{-3}
$n = 200 \times 200$	1	5 203.15	17 472	3.65	0.5	4 178.17	14 044	3.67	1.38×10^{-3}
	5	302.37	1 440	4.07	10	324.45	1 455	4.06	2.57×10^{-3}
	10	59.65	349	4.66	50	94.01	526	4.67	3.14×10^{-3}
	15	35.21	231	5.24	120	37.71	245	5.21	3.61×10^{-3}
$n = 300 \times 300$	1	47 485.36	54 581	3.15	0.7	34 345.81	41 010	3.15	1.69×10^{-3}
	5	2 082.86	3 214	3.48	15	2 797.02	4 446	3.52	2.28×10^{-3}
	10	624.91	1 143	3.92	60	741.34	1 379	3.93	2.58×10^{-3}
	15	283.64	595	4.32	140	348.81	726	4.31	2.92×10^{-3}

Table 6. Best g w.r.t. $e(u^*)$ for the s-formulation (7) in example 6.1 for $n = 300$.

s	g	$e(u^*)$
1.0	9×10^{-4}	5.00
1.1	7×10^{-4}	4.87
1.3	4×10^{-4}	4.95
1.5	2×10^{-4}	5.15
1.7	1×10^{-4}	5.40
2.0	6×10^{-5}	5.65

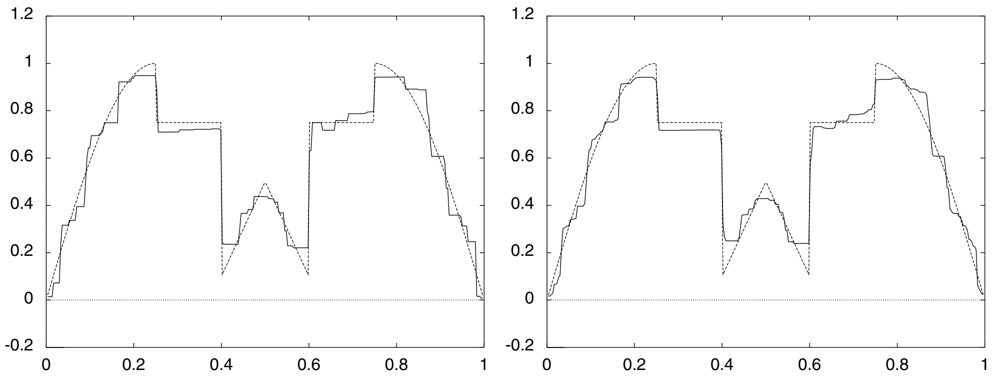

Figure 5. The results of PB are given for the s-formulation (7) in example 6.1, for $n = 300$ and $g = 0.001$. Left: $s = 1$, right: $s = 1.1$.

6.1. Comparison of optimization methods

Here, we present a short comparison of the optimization methods used.

For solving the s-defined image restoration problems in the one-dimensional case, the rate of convergence increased along the regularity of the problem for PB, BN and CG. PB was faster than BN in most of the cases. Despite the theory, CG was able to solve the problems also for $1 < s < 2$ but needed a very accurate stopping criterion and a huge amount of iterations (cf section 5.2). (In table 2, the value of the cost function $\mathcal{J}(u^*)$ remains slightly too large

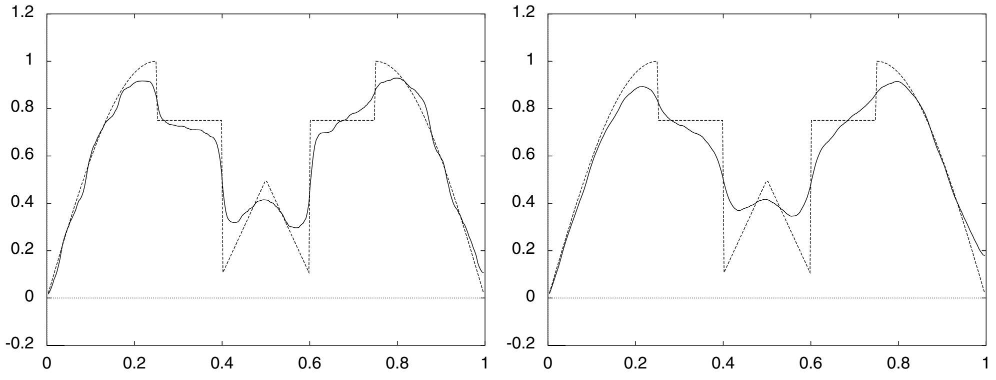

Figure 6. The results of PB are given for the s-formulation (7) in example 6.1, for $n = 300$ and $g = 0.001$. Left: $s = 1.5$, right: $s = 2.0$.

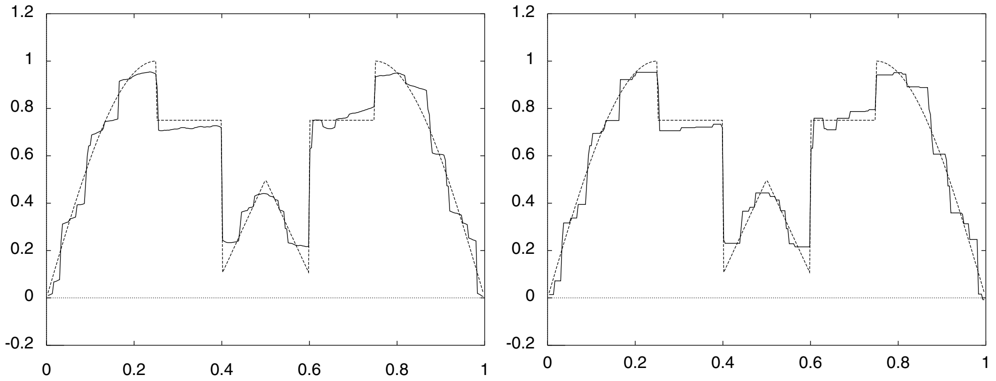

Figure 7. The results are presented in example 6.1 for $n = 300$. On the left, the δ-formulation (10) for $g = 0.001$ and on the right, the s-formulation (7) for the best value of g w.r.t. $e(u^*)$. Left: $\delta = 1$, right: $s = 1.0$.

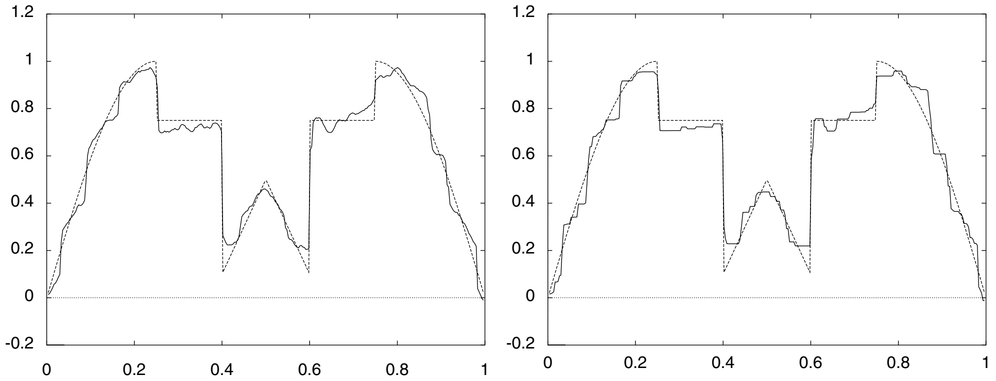

Figure 8. The results are presented in example 6.1 for $n = 300$. On the left, the δ-formulation (10) for $g = 0.001$ and on the right, the s-formulation (7) for the best value of g w.r.t. $e(u^*)$. Left: $\delta = 5$, right: $s = 1.1$.

in the case $s = 1.1$, but after further decreasing the stopping criterion to tol $= 10^{-11}$, CG obtained the same value of the cost function as PB.)

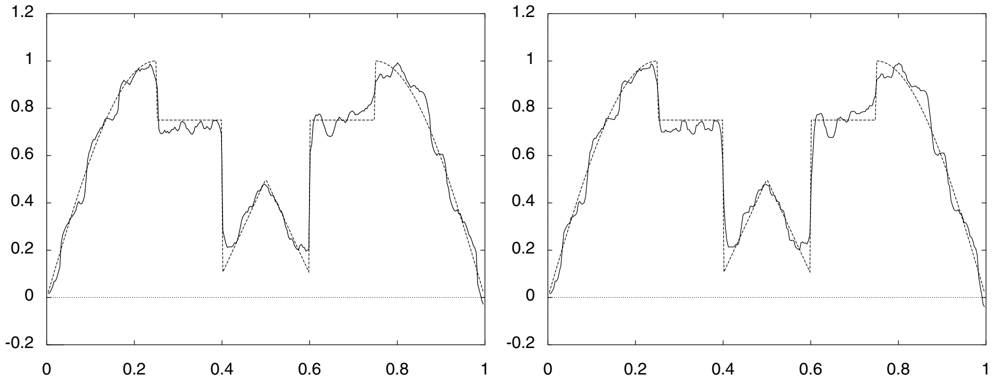

Figure 9. The results are presented in example 6.1 for $n = 300$. On the left, the δ-formulation (10) for $g = 0.001$ and on the right, the s-formulation (7) for the best value of g w.r.t. $e(u^*)$. Left: $\delta = 10$, right: $s = 1.5$.

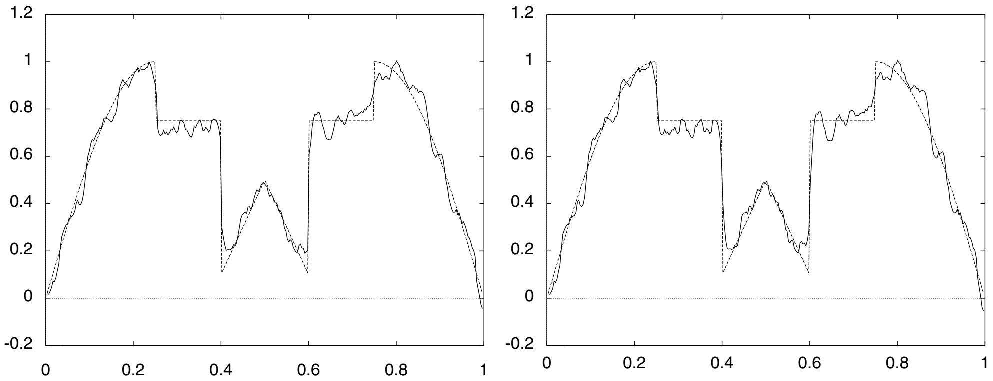

Figure 10. The results are presented in example 6.1 for $n = 300$. On the left, the δ-formulation (10) for $g = 0.001$ and on the right, the s-formulation (7) for the best value of g w.r.t. $e(u^*)$. Left: $\delta = 15$, right: $s = 1.7$.

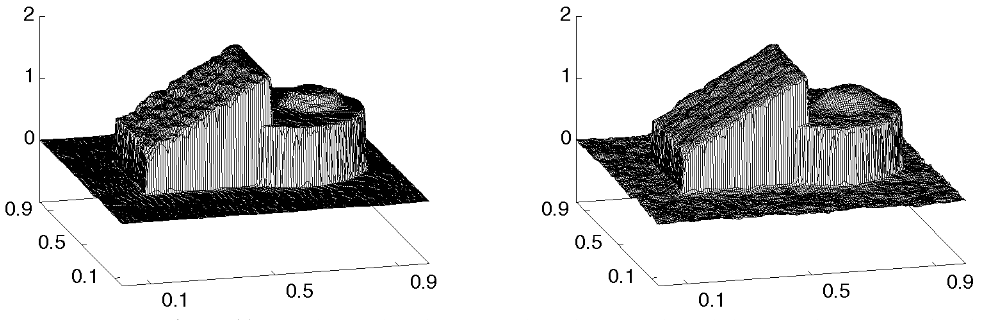

Figure 11. The results of CG are given for the δ-formulation (10) in example 6.2, for $n = 300 \times 300$, $\eta = 0.3$, and $g = 0.002$. Left: $\delta = 1$, right: $\delta = 5$.

For $s \leqslant 1.2$, AS gave a slightly larger cost function value than PB due to inexact inner iteration. However, visually and with respect to the reconstruction error $e(u^*)$, the results of AS were as good as the results of PB. AS was considerably faster than the other methods (cf tables 2 and 3). However, opposite to other methods, AS took more iterations as s grew larger. This is due to the weaker formation of the staircase-like structure for larger s (cf $|J^*|$

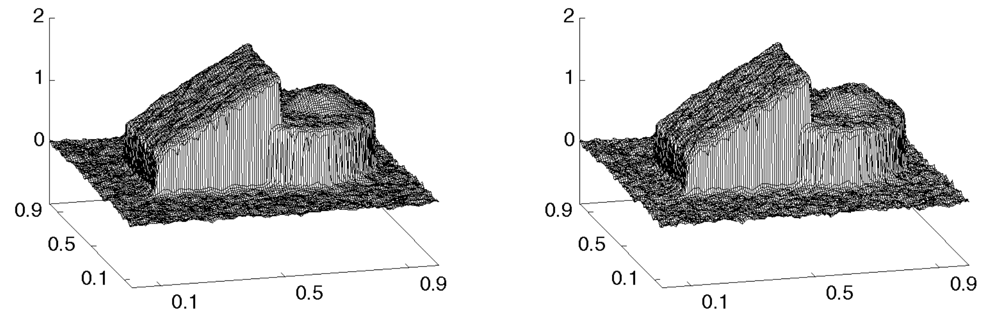

Figure 12. The results of CG are given for the δ-formulation (10) in example 6.2, for $n = 300 \times 300$, $\eta = 0.3$, and $g = 0.002$. Left: $\delta = 10$, right: $\delta = 15$.

in table 3), which readily decreases the significance of active/inactive partitioning in step 2 of algorithm 4.2. This also reflects the initial purpose in [6–8] to develop an efficient method especially for nonsmooth problems.

We also tested the monotone active-set method of [7, 8] in example 6.2 for $s = 1$. The CPU-times for three different problem sizes in table 5 were 0.86, 4.00, and 13.86 with errors of order 1×10^{-2} between the solution and the corresponding solution of CG for $\delta = 1$. Moreover, it is also known from [8] that the computational complexity of the monotone active-set method has roughly a linear growth w.r.t. the size of a problem. Altogether, this and the other tests show that general methods of smooth or nonsmooth optimization compared to special methods do not yield reasonable efficiency in image restoration.

An important question concerning the solution methods is when to stop the optimization procedure. In tables 2 and 3, the method with a larger cost function value has always a smaller reconstruction error $e(u^*)$. Hence, according to the error measure $e(u^*)$, the optimization problems should not be solved exactly, but a difficulty to quantify an inexact stopping criterion remains.

6.2. Comparison of formulations

Let us first briefly comment the s-formulation (7) in the one-dimensional case. The smoothing effect of jumps is dominating the reconstruction error $e(u^*)$ for $s > 1$. For obtaining the smallest reconstruction error, we had to decrease g when s was increased, which caused the best results to contain visually too much variation (cf table 6 and figures 7–10 (right)). This, however, conflicts the fact that when the noise statistics is fixed and known *a priori*, g is nothing more than the inverse of the Lagrange multiplier for the corresponding noise constraint [4], independently of the form of regularization.

The δ-formulation (10) and the s-formulation (7) give practically the same reconstructions when s, δ and g are chosen appropriately (cf figures 7–10). The δ-formulation, however, recovers the large jumps slightly better. Naturally, this is due to the different behaviour of the 'tales' in figure 1. Finally, tables 4 and 5 show that by suitable choices of δ and ε, also the δ-formulation (10) and the ε-formulation (11) yield practically the same reconstructions: errors between the results u_δ^* and u_ε^* are at least one order of magnitude smaller than the reconstruction errors $e(u^*)$. Moreover, the ε-formulation (11) is more efficient by means of the number of iterations and the elapsed CPU-time for small values of ε, whereas the δ-formulation seems to be slightly more efficient in smoother cases.

To conclude, practically all regularization functions considered can realize a smooth transition between the nonsmooth BV-case and the smooth H_0^1-case.

Table 7. Error $e(u^*)$ for different values of g, η. Example 6.2 for $n = 200 \times 200$.

	$\delta = 5$					$\delta = 10$				
	η					η				
g	0.05	0.1	0.3	0.5	1.0	0.05	0.1	0.3	0.5	1.0
0.0001	0.0236	0.0692	0.2685	0.4695	0.9728	0.0262	0.0696	0.2685	0.4695	0.9728
0.0005	$\underline{0.0126}$	$\underline{0.0202}$	0.1538	0.3440	0.8398	$\underline{0.0153}$	0.0250	0.1542	0.3441	0.8398
0.001	0.0178	0.0205	0.0673	0.2167	0.6866	0.0199	$\underline{0.0240}$	0.0716	0.2172	0.6866
0.003	0.0375	0.0379	$\underline{0.0440}$	$\underline{0.0580}$	0.2457	0.0416	0.0421	$\underline{0.0495}$	$\underline{0.0653}$	0.2474
0.005	0.0534	0.0535	0.0559	0.0620	0.1008	0.0598	0.0599	0.0626	0.0693	0.1095
0.01	0.0870	0.0870	0.0875	0.0890	$\underline{0.0980}$	0.0982	0.0982	0.0986	0.1001	$\underline{0.1087}$

Table 8. Error $e(u^*)$ for different values of g, η. Example 6.2 for $n = 300 \times 300$, $\delta = 5$.

	η				
g	0.05	0.1	0.3	0.5	1.0
0.0001	0.0153	0.0550	0.2503	0.4495	0.9492
0.0005	$\underline{0.0117}$	$\underline{0.0156}$	0.0989	0.2725	0.7550
0.001	0.0179	0.0192	$\underline{0.0370}$	0.1228	0.5432
0.003	0.0374	0.0376	0.0407	$\underline{0.0479}$	0.1036
0.005	0.0531	0.0531	0.0543	0.0575	$\underline{0.0727}$
0.01	0.0864	0.0864	0.0866	0.0874	0.0916
g^*_{BV}	0.0002	0.0005	0.002	0.003	0.008

6.3. Choice of g and convergence w.r.t. mesh size and noise

Next, we briefly study the convergence of solutions as the mesh size h and/or the noise level η tend to zero, and the determination of regularization parameter g as a function of η [30, 31].

In tables 7 and 8, obtained errors $e(u^*)$ for different values of the regularization parameter g and the noise level η are given for the δ-formulation (10) in example 6.2. The smallest value of $e(u^*)$ is underlined for each η. The tables indicate that best g w.r.t. η is independent of δ or the size of the problem. Moreover, it can be seen that the results become more accurate as η decreases or the grid gets finer.

We also tested the heuristic determination of g, as proposed in [32] for the BV-regularized problem. The obtained parameters, well in agreement with the best ones, are also given in table 8 in the final row g^*_{BV}. Hence, the approach in [32] also yields an efficient choice of g for the δ-formulation.

6.4. Towards a better formulation

The reconstructions obtained using larger values of s and δ in formulations (10) and (7) are no better than the results obtained for $s = 1$ or $\delta = 1$ (which coincides with $s = 1$). Hence, it is obvious that something more must be done for recovering smooth (sub)surfaces better, that is, some form of adaptivity is needed. Attempts to this direction have been made, for example, [4, 11–13]. In [11], the idea was to use a BV-like regularization ψ_{BV} near edges, smooth regularization ψ_{smooth} in flat regions, and regularization ψ_s, $1 < s < 2$, in between. The exponent s was chosen to be gradient driven; $s = s(|\nabla u|)$. In [12], the authors proposed adding a nonlinear fourth-order diffusive term to the Euler–Lagrange equations of the BV model that substantially reduced the staircase effect while preserving sharp discontinuities. In [13], the proposed regularization was of the form $\psi\left(|\nabla u|^2\right)$, where the function ψ was chosen such that a BV-like regularization is used near edges as well as in flat regions and smooth regularization between. Finally, in [4], a method based on inf-convolution decomposed the unknown function u into a sum of a smooth function and a function containing the jumps which were then used as unknowns in the enlarged optimization problem. As was already pointed out, such adaptive formulations often lack convexity and smoothness, and the number of unknowns and free parameters is usually increased. In [33], we proposed a semi-adaptive, strictly convex formulation lying between no adaptivity and full adaptivity, which gave promising results in the preliminary tests.

6.5. How to compare different reconstructions

Usually, the reconstruction giving smallest $e(u^*)$ is not the best one visually. Hence, the squared error measure $e(u^*)$ is perhaps not an optimal one. To conclude, $e(u^*)$ contains quantitative information, but how a result actually looks like is also a qualitative matter, and we should be able to combine these two characteristics to obtain a robust error measure for comparing different reconstructions.

Acknowledgments

We would like to thank the anonymous referees for their valuable comments, which improved the contents of the paper significantly. We are also deeply indebted to Professor Roland Glowinski for pointing out the ε-formulation. This work was supported by the Academy of Finland under projects 8583 and 50138, and by The Finnish Cultural Foundation.

References

[1] Chen Z and Zhou J 1999 An augmented Lagrangian method for identifying discontinuous parameters in elliptic systems *SIAM J. Control Optim.* **37** 892–910

[2] Hermann U and Noll D 2000 Adaptive image reconstruction using information measures *SIAM J. Control Optim.* **38** 1223–40

[3] Dobson D and Santosa F 1994 An image enhancement technique for electrical impedance tomography *Inverse Problems* **10** 317–34

[4] Chambolle A and Lions P L 1997 Image recovery via total variation minimization and related problems *Numer. Math.* **76** 167–88

[5] Rudin L I, Osher S and Fatemi E 1992 Nonlinear total variation based noise removal algorithms *Physica* D **60** 259–68

[6] Ito K and Kunisch K 1999 An active set strategy based on the augmented Lagrangian formulation for image restoration *Math. Mod. Numer. Anal.* **33** 1–21

[7] Kärkkäinen T and Majava K 2000 Nonmonotone and monotone active-set methods for image restoration: part 1. Convergence analysis *J. Optim. Theory Appl.* **106** 61–80

[8] Kärkkäinen T and Majava K 2000 Nonmonotone and monotone active-set methods for image restoration: part 2. Numerical results *J. Optim. Theory Appl.* **106** 81–105

[9] Mäkelä M M and Neittaanmäki P 1992 *Nonsmooth Optimization. Analysis and Algorithms with Applications to Optimal Control* (Singapore: World Scientific)

[10] Lukšan L and Vlček J 1998 A bundle-Newton method for nonsmooth unconstrained minimization *Math. Program.* **83** 373–91

[11] Blomgren P, Chan T and Mulet P 1997 Extensions to total variation denoising *Advanced Signal Processing Algorithms, Architectures and Implementations (Proc. SPIE vol 3162)* vol 7, ed F Luk (Bellingham, WA: SPIE Optical Engineering Press)

[12] Chan T, Marquina A and Mulet P 2000 High-order total variation-based image restoration *SIAM J. Sci. Comput.* **22** 503–16

[13] Ito K and Kunisch K 2000 BV-type regularization methods for convoluted objects with edge, flat and grey scales *Inverse Problems* **16** 909–28

[14] Giusti E 1984 Minimal surfaces and functions of bounded variation *Monographs in Mathematics* vol 80 (Boston, MA: Birkhäuser)

[15] Dobson D C and Santosa F 1996 Recovery of blocky images from noisy and blurred data *SIAM J. Appl. Math.* **56** 1181–98

[16] Ring W 2000 Structural properties of solutions to total variation regularization problems *Math. Mod. Numer. Anal.* **34** 799–810

[17] Miettinen K 1999 Nonlinear multiobjective optimization *Kluwer's Int. Series in Operations Research and Management Science* vol 12 (Boston: Kluwer)

[18] Vogel C R and Oman M E 1996 Iterative methods for total variation denoising *SIAM J. Sci. Comput.* **17** 227–38

[19] Glowinski R 2000 Private discussion

[20] Aubert G and Vese L 1997 A variational method in image recovery *SIAM J. Numer. Anal.* **34** 1948–79

[21] Kärkkäinen T, Majava K and Mäkelä M M 2000 Comparison of formulations and solution methods for image restoration problems *Technical Report* B 14/2000 Department of Mathematical Information Technology, University of Jyväskylä

[22] Bertsekas D P 1982 *Constrained Optimization and Lagrange Multiplier Methods* (New York: Academic)

[23] Ito K and Kunisch K 2000 Augmented Lagrangian methods for nonsmooth, convex optimization in Hilbert spaces *Nonlinear Anal.* **41** 591–616

[24] Kelley J E 1960 The cutting plane method for solving convex programs *J. SIAM* **8** 703–12

[25] Kiwiel K C 1990 Proximity control in bundle methods for convex nondifferentiable optimization *Math. Program.* **46** 105–22

[26] Kiwiel K C 1985 *Methods of Descent for Nondifferentiable Optimization (Lecture Notes in Mathematics vol 1133)* (Berlin: Springer)

[27] Kiwiel K C 1986 A method for solving certain quadratic programming problems arising in nonsmooth optimization *IMA J. Numer. Anal.* **6** 137–52

[28] Lukšan L 1984 Dual method for solving a special problem of quadratic programming as a subproblem at linearly constrained nonlinear minmax approximation *Kybernetika* **20** 445–57

[29] Nocedal J and Wright S 1999 *Numerical Optimization (Springer Series in Operations Research)* (New York: Springer)

[30] Leonov A S 1999 Numerical piecewise-uniform regularization for two-dimensional ill-posed problems *Inverse Problems* **15** 1165–76

[31] Scherzer O, Engl H W and Kunisch K 1993 Optimal *a posteriori* parameter choice for Tikhonov regularization for solving nonlinear ill-posed problems *SIAM J. Numer. Anal.* **30** 1796–838

[32] Kärkkäinen T and Majava K 2000 Determination of regularization parameter in monotone active set method for image restoration *Proc. 3rd European Conf. on Numerical Mathematics and Advanced Applications* ed T Tiihonen, P Neittaanmäki and P Tarvainen (Singapore: World Scientific)

[33] Kärkkäinen T and Majava K 2000 SAC-methods for image restoration *Recent Advances in Applied and Theoretical Mathematics* ed N Mastorakis (Greece: World Scientific and Engineering Society)