Proximal Algorithms in Inverse Imaging Problems: First and Second-Order Splitting

Simão Eduardo, e-mail: simao_eduardo@outlook.com

Abstract—**Large-scale convex optimization problems with nonsmooth functionals are found in many areas. Several convex splitting strategies using (Moreau) Proximity operators have been developed. This report compares competitive First-Order strategies (e.g. Forward-Backward, Chambolle-Pock) to a relevant Second-Order one: the Alternating Direction Method of Multipliers. We analyze convergence rates for these through empiric results, recurring to two pertinent examples in (regularized) Inverse Problems in Imaging: "Super-Resolution Total Variation Deblurring" and "Hyperspectral Image Segmentation". Indeed, the Second-Order approach reveals a marked empirical advantage, where the problem structure allows for efficient implementation.**

I. Introduction

PROBLEMS in convex optimization containing nonsmooth functionals in their overall objective function are common in many areas. Examples range from ill-posed inverse problems in imaging to machine learning, or even other areas of signal processing [1], [2], [3], [4], [5], [6], [7]. The so-called Proximal algorithms can tackle this type of problems.

All Proximal algorithms make use of a base operation, the (Moreau) Proximity operator of a function [1], which in turn involves solving several smaller convex optimization problems. Furthermore, these can be solved with standard methods, but usually offer closed-form solutions or make use of straightforward specialized methods. In fact, frequently the overall objective function can be decomposed into a sum of several lower-dimension functionals (i.e. Splitting property), such that the former properties can be used to promotes parallelism. An algorithm which aims at minimizing a sum of functionals by successive evaluations of their gradients or Proximity operators is a Proximal algorithm.

Indeed, the Proximity operator is the resolvent of the inclusion involving the subdifferential operator, which arises from the nonsmoothness in the objective function [1], [8], [5], [9]. A clear formulation will be given.

Firstly, we present some definitions: $\mathcal{X}$ is some Hilbert space (Euclidean space $\mathbb{R}^n$ is used throughout); $\|.\|$ is the Euclidean norm; Γ_0 is the set of all functions $f : \mathcal{X} \to \mathbb{R} \cup \{\infty\}$ that are closed, proper (i.e., nonempty effective domain) and convex; $\boldsymbol{l}$ is defined as a vector, and $\boldsymbol{L}$ as a matrix; $\boldsymbol{I}_d$ is the identity matrix; $\boldsymbol{L}^*$ is the conjugate transpose of matrix $\boldsymbol{L}$; $f^*(.)$ is the (Fenchel) conjugate of function $f(.)$, in [9]; and function $f(.)$ is β-Lipschitz continuous if $\|f(\boldsymbol{x}) - f(\boldsymbol{x}')\| \leq \beta\|\boldsymbol{x} - \boldsymbol{x}'\|$, such that $\forall \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$, $\beta > 0$.

Prof. José Bioucas-Dias and Prof. Mário Figueiredo (Instituto de Telecomunicações) were integral parts of this project. The author would like to thank them for the opportunity of working together. They are with Department of Electrical and Computer Engineering, Instituto Superior Técnico, Lisbon. Simão Eduardo was a Research Assistant Fellow at the same institution.

Now, we are ready to present the inclusion associated with the Proximity operator, which is normally defined for every $(\boldsymbol{p}, \boldsymbol{x}) \in \mathcal{X} \times \mathcal{X}$ and $\eta > 0$ as

$$\boldsymbol{p} \in \boldsymbol{x} + \eta\partial f(\boldsymbol{x}) \Leftrightarrow \boldsymbol{x} \in (\boldsymbol{I}_d + \eta\partial f)^{-1}(\boldsymbol{p}), \tag{1}$$

where $\boldsymbol{p}$ is to be mapped into $\boldsymbol{x}$, and $prox_{\eta f} = (\boldsymbol{I}_d + \eta\partial f)^{-1}$ is the Proximity operator (i.e. the resolvent of subgradient ∂f). Furthermore, the inclusion above is equivalent to the unique solution of the following optimization problem

$$prox_{\eta f} : \boldsymbol{p} \to \underset{\boldsymbol{x}\in\mathcal{X}}{\operatorname{argmin}} \frac{1}{2\eta}\|\boldsymbol{p} - \boldsymbol{x}\|^2 + f(\boldsymbol{x}), \tag{2}$$

another definition of the Proximity operator mapping. In [7] information on *Proximal* calculus, and particular closed-form formulations for typical *Proximity* operators are given.

Another important aspect is how to define First and Second-Order Splitting methods, see [5], [10], [11] and [8]. The difference relies on the absence or use of second order information. A typical Second-Order method is the Newton method, which uses the inverse of the Hessian, requiring the objective function to be twice differentiable. However, second order information can be defined more broadly, as the inverse of some linear operator $\boldsymbol{L}$ or $\boldsymbol{L}^*\boldsymbol{L}$ that appears in the objective function. This is akin to solving a linear equation, which arises in the case that all terms in the objective function are either linear or quadratic ([5], part 4). Further, it often contains the function's curvature information, permitting a more direct route to the minimum/ maximum. Moreover, Second-Order methods require this operator inversion to have relatively efficient implementation. On the other hand, First-Order methods only rely on linear operators, gradients and Proximity operators of functions ($\boldsymbol{L}$, $\boldsymbol{L}^*$, ∇f, $prox_{\eta f}$, respectively), without operator inversion.

Instinctively, it is expected that Second-Order methods will offer faster convergence rates than First-Order ones, presuming efficient exploitation of the problem's structure (e.g. matrix factorization, splitting of the objective function, closed-form formulations). Additionally, our work focus on problems with nonsmooth functionals, so iterative approaches like Proximal algorithms are necessary.

The remainder of this report is organized as follows. Section II introduces relevant *Second-Order* and *First-Order Proximal Algorithms*: *Alternating Direction Method of Multipliers* (ADMM), and *Generic Proximal Algorithm* (by Laurent Condat). Optimization problem formulation, convergence conditions, and algorithm specification is given. Afterwards, we use two applications from the inverse problems in imaging domain to exemplify convergence rate performance, in an experimental fashion; whilst at the same time providing algorithm implementation, convergence and computational analysis. We start with super-resolution image deblurring, in section III, and then finish with hyperspectral image segmentation, in section IV. Lastly, in section V, we present some remarks about the algorithms, and their associated convergence results.

II. Algorithm Definition and Specification

Now we introduce the framework for the *Proximal Algorithms* that will be used throughout the document.

A. Alternating Direction Method of Multipliers - ADMM

First off, we introduce a Second-Order Proximal algorithm, a version of the *Alternating Direction Method of Multipliers* (ADMM) [2] [3] [5] [12] [4] [13], also known as *Split Bregman*, or *Douglas-Rachford Splitting* (on the Dual problem). This is a popular Primal-Dual Splitting strategy for large-scale convex optimization, with a wide array of applications [2].

We will adopt an ADMM approach akin to [14] [3] [15], leading to Algorithm 1. In order to understand how a generalized (convex) optimization problem is mapped into Algorithm 1, we start with the subsequent unconstrained minimization problem:

$$\min_{\boldsymbol{z}\in\mathbb{R}^d} \Phi(\boldsymbol{z}) = \min_{\boldsymbol{z}\in\mathbb{R}^d} f_1(\boldsymbol{z}) + f_2(\boldsymbol{G}\boldsymbol{z}), \tag{3}$$

where $f_1 : \mathbb{R}^d \to \mathbb{R}$ such that $f_1 \in \Gamma_0(\mathcal{X})$, $f_2 : \mathbb{R}^p \to \mathbb{R}$ such that $f_2 \in \Gamma_0(\mathcal{X})$, while $\boldsymbol{G} \in \mathbb{R}^{p\times d}$ has full column rank and is given by

$$\boldsymbol{G} = [\boldsymbol{H}^{*(1)} \cdots \boldsymbol{H}^{*(j)} \cdots \boldsymbol{H}^{*(J)}]^*, \tag{4}$$

where linear operators $\boldsymbol{H}^{(j)} \in \mathbb{R}^{p_j\times d}$ are arbitrary matrices such that $p = p_1 + \cdots + p_J$. Further, f_2 can also be given by

$$f_2(\boldsymbol{v}) = \sum_{j=1}^{J} g_j(\boldsymbol{v}^{(j)}), \tag{5}$$

where $g_j : \mathbb{R}^{p_j} \to \mathbb{R}$ such that $g_j \in \Gamma_0(\mathcal{X})$, and $\boldsymbol{v}^{(j)} \in \mathbb{R}^{p_j}$ such that $\boldsymbol{v} = [\boldsymbol{v}^{*(1)} \cdots \boldsymbol{v}^{*(J)}]^*$, with accordance to $\boldsymbol{v} \in \mathbb{R}^p$; and therefore we can have $\boldsymbol{v} = \boldsymbol{G}\boldsymbol{z}$.

Accordingly, problem (3) can take the following alternative form

$$\min_{\boldsymbol{z}\in\mathbb{R}^d} \Phi(\boldsymbol{z}) = \min_{\boldsymbol{z}\in\mathbb{R}^d} f_1(\boldsymbol{z}) + \sum_{j=1}^{J} g_j(\boldsymbol{H}^{(j)}\boldsymbol{z}), \tag{6}$$

As a result, it reveals both function and variable Splitting in ADMM, by use of g_j and $\boldsymbol{v}^{(j)}$ respectively.

Now, considering the problem defined in (6) and choosing $f_1(.) = 0$, ADMM algorithm structure is fully defined in Algorithm 1, which includes variation of μ, algorithm step-size parameter, and α, relaxation step, as seen in [2]. Jointly, μ, also known as augmented Lagrangian penalty parameter, and vector $\boldsymbol{d}^{(j)} \in \mathbb{R}^{p_j}$ such that $\boldsymbol{d} = [\boldsymbol{d}^{*(1)} \cdots \boldsymbol{d}^{*(J)}]^* \in \mathbb{R}^p$, define the algorithm's Lagrange multipliers. In fact, the values of vector $\boldsymbol{d}$ and μ are the variables and the step of the Dual problem, respectively; while $\boldsymbol{z}$ and auxiliary vector $\boldsymbol{v}$ represent the variables of the Primal problem [2].

Step 6 in Algorithm 1 is the relaxation process (over-relaxation if $\alpha > 1$), which has been found to increase convergence if $\alpha \in [1.5, 1.8]$, see [2].

From step 10 to 25, every n iterations (to lower computational overhead), ADMM's residues r^p and r^d are computed. These are then used for μ value variation decision, multiplying or diving by ϵ, if they differ in a factor of ζ or more, We use the values recommended in [2], $\epsilon = 2$ and $\zeta = 10$. Moreover, if indeed μ changes, then operators like $prox_{g_j/\mu}(.)$, or any pre-calculated structures dependent on μ, must be updated before the next iteration.

Algorithm 1 ADMM Structure

```
1:  init set $k = 0$, $\boldsymbol{\vartheta} = \boldsymbol{0}$, $r^p \in \mathbb{R}$, $r^d \in \mathbb{R}$, choose $\mu > 0$,
    $n \in \mathbb{N}$, $\alpha \in (0, 2)$, $\boldsymbol{z}_0$, $\boldsymbol{v}_0$, $\boldsymbol{d}_0$.
2:  repeat
3:      $\boldsymbol{z}_{k+1} \leftarrow \big(\sum_{j=1}^J \boldsymbol{H}^{*(j)}\boldsymbol{H}^{(j)}\big)^{-1} \sum_{j=1}^J \boldsymbol{H}^{*(j)}\big(\boldsymbol{v}_k^{(j)} +$
4:          $+\boldsymbol{d}_k^{(j)}\big),$
5:      for $j = 1$ to $J$ do
6:          $\boldsymbol{\vartheta}^{(j)} = \alpha\boldsymbol{H}^{(j)}\boldsymbol{z}_{k+1} + (1-\alpha)\boldsymbol{v}_k^{(j)},$
7:          $\boldsymbol{v}_{k+1}^{(j)} \leftarrow prox_{g_j/\mu}\big(\boldsymbol{\vartheta}^{(j)} - \boldsymbol{d}_k^{(j)}\big),$
8:          $\boldsymbol{d}_{k+1}^{(j)} \leftarrow \boldsymbol{d}_k^{(j)} - \big(\boldsymbol{\vartheta}^{(j)} - \boldsymbol{v}_{k+1}^{(j)}\big),$
9:      end for
10:     if $n$ iterations passed then
11:         $r^p = \big(\sum_{j=1}^J \|\boldsymbol{H}^{(j)}\boldsymbol{z}_{k+1} - \boldsymbol{v}_{k+1}^{(j)}\|^2\big)^{1/2},$
12:         $r^d = \mu\big(\sum_{j=1}^J \|\boldsymbol{H}^{*(j)}(\boldsymbol{v}_{k+1}^{(j)} - \boldsymbol{v}_k^{(j)})\|^2\big)^{1/2},$
13:         if $r^p > \zeta r^d$ then
14:             $\mu \leftarrow \epsilon\mu,$
15:             for $j = 1$ to $J$ do
16:                 $\boldsymbol{d}_{k+1}^{(j)} \leftarrow \frac{\boldsymbol{d}_{k+1}^{(j)}}{\epsilon},$
17:             end for
18:         end if
19:         if $r^d > \zeta r^p$ then
20:             $\mu \leftarrow \frac{\mu}{\epsilon},$
21:             for $j = 1$ to $J$ do
22:                 $\boldsymbol{d}_{k+1}^{(j)} \leftarrow \epsilon\boldsymbol{d}_{k+1}^{(j)},$
23:             end for
24:         end if
25:     end if
26:     $k \leftarrow k + 1,$
27: until stopping criterion is satisfied
```

The stopping criteria can be given by some relative error on the optimal (minimum) solution $\boldsymbol{z}$, or on the objective function (6). However, other criteria like a tolerance bound on ADMM's residues r^p and r^d as in [2], or a maximum number of iterations for execution is also used.

Further, ADMM convergence is guaranteed by *Eckstein-Bertsekas*: considerations about the generalized version of ADMM are found in [16]; however, for the ADMM version found in this section, see [3]. Hence, for this version of ADMM to converge we define the following theorem:

Theorem 1 (modified Eckstein-Bertsekas):

- $\boldsymbol{G} \in \mathbb{R}^{p\times d}$ has full column rank,
- f_1, $\forall_j\ g_j \in \Gamma_0(\mathcal{X})$,

- $\mu > 0$,
- $\boldsymbol{d} \in \mathbb{R}^p$,
- Set of minimizers is nonempty.

Then, if the set of minimizers is nonempty, sequence $(\boldsymbol{z}_k)_{k\in\mathbb{N}}$ generated in Algorithm 1 converges to a set of minimizers $\hat{\boldsymbol{z}}$ of (3).

Furthermore, linear convergence, at this point, is not possible to guarantee since most proofs rely on either strict (or even strong) convexity of the function terms present in objective (6). Still, recent work on the overall convergence of the ADMM, particularly on linear convergence, can be found in [17], [18], [19].

Indeed, given the applications that will be showcased, none has an objective function where its terms are strictly convex. Nevertheless, the overall objective is expected to be strictly convex, because we will be dealing with regularization applications, even though sometimes this turns out not to be enough. So, as stated before, we will be dealing with ill-posed problems that have this characteristic. Seeing that we just seek to prove, experimentally, that in certain applications a specialized ADMM is more advantageous than other recent or relevant *Proximal* algorithms, convergence guarantee *Theorem 1* is enough.

Moreover, as discussed in [2], μ variation during runtime (step 10 to 25) eventually leads to μ stabilization, and eventual faster ADMM convergence. Hence, this being assumed, all arguments made for static-μ ADMM about convergence, also hold here. In fact, now initial μ value is somewhat less relevant for convergence, still, a tailored value may shorten μ stabilization (i.e. an initial optimal value, since update may take some iterations, as defined by the user).

Lastly, for this version of ADMM, as specified in Algorithm 1, we can easily see that in step 4, the second order information is defined by $\left(\sum_{j=1}^{J} \boldsymbol{H}^{*(j)}\boldsymbol{H}^{(j)}\right)^{-1}$, as described in the Introduction.

B. Generic Proximal Algorithm

Now we finally introduce a First-Order strategy, the *Generic Proximal Algorithm*. This Primal-Dual convex formulation encompasses a family of First-Order algorithms, and it has been devised by L. Condat in [8], [11]. It provides a solution to the subsequent generalized unconstrained (convex) optimization problem:

$$\min_{\boldsymbol{z}\in\mathbb{R}^d} \Phi(\boldsymbol{z}) = \min_{\boldsymbol{z}\in\mathbb{R}^d} f_1(\boldsymbol{z}) + f_2(\boldsymbol{z}) + \sum_{j=1}^{J} g_j(\boldsymbol{H}^{(j)}\boldsymbol{z}), \qquad (7)$$

where $f_1,\ f_2 : \mathbb{R}^d \to \mathbb{R}$ and $g_j : \mathbb{R}^{p_j} \to \mathbb{R}$, such that f_1, f_2, $g_j \in \Gamma_0(\mathcal{X})$; $\boldsymbol{H}^{(j)} \in \mathbb{R}^{d\times p_j}$ operators are linear and bounded; f_1 is differentiable on $\mathbb{R}^d$ and its gradient ∇f_1 is β-Lipschitz continuous, for some real constant $\beta > 0$; the set of minimizers is nonempty.

In [11] (the *Proposed Algorithm 1*), L. Condat presents a generic algorithm for solving problem (7). Algorithm 2 displays Condat's proposal, where $\boldsymbol{u}^{(j)} \in \mathbb{R}^{p_j}$ such that $p = p_1 + \cdots + p_J$ is comprised of the variables of the Dual problem. For clarity's sake, one defines $\tilde{\boldsymbol{z}}_{k+1}$ just as an auxiliary variable.

Algorithm 2 Generic Proximal Algorithm - L. Condat

```
1:  init choose $\tau > 0$, $\sigma > 0$, $\rho > 0$, $\boldsymbol{z}_0$, $\forall_j\ \boldsymbol{u}_0^{(j)}$.
2:  repeat
3:      $\tilde{\boldsymbol{z}}_{k+1} \leftarrow prox_{\tau f_2}\Big(\boldsymbol{z}_k - \tau\big(\nabla f_1(\boldsymbol{z}_k) +$
4:                  $+ \sum_{j=1}^{J} \boldsymbol{H}^{*(j)}\boldsymbol{u}_k^{(j)}\big)\Big)$,
5:      $\boldsymbol{z}_{k+1} \leftarrow \rho\tilde{\boldsymbol{z}}_{k+1} + (1-\rho)\boldsymbol{z}_k$,
6:      for $j = 1$ to $J$ do
7:          $\tilde{\boldsymbol{u}}_{k+1}^{(j)} \leftarrow prox_{\sigma g_j^*}\Big(\boldsymbol{u}_k^{(j)} + \sigma\boldsymbol{H}^{(j)}(2\tilde{\boldsymbol{z}}_{k+1} - \boldsymbol{z}_k)\Big)$,
8:          $\boldsymbol{u}_{k+1}^{(j)} \leftarrow \rho\tilde{\boldsymbol{u}}_{k+1}^{(j)} + (1-\rho)\boldsymbol{u}_k^{(j)}$,
9:      end for
10:     $k \leftarrow k+1$,
11: until stopping criterion is satisfied
```

In relation to convergence guarantees, whose proofs can be found in [8], though also enunciated in [11], two theorems are defined:

Theorem 2 (Full Generic Proximal Algorithm): Suppose that the parameters in Algorithm 2 satisfy the following

- $\tau\left(\frac{\beta}{2} + \sigma\left\|\sum_{j=1}^{J} \boldsymbol{H}^{*(j)}\boldsymbol{H}^{(j)}\right\|\right) < 1$, where β is the already defined Lipschitz constant.
- $\rho \in]0,1]$, where ρ is the relaxation constant.

Then both sequences $(\tilde{\boldsymbol{z}}_k)_{k\in\mathbb{N}}$ and $(\boldsymbol{z}_k)_{k\in\mathbb{N}}$ generated in Algorithm 2 converge to a set of minimizers $\hat{\boldsymbol{z}}$ of (7).

Theorem 3 (Chambolle-Pock Algorithm): Suppose that $f_1(.) = 0$, and that the parameters in Algorithm 2 satisfy the following

- $\tau\sigma\left\|\sum_{j=1}^{J} \boldsymbol{H}^{*(j)}\boldsymbol{H}^{(j)}\right\| \leq 1$,
- $\rho \in]0,2[$, where ρ is the relaxation constant.

Then both sequences $(\tilde{\boldsymbol{z}}_k)_{k\in\mathbb{N}}$ and $(\boldsymbol{z}_k)_{k\in\mathbb{N}}$ generated in Algorithm 2 converge to a set of minimizers $\hat{\boldsymbol{z}}$ of (7).

Moreover, from the generic approach in problem (7), we can further construe other relevant *Proximal* algorithm versions, with a relaxation option.

In fact, in the case of $f_1(.) = 0$, the proposed algorithm reverts to the (Primal-Dual) *Chambolle-Pock (CP)* method [6], with additional relaxation. In this case, according to Theorem 3, we should allow for a value ρ close to 2 [11], instead of $\rho = 1$ [6], which can significantly accelerate convergence. Furthermore, convergence is guaranteed with the choice $\tau\sigma\left\|\sum_{j=1}^{J} \boldsymbol{H}^{*(j)}\boldsymbol{H}^{(j)}\right\| = 1$, which is recommended by [11] in practice, whereas $\tau\sigma\left\|\sum_{j=1}^{J} \boldsymbol{H}^{*(j)}\boldsymbol{H}^{(j)}\right\| \leq 1$ was given by [6].

However, for $J = 0$ one simply minimizes $f_1(\boldsymbol{z}) + f_2(\boldsymbol{z})$, and the proposed algorithm reverts to the (Primal) *Forward-Backward (FB)* method [7], with additional relaxation.

Yet, other ways to assign the functions in (7) exist, and we will explore that fact in our applications.

Generally, using function f_1 whenever possible is probably better for convergence speed.

Firstly, because of the serial way variables are updated: the step of the gradient descent with respect to f_1 updates and enhances $\boldsymbol{z}$, and then this new value is used to update the several $\boldsymbol{u}^{(j)}$ (i.e. dual variables). In contrast, variables $\boldsymbol{u}^{(j)}$ are updated independently, with respect to g_j functions, before being basically averaged to form the new estimate of $\boldsymbol{z}$.

Secondly, *subgradient* methods, such as *Proximal algorithms*, are at best as fast as *direct gradient* methods when the function at play is differentiable (in this case f_1), though, usually they are slower. So, once more, if one of the function terms in the objective is differentiable, it should be assigned to f_1, as there will be the least cost to convergence. Hence, all things being equal, one could roughly say *Forward-Backward* algorithm, if applicable, is preferred to *Chambolle-Pock*.

III. Applications in Inverse Problems in Imaging: Super-Resolution Total Variation Deblurring (SR-TV)

A. Super-Resolution Total Variation (SR-TV) Deblurring: Problem Definition

First, we define the blurring process, and then make an introduction to the Total Variation (TV) deblurring formulation.

In image blurring, let $\boldsymbol{x} \in \mathbb{R}^N$ be the column-vectorized original image (each entry is a grey-scale pixel intensity), $\boldsymbol{y} \in \mathbb{R}^N$ be the vectorized blurred image, and $\boldsymbol{w} \in \mathbb{R}^N$ be a noise vector, and $\boldsymbol{T} \in \mathbb{R}^{N\times N}$ a block Toeplitz linear operator defining the (blurring) convolution applied to $\boldsymbol{x}$. N is the number of pixels of a $n \times n$ image. So, we have that

$$\boldsymbol{y} = \boldsymbol{T}\boldsymbol{x} + \boldsymbol{w} \tag{8}$$

where element w_j, from $\boldsymbol{w}$, in the case of white Gaussian noise, is such that $w_j \sim \mathcal{N}(0, \sigma^2_{\text{BSNR}})$ - BSNR is the Blurred Signal-to-Noise Ratio.

Now, since such a problem is usually ill-posed, estimation using $\hat{\boldsymbol{x}} = \boldsymbol{T}^{-1}\boldsymbol{y}$ is not usually an option; even if $\boldsymbol{T}$ was known and invertible, noise term $\boldsymbol{w}$ would still render it ill-posed. Actually, in order to properly accomplish image deblurring, one has to recur to regularization and perhaps additional constraints.

In addition, in Super-Resolution (SR) problems, operator $\boldsymbol{T}$ encapsulates further transformations. SR problems have as an objective the restoration of a high-resolution image $\boldsymbol{x} \in \mathbb{R}^N$, from a low-resolution one measured as $\boldsymbol{y}_m \in \mathbb{R}^{N/S}$. If no decent regularization is applied, then this type of problem is extremely ill-posed, since an infinity of high-resolution images $\boldsymbol{x}$ exist for a specific $\boldsymbol{y}_m$. A scale factor Ds typically defines image size down-sampling, with finite support size of $S = Ds \times Ds$ pixels.

An effective regularizer for ill-posed inverse problems in imaging is the TV semi-norm. Subsequently, the deblurring problem can be described as a convex optimization problem using the TV regularizer.

Generally, in SR problems, image estimate $\hat{\boldsymbol{x}}$ for TV-based deblurring can be defined as

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}\in\mathbb{R}^N}{\operatorname{argmin}}\, \Phi(\boldsymbol{x}) \tag{9}$$

$$= \underset{\boldsymbol{x}\in\mathbb{R}^N}{\operatorname{argmin}}\, \frac{1}{2}\|\boldsymbol{T}(\boldsymbol{x}) - \boldsymbol{y}_m\|^2 + \lambda_{TV}\|\boldsymbol{D}\boldsymbol{x}\|_{iso}, \tag{10}$$

$$\boldsymbol{T}(\ .\ ) = \boldsymbol{M}(\boldsymbol{A}\ .\ ), \tag{11}$$

where the first term of objective function (10) sees a quadratic penalty as the data-fidelity term (logarithmic penalties are also used, depending on noise type). The second term is a smoothness regularization term, where $\|\boldsymbol{D}\ .\|_{iso}$ is the two-dimensional TV isotropic semi-norm, as in [5], [11]. Further, TV isotropic semi-norm is indeed nonsmooth, which can be seen in its definition ahead.

Operator $\boldsymbol{D} \in \mathbb{R}^{2N\times N}$ is a linear convolution, containing the concatenated horizontal and vertical discrete gradient operators, with periodic conditions, given by:

$$\boldsymbol{D} = \begin{bmatrix} \boldsymbol{D}_h \\ \boldsymbol{D}_v \end{bmatrix} = \begin{bmatrix} \boldsymbol{I}_d(h) & \otimes & \boldsymbol{D}_{aux}(h) \\ \boldsymbol{D}_{aux}(v) & \otimes & \boldsymbol{I}_d(v) \end{bmatrix}, \tag{12}$$

such that, for some $b \in \mathbb{N}_+$ and $k \in \{v, h\}$, we have

$$\boldsymbol{D}_{aux}(k) = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 \\ 0 & 0 & -1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -1 & 1 \\ 1 & 0 & 0 & \cdots & 0 & -1 \end{bmatrix} \in \mathbb{R}^{b\times b},$$

where $\boldsymbol{I}_d(h) \in \mathbb{R}^{n_c\times n_c}$ and $\boldsymbol{I}_d(v) \in \mathbb{R}^{n_{cp}\times n_{cp}}$ are identity matrices; $\boldsymbol{D}_{aux}(h) \in^{n_{cp}\times n_{cp}}$ and $\boldsymbol{D}_{aux}(v) \in^{n_c\times n_c}$ are auxiliary discrete gradient matrices; with n_c as the number of columns in image $\boldsymbol{x}$ (i.e. image is size $n_c \times n_c$), while n_{cp} as the number of pixels per image column.

Then we have that for some i-th pixel $\|.\|_{iso} : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$ is defined as

$$\|(u_h, u_v)\|_{iso} = \sum_i^N \sqrt{u_{hi}^2 + u_{vi}^2}, \tag{13}$$

where $(u_h, u_v) \in \mathbb{R}^{2N}$ is the output of operator $\boldsymbol{D}$, concatenating horizontal and vertical vector results. In addition, $\lambda_{TV} > 0$ is a tunable tradeoff parameter, dependent on BSNR and linear convolution $\boldsymbol{A}$.

Moreover, factorization of $\boldsymbol{T} : \mathbb{R}^N \to \mathbb{R}^{N/S}$ in (11) allows its representation by two distinct linear operators, $\boldsymbol{M} : \mathbb{R}^N \to \mathbb{R}^{N/S}$ and $\boldsymbol{A} : \mathbb{R}^N \to \mathbb{R}^N$. The former includes both down-sampling and unknown boundary conditions (UBC) operations; whilst $\boldsymbol{A}$ is the equivalent (blurring) linear convolution operator if $\boldsymbol{T}$ was block circulant (i.e. periodic boundaries), and no down-sampling existed. This type of factorization, particularly for UBC, has already been proposed in [14]; whilst other work has assumed $\boldsymbol{T}$ as having periodic boundaries for simplicity [15] [3].

Further, $\boldsymbol{M}$ is a (linear) masking operator, filtering out the subset of pixels that either lay outside the original high-resolution boundary of $\boldsymbol{x}$, or that are not selected by the down-sampling process. Besides, under UBC, we can assume linear operators $\boldsymbol{A}$ and $\boldsymbol{D}$ to be block circulant matrices, so they can be DFT factorized (2D Discrete Fourier Transform). However, size of image N must be increased to include a sleeve of size l around image $\boldsymbol{x}$, since the blurring (kernel) support present in $\boldsymbol{A}$ has a size of $KS = (2l+1) \times (2l+1)$ pixels, so UBC can be respected [14].

B. Super-Resolution Total Variation (SR-TV) Deblurring: Applying ADMM

Now, applying the structure of ADMM, in (6), to the regularized problem in (10) and (11), the splitting process takes the form ($J = 2$):

$$\boldsymbol{z} \in \mathbb{R}^N, \qquad \boldsymbol{z} = \boldsymbol{x}, \tag{14}$$

$$\boldsymbol{v}^{(1)} \in \mathbb{R}^N, \qquad \boldsymbol{v}^{(1)} = \boldsymbol{H}^{(1)}\boldsymbol{x}, \tag{15}$$

$$\boldsymbol{v}^{(2)} \in \mathbb{R}^{2N}, \qquad \boldsymbol{v}^{(2)} = \boldsymbol{H}^{(2)}\boldsymbol{x}, \tag{16}$$

$$\boldsymbol{H}^{(1)} : \mathbb{R}^N \to \mathbb{R}^N, \qquad \boldsymbol{H}^{(1)} = \boldsymbol{A}, \tag{17}$$

$$\boldsymbol{H}^{(2)} : \mathbb{R}^N \to \mathbb{R}^{2N}, \qquad \boldsymbol{H}^{(2)} = \boldsymbol{D}, \tag{18}$$

$$f_1 : \mathbb{R}^N \to \{0\}, \qquad f_1(.) = 0, \tag{19}$$

$$g_1 : \mathbb{R}^N \to \mathbb{R}, \qquad g_1(.) = \frac{1}{2}\|\boldsymbol{M}\ . - \boldsymbol{y}_m\|^2, \tag{20}$$

$$g_2 : \mathbb{R}^{2N} \to \mathbb{R}, \qquad g_2(.) = \lambda_{TV}\|\ .\ \|_{iso}. \tag{21}$$

So, taking the above into account, proximity operators for g_1 and g_2 must be clarified. For g_1 and g_2, using the structure in (2), the proximity operators are defined as

$$\begin{aligned} prox_{g_1/\mu}(.) &= \underset{\boldsymbol{v}^{(1)}\in\mathbb{R}^N}{\operatorname{argmin}} \frac{\mu}{2}\|. - \boldsymbol{v}^{(1)}\|^2 + \frac{1}{2}\|\boldsymbol{M}\boldsymbol{v}^{(1)} - \boldsymbol{y}_m\|^2 \\ &= \big(\boldsymbol{M}^*\boldsymbol{M} + \mu\boldsymbol{I}_d\big)^{-1}\big[\boldsymbol{M}^*\boldsymbol{y}_m + \mu(.)\big], \end{aligned} \tag{22}$$

$$\begin{aligned} prox_{g_2/\mu}(.) &= \underset{\boldsymbol{v}^{(2)}\in\mathbb{R}^{2N}}{\operatorname{argmin}} \frac{\mu}{2}\|. - \boldsymbol{v}^{(2)}\|^2 + \lambda_{TV}\|\boldsymbol{v}^{(2)}\|_{iso} \\ &= \text{vect–soft}\left(\frac{\lambda_{TV}}{\mu}, \ .\ \right). \end{aligned} \tag{23}$$

Both proximity operator (22) and vector-soft operator (23) have been formulated previously, and can be found in [7], [20], [21]. Moreover, the operator in (23) can be further defined. So, for every pixel $i \in (1, \cdots, |N|)$, we define a pair-wise group $\big(u_{hi}, u_{vi}\big) \in \mathbb{R}^2$ obtained from $\boldsymbol{v}^{(2)}$, the output of operator $\boldsymbol{D}$ (16), such that we have

$$\begin{aligned} &\text{vector–soft}\left(\frac{\lambda_{TV}}{\mu}, \ .\ \right) : \\ &\big(u_{hi}, u_{vi}\big) \to \frac{\Big(\sqrt{\big|u_{hi}\big|^2 + \big|u_{vi}\big|^2} - \frac{\lambda_{TV}}{\mu}\Big)^+}{\sqrt{\big|u_{hi}\big|^2 + \big|u_{vi}\big|^2}} . \big(u_{hi}, u_{vi}\big), \end{aligned} \tag{24}$$

where $|.|$ is the modulus, and $(.)^+$ is the positive part.

Now it is important to explain that the inversion in (22) is indeed computationally efficient. Following the same approach as in [14], exploring the structure in $\boldsymbol{M}^*\boldsymbol{M}$, we see that it defines a diagonal matrix. In fact, the diagonal registers value 1 for pixels that are selected by masking matrix $\boldsymbol{M}$, and 0 for those not selected. Meanwhile, matrix summation $(\boldsymbol{M}^*\boldsymbol{M} + \mu\boldsymbol{I}_d)$ results in a matrix with diagonal values all different from 0, as long as $\mu > 0$ - guaranteed by ADMM definition, in Algorithm 1. Hence, this resulting matrix is easily invertible, in that diagonal values are inverted individually. In addition, $\boldsymbol{M}^*\boldsymbol{y}_m$ can be pre-computed in the beginning, while computation of $(\boldsymbol{M}^*\boldsymbol{M} + \mu\boldsymbol{I}_d)$ is only necessary if μ value is changed, during Algorithm 1.

On the other hand, from previous section III-A, we know that both $\boldsymbol{A}$ and $\boldsymbol{D}$ can be DFT factorized (a type of Eigendecomposition), such that some $\boldsymbol{H}^{(j)} = \boldsymbol{F}^*\boldsymbol{\Lambda}_{\boldsymbol{H}^{(j)}}\boldsymbol{F}$; where $\boldsymbol{F}$ is the unitary DFT matrix (i.e. $\boldsymbol{F}^{-1} = \boldsymbol{F}^*$), and $\boldsymbol{\Lambda}_{\boldsymbol{H}^{(j)}}$ is the diagonal matrix containing the DFT coefficients of operator $\boldsymbol{H}^{(j)}$. So, all matrix products invoking operators $\boldsymbol{A}$, $\boldsymbol{A}^*$, $\boldsymbol{D}$, $\boldsymbol{D}^*$ can be efficiently calculated via FFT algorithm with cost $O(Nlog(N))$. In fact, for some vector $\boldsymbol{a}$, $\boldsymbol{F}\boldsymbol{a}$ is the Fourier Transform of $\boldsymbol{a}$, while $\boldsymbol{F}^*\boldsymbol{a}$ is the inverse Fourier Transform.

Furthermore, computation of step 4 in Algorithm 1 becomes

$$\begin{aligned} \boldsymbol{x}_{k+1} = \big(\boldsymbol{A}^*\boldsymbol{A} + \boldsymbol{D}^*\boldsymbol{D}\big)^{-1}\big[\boldsymbol{A}^*(\boldsymbol{v}_k^{(1)} + \boldsymbol{d}_k^{(1)}) + \boldsymbol{D}^*(\boldsymbol{v}_k^{(2)} \\ + \boldsymbol{d}_k^{(2)})\big], \end{aligned} \tag{25}$$

where matrix $\big(\boldsymbol{A}^*\boldsymbol{A}+\boldsymbol{D}^*\boldsymbol{D}\big)^{-1}$, the second order information part, can be translated into a simple diagonal inversion in DFT space as in [3], [14]. So, after some manipulation, we now have

$$\begin{aligned} \big(\boldsymbol{A}^*\boldsymbol{A}+\boldsymbol{D}^*\boldsymbol{D}\big)^{-1} &= \big[\boldsymbol{F}^*\big(|\boldsymbol{\Lambda}_{\boldsymbol{A}}|^2+|\boldsymbol{\Lambda}_{\boldsymbol{D}_h}|^2+|\boldsymbol{\Lambda}_{\boldsymbol{D}_v}|^2\big)\boldsymbol{F}\big]^{-1} \\ &= \boldsymbol{F}^*\big(|\boldsymbol{\Lambda}_{\boldsymbol{A}}|^2 + |\boldsymbol{\Lambda}_{\boldsymbol{D}_h}|^2 + |\boldsymbol{\Lambda}_{\boldsymbol{D}_v}|^2\big)^{-1}\boldsymbol{F}, \end{aligned} \tag{26}$$

where $|\boldsymbol{\Lambda}_{\boldsymbol{H}^{(j)}}|^2$ corresponds to the squaring of matrix $\boldsymbol{\Lambda}_{\boldsymbol{H}^{(j)}}$ diagonal values. So, just for the sake of clarification, we obviously have

$$\boldsymbol{A}^*\boldsymbol{A} = \boldsymbol{F}^*|\boldsymbol{\Lambda}_{\boldsymbol{A}}|^2\boldsymbol{F}. \tag{27}$$

Further, taking into account the definition of $\boldsymbol{D}$ in (12):

$$\boldsymbol{D}^*\boldsymbol{D} = \boldsymbol{D}_h^*\boldsymbol{D}_h + \boldsymbol{D}_v^*\boldsymbol{D}_v = \boldsymbol{F}^*(|\boldsymbol{\Lambda}_{\boldsymbol{D}_h}|^2 + |\boldsymbol{\Lambda}_{\boldsymbol{D}_v}|^2)\boldsymbol{F}, \tag{28}$$

and also, for some vector $\boldsymbol{u} = [\boldsymbol{u}_1^*\ \boldsymbol{u}_2^*]^* \in \mathbb{R}^{2N}$, we have

$$\boldsymbol{D}^*\boldsymbol{u} = \boldsymbol{D}_h^*\boldsymbol{u}_1 + \boldsymbol{D}_v^*\boldsymbol{u}_2 = \boldsymbol{F}^*(\boldsymbol{\Lambda}_{\boldsymbol{D}_h}^*\boldsymbol{F}\boldsymbol{u}_1 + \boldsymbol{\Lambda}_{\boldsymbol{D}_v}^*\boldsymbol{F}\boldsymbol{u}_2) \tag{29}$$

and surely, for some vector $\boldsymbol{a} \in \mathbb{R}^N$, we have

$$\boldsymbol{D} = \begin{bmatrix} \boldsymbol{D}_h \\ \boldsymbol{D}_v \end{bmatrix} \boldsymbol{a} = \begin{bmatrix} \boldsymbol{F}^*\boldsymbol{\Lambda}_{\boldsymbol{D}_h}\boldsymbol{F}\boldsymbol{a} \\ \boldsymbol{F}^*\boldsymbol{\Lambda}_{\boldsymbol{D}_v}\boldsymbol{F}\boldsymbol{a} \end{bmatrix}. \tag{30}$$

In relation to convergence conditions, in section II-A, Theorem 1: one has that $\boldsymbol{G} = [\boldsymbol{A}^*\boldsymbol{D}^*]^*$ is full column rank, due to $\boldsymbol{D}$ being the invertible discrete gradient operator, which can be seen in (12), where all its columns are linearly independent, so $\boldsymbol{G}$ is invertible as well; further, both assigned functions g_1 and g_2 - (20) and (21), respectively - belong to set $\Gamma_0(\mathcal{X})$; and we assume the set of minimizers is nonempty. Hence, ADMM convergence for the TV deblurring problem (10) is assured.

Lastly, cost of operator computation for this algorithm either incurs in $O(Nlog(N))$ if using FFT, or $O(N)$ for all others. In fact, vector sums, proximity operators (with $\boldsymbol{M}^*\boldsymbol{y}_m$ term being pre-computed for g_1 (22)) and matrix inversions (in the case of (22)) have $O(N)$ cost.

C. Super-Resolution Total Variation (SR-TV) Deblurring: Applying Proximal Algorithm I (from Generic Proximal Algorithm)

In [11], Algorithm 2 was already specified for the TV deblurring problem, however without the Super-Resolution component (i.e. operator $\boldsymbol{M} = \boldsymbol{I}_d$). So, now we adapt that approach for the Super-Resolution TV deblurring problem in

(10), which is much more ill-posed. In this report we will call this algorithm the *Proximal Algorithm I (ProxAlg_I)* . Hence, we now specify ($J = 1$):

$$\boldsymbol{z} \in \mathbb{R}^N, \qquad \boldsymbol{z} = \boldsymbol{x}, \tag{31}$$

$$\boldsymbol{u}^{(1)} \in \mathbb{R}^{2N}, \qquad (Dual\ Problem\ Var), \tag{32}$$

$$\boldsymbol{H}^{(1)} : \mathbb{R}^N \to \mathbb{R}^{2N}, \qquad \boldsymbol{H}^{(1)} = \boldsymbol{D}, \tag{33}$$

$$f_1 : \mathbb{R}^N \to \mathbb{R}, \qquad f_1(.) = \frac{1}{2}\|\boldsymbol{MA}\ . - \boldsymbol{y}_m\|^2, \tag{34}$$

$$f_2 : \mathbb{R}^N \to \{0\}, \qquad f_2(.) = 0, \tag{35}$$

$$g_1 : \mathbb{R}^{2N} \to \mathbb{R}, \qquad g_1(.) = \lambda_{TV}\|.\|_{iso}. \tag{36}$$

Further, we now formulate $\nabla f_1(.)$, $prox_{\sigma g_1^*}(.)$, and $prox_{\tau f_2}(.)$, for complete algorithm definition:

$$\nabla f_1(.) = \boldsymbol{A}^*\big(\boldsymbol{M}^*\boldsymbol{MA}\ . - \boldsymbol{M}^*\boldsymbol{y}_m\big), \tag{37}$$

$$prox_{\tau f_2}(.) = \boldsymbol{I}_d\ . , \tag{38}$$

and using $\big(u_{hi}, u_{vi}\big) \in \mathbb{R}^2$, the output of operator $\boldsymbol{D}$, we have that

$$\begin{aligned} prox_{\sigma g_1^*}(.) &: \big(u_{hi}, u_{vi}\big) \to \\ &\to \big(u_{hi}, u_{vi}\big)/\max\Big\{\sqrt{\big(u_{hi}\big)^2 + \big(u_{vi}\big)^2}/\lambda_{TV}, 1\Big\}, \end{aligned} \tag{39}$$

where all can be efficiently computed. Indeed, in section III-B, structures $\boldsymbol{M}^*\boldsymbol{M}$ and $\boldsymbol{M}^*\boldsymbol{y}_m$, which also appear in (37), have already been exploited; reducing, in this case, the former to a diagonal matrix product, and the latter to a pre-computation. Meanwhile, proximity operators (38) and (39) have already been presented in [11], plus they can be deduced based on the properties in [7]. Further, operators $\boldsymbol{A}$, $\boldsymbol{A}^*$, $\boldsymbol{D}$, $\boldsymbol{D}^*$ can all be computed here in the same way as they were in the previous chapter III-B for ADMM - using DFT factorization.

In relation to algorithm convergence, we notice that: f_1, f_2, $g_1 \in \Gamma_0(\mathcal{X})$; $\boldsymbol{H}^{(1)}$ is a bounded liner operator; f_1 is differentiable and its gradient is β-Lipschitz continuous (β defined ahead); and we assume the set of minimizers is nonempty. Hence, the algorithm is convergent towards a solution.

Moreover, we use Theorem 2 for convergence analysis, since $f_1(.) \neq 0$. So, this theorem can be used for choosing near optimum initial algorithm parameters, whilst making sure they comply with convergence criteria. Hence, we can now associate the value of τ to the chosen initial value of σ, such that

$$\tau < \frac{1}{\beta/2 + \sigma\|\boldsymbol{D}^*\boldsymbol{D}\|} \Rightarrow \tau = \frac{0.99}{\|\boldsymbol{A}\|^2/2 + 8\sigma}, \tag{40}$$

where we know that $\|\boldsymbol{D}^*\boldsymbol{D}\| \leq 8$ from [11] and [6]; and it can be proved that Lipschitz constant is $\beta = \|\boldsymbol{A}\|^2$, corresponding to the spectral component of convolution $\boldsymbol{A}$ (i.e. its eigenvalue) with the highest absolute value - since we will be using a low-pass filter, which can be normalized for unitary DC gain, one easily has that $\|\boldsymbol{A}\|^2 = 1$. So, according to Theorem 2, now we only have to calibrate Algorithm 2 for σ and ρ: with σ we only have to choose a value that leads to convergence, and then using equation (40) we get an optimal value for τ; with ρ we will have to choose it on a trial-and-error basis, though empirically, the best results tend to be near $\rho = 1$.

D. Super-Resolution Total Variation (SR-TV) Deblurring: Applying Chambolle-Pock (from Generic Proximal Algorithm)

As an alternative, we take the formulation in (7), assigning its functions differently to that of III-C, the previous section. So, if we define $f_1(.) = 0$, formulation (7) reverts to a relaxed version of Chambolle-Pock (CP) [6] [11], this was suggested by L. Condat [11] as an alternative, but not fully defined or implemented.

Taking into account Algorithm 2, and the Super-Resolution TV deblurring problem in (10), we now define another setup ($J = 2$):

$$\boldsymbol{z} \in \mathbb{R}^N, \qquad \boldsymbol{z} = \boldsymbol{x}, \tag{41}$$

$$\boldsymbol{u}^{(1)} \in \mathbb{R}^{N/S}, \qquad (Dual\ Problem\ Var), \tag{42}$$

$$\boldsymbol{u}^{(2)} \in \mathbb{R}^{2N}, \qquad (Dual\ Problem\ Var), \tag{43}$$

$$\boldsymbol{H}^{(1)} : \mathbb{R}^N \to \mathbb{R}^{N/S}, \qquad \boldsymbol{H}^{(1)}\ . = \boldsymbol{MA}\ . , \tag{44}$$

$$\boldsymbol{H}^{(2)} : \mathbb{R}^N \to \mathbb{R}^{2N}, \qquad \boldsymbol{H}^{(2)} = \boldsymbol{D}, \tag{45}$$

$$f_1 : \mathbb{R}^N \to \{0\}, \qquad f_1(.) = 0, \tag{46}$$

$$f_2 : \mathbb{R}^N \to \{0\}, \qquad f_2(.) = 0, \tag{47}$$

$$g_1 : \mathbb{R}^{N/S} \to \mathbb{R}, \qquad g_1(.) = \frac{1}{2}\|. - \boldsymbol{y}_m\|^2, \tag{48}$$

$$g_2 : \mathbb{R}^{2N} \to \mathbb{R}, \qquad g_2(.) = \lambda_{TV}\|.\|_{iso}. \tag{49}$$

Additionally, it is important to clarify some operators that originate from this setup. As defined before, $prox_{\sigma g_2^*}$ takes the same form as (39), and $prox_{\tau f_2}$ the same as (38). Besides, taking into account the definitions of proximity operator, plus the (Fenchel) conjugation property, present in [7], one can now define

$$prox_{\sigma g_1^*}(.) = \frac{1}{1+\sigma}\big[. - \sigma\boldsymbol{y}_m\big], \tag{50}$$

this result is fully explained in Appendix A. In addition, we have that operator $\boldsymbol{H}^{*(1)} = \boldsymbol{A}^*\boldsymbol{M}^*$, and also, operators $\boldsymbol{A}$, $\boldsymbol{A}^*$, $\boldsymbol{D}$, $\boldsymbol{D}^*$ can all be computed using DFT factorization.

In relation to convergence, we notice that again all $\boldsymbol{H}^{(j)}$ are linear and bounded; all function terms belong to $\Gamma_0(\mathcal{X})$; and we assume a nonempty set of minimizers. Since f_1 is not used, no further considerations are necessary.

Conversely to III-C, we now use Theorem 3 from *Generic Proximal Algorithm*, since $f_1(.) = 0$. Once more, it is possible to define a near optimal value for τ based upon choosing σ, while at the same time complying with convergence conditions. So, now we can define:

$$\tau \leq \frac{1}{\sigma\|\boldsymbol{A}^*\boldsymbol{A} + \boldsymbol{D}^*\boldsymbol{D}\|} \Rightarrow \tau = \frac{1}{\sigma\big(\|\boldsymbol{A}\|^2 + \|\boldsymbol{D}^*\boldsymbol{D}\|\big)}, \tag{51}$$

where values for $\|\boldsymbol{A}\|^2$ and $\|\boldsymbol{D}^*\boldsymbol{D}\|$ in III-C also apply here. Moreover, the above assignment for τ was obtained using the triangle inequality property for norms; the use of equality in this expression, and a value for ρ near 2 is based upon section II-B on parameter selection.

E. Super-Resolution Total Variation (SR-TV) Deblurring: Remarks on Generic Proximal Algorithm (Algorithm 2)

In terms of operator complexity costs for the *Generic Proximal Algorithm* in the SR-TV problem (10), as stated in

[11], these are all $O(N)$ or $O(Nlog(N))$, the latter associated with operators $\boldsymbol{A}$ and $\boldsymbol{D}$, due to their FFT implementations.

In addition, we should discuss the options made regarding function assignment in both sections III-C and III-D.

Firstly, any efficient implementation that solves SR-TV problem (10) needs to decouple the function term $\lambda_{TV}\|\boldsymbol{D}.\|_{iso}$ into efficient closed-form computations. As seen earlier, this decoupling, into $\lambda_{TV}\|.\|_{iso}$ and $\boldsymbol{D}$, leads to effective operators being used (e.g. the vector-soft$(.,.)$ proximal operator). Not doing so leads to a situation that is intractable, or numerical at best. Therefore, the only efficient way is to use the last term $\sum_{j=1}^{J} g_j(\boldsymbol{H}^{(j)}\boldsymbol{z})$ in (7), which singles out a *Forward-Backward* implementation - *FB* only uses f_1 and f_2, with $J = 0$.

Secondly, assigning the term $\frac{1}{2}\|\boldsymbol{M}(\boldsymbol{A}\ .) - \boldsymbol{y}_m\|^2$ to f_2 will lead to an intractable operator as well:

$$prox_{\tau f_2}(.) = (\boldsymbol{I}_d + \tau\boldsymbol{A}^*\boldsymbol{M}^*\boldsymbol{M}\boldsymbol{A})^{-1}\left[\tau\boldsymbol{A}^*\boldsymbol{M}^*\boldsymbol{y}_m +\ .\ \right], \tag{52}$$

since the inversion of $(\boldsymbol{I}_d + \tau\boldsymbol{A}^*\boldsymbol{M}^*\boldsymbol{M}\boldsymbol{A})$ is not possible in due time, mainly because there is no exploitable structure, much like a diagonal matrix or a DFT factorization.

Hence, we are left with only two options, which we described earlier in III-C (*Proximal Algorithm I*) and III-D (*Chambolle-Pock*). The first implementation is preferred, since it capitalizes on $\nabla f_1(.)$ for possible better convergence.

F. Super-Resolution Total Variation (SR-TV) Deblurring: Practical Results

In this section, we present the results for algorithm convergence, for the SR-TV (Super-Resolution Total Variation) problem in (10), all algorithms solve the same objective function.

Images suffer Gaussian blurring, with different sizes of kernel (KS) support; also, different scale factors (Ds) support sizes for SR are used - with $Ds = 1$ representing no SR, i.e. no down-sampling.

White Gaussian Noise was added at a level of 30 dB BSNR, for each image. Lowpass (linear) convolution operators have been normalized for unitary DC gain, i.e. $\|\boldsymbol{A}\| = 1$.

Input parameters have been tailored for all algorithms, to speed up convergence, taking into account the particular scenario (image, blurring and scale factor). In Appendix F, we can see the algorithm parameter selection process and results, which were then used for the experiments in this section, for SR-TV.

In relation to the TV deblurring parameter λ_{TV}, Cameraman uses 0.02, and all other use 0.2.

Results comprised grayscale images taken from the “USC SIPI Image Database”: Cameraman, Elaine and Man in .tiff format. Sizes are 256x256, 512x512, and 1024x1024 pixels respectively.

Accordingly, results will be presented for *Alternating Direction Method of Multipliers* (ADMM), *Chambolle-Pock* (CP), and *Proximal Algorithm I* (ProxAlg_I). These will be evaluated on the basis of their convergence regarding the objective function relative error, for the optimal solution. Algorithms ran until an error of 1% was reached. The optimal solution was

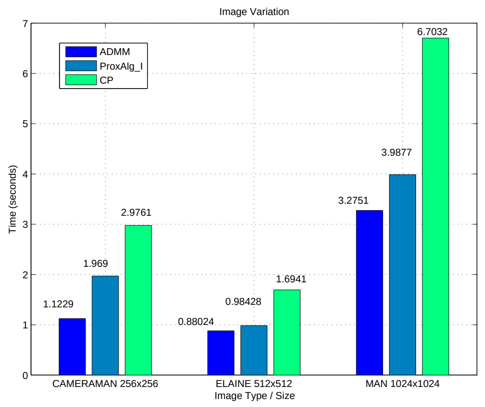


Fig. 1. Variation of Image size, and related times to convergence, until a 1% objective function $\Phi(.)$ relative error was reached. Each algorithm is presented: *ADMM*, *Proximal Algorithm I* (ProxAlg_I), *Chambolle-Pock* (CP). Constant values for this scheme: $Ds = 1$; $KS = 3$. Cameraman uses $\lambda_{TV} = 0.02$, while other images use 0.2.

obtained via a long-run of ADMM, for each scenario. ADMM ran until a tolerance of 10^{-20} was reached for its residuals, otherwise it ran 10,000 iterations maximum.

Lastly, in this section results were obtained using the ”Mighty4” computer available in the lab, with OS Linux, and MATLAB software package.

1) Case I – Image Size (Without Downsampling) - Fig.1: First we start by showcasing image size variation and its effect on convergence performance. Fig.1 follows the order of increasing image size, from Cameraman (256x256), through Elaine (512x512) to Man (1024x1024). Also, the results share the same scale factor size $Ds = 1$ and Gaussian Blur kernel size $KS = 3$. It is clear that generally CP has poor convergence compared with both ADMM and ProxAlg_I. ADMM always manages to reach the lower limit of 1% error first, thus having the lowest convergence time. Further, results are also dependent on the content of the image that we wish to apply deblurring to, not only its size.

2) Case II – Blur Size (Super-Resolution) - Fig.2: Now we reveal the Blur kernel size KS impact on algorithm convergence. Fig.2 results are for Man image, and scale factor size $Ds = 3$. Distinct Blur kernel sizes $KS = 3$ and $KS = 5$ pixels are tested. Also, there is no point in testing $KS = 1$, since only a normalized gain of 1 would be applied to the image pixels, without any consequent blurring. Further, the figure exposes the fact that as we increase Blur kernel size, convergence rates tend to slow down. Moreover, increasing Blur kernel size leads to clear divergence in rates, notably, ADMM and ProxAlg_I see their convergence time gap increased. On both scenarios CP rate is weaker, and ADMM has a clear edge on CP and ProxAlg_I, achieving the lowest convergence time.

3) Case III – Scale Factor (Super-Resolution) - Fig.3: Lastly, we now focus on scale factor size Ds influence on convergence. In Fig.3, the results pertain to Man image, Blur kernel size $KS = 5$, and variable scale factor size Ds: 1, 3 and 5 pixels. On all three scenarios CP and ProxAlg_I convergence times are evidently weaker than ADMM. Also, increasing the scale factor only hinders the convergence of

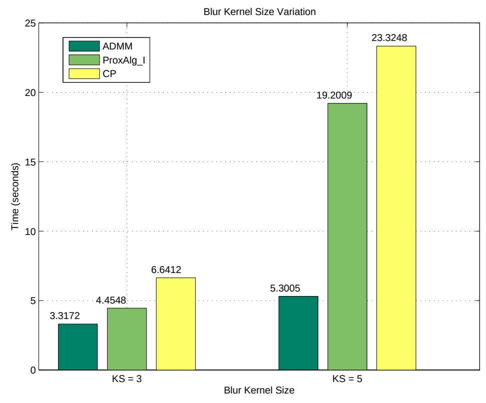


Fig. 2. Variation of (Gaussian) Blur kernel size KS, and related times to convergence, until a 1% objective function $\Phi(.)$ relative error was reached. Each algorithm is presented: *ADMM*, *Proximal Algorithm I* (ProxAlg_I), *Chambolle-Pock* (CP). Constant values for this scheme: Man (1024x1024) image; $Ds = 3$; $\lambda_{TV} = 0.2$.

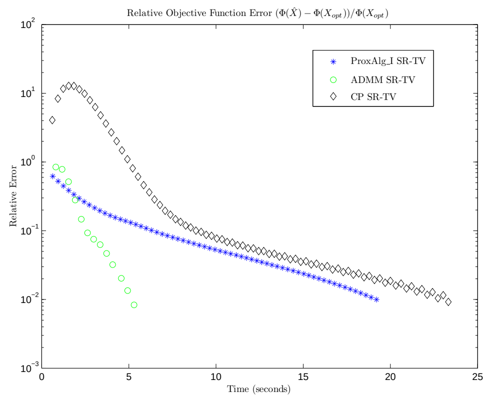


Fig. 4. Relative error of objective function $\Phi(.)$ vs. time, in one entire run until solution is obtained - convergence curve. For Man image (1024x1024), $Ds = 3$, $KS = 5$, $\lambda_{TV} = 0.2$. Algorithms are run until 1% objective function $\Phi(.)$ relative error is reached. Each algorithm is presented: *ADMM*, *Proximal Algorithm I* (ProxAlg_I), *Chambolle-Pock* (CP).

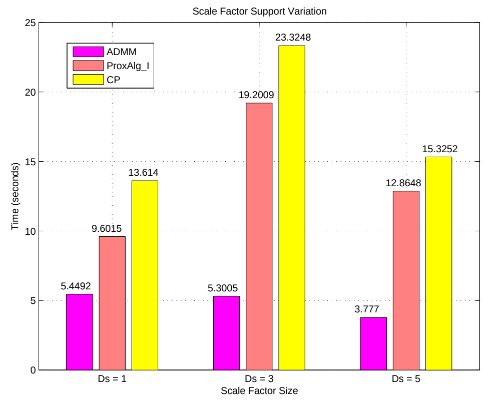


Fig. 3. Variation of Scale Factor Ds support size, and related times to convergence, until a 1% objective function $\Phi(.)$ relative error was reached. Each algorithm is presented: *ADMM*, *Proximal Algorithm I* (ProxAlg_I), *Chambolle-Pock* (CP). Constant values for this scheme: Man (1024x1024) image; $KS = 5$; $\lambda_{TV} = 0.2$.

these two algorithms, leaving ADMM relatively more or less unscathed (roughly the same convergence time).

4) Case IV – Lowest Convergence Scenario - Fig.4: In this specific case, we choose to portray in more detail the lowest convergence case for the previous sections III-F1, III-F2 and III-F3 - which by the way a very ill-posed scenario. So, this is an opportunity to showcase results for each iteration within a scenario, in other words, the objective function relative $\Phi(.)$ error vs. time, the convergence curve. The scenario is clearly Man image (1024x1024), $Ds = 3$, $KS = 5$ with $\lambda_{TV} = 0.2$. It can be seen in Fig.4.

Interestingly, we notice that ADMM converge is not exactly decreasing monotone, though monotonicity is not guaranteed by ADMM's convergence conditions. In fact, neither *Generic Proximal Algorithm's (Algorithm 2)* convergence conditions guarantees this.

Moreover, more results are presented for this scenario in Appendix C, though some results are only updated/ seen every 10 iterations, to avoid computational burden. The results include: ADMM μ assignment variation; ADMM residuals; duration of each iteration, for each algorithm, during their runtime; and blurred, plus deblurred images for each algorithm. It is noteworthy to say that although per iteration ADMM does take longer (duration), it offers much less iterations until convergence is attained.

IV. Applications in Inverse Problems in Imaging: Hyperspectral Image Segmentation (SMLR)

A. Hyperspectral Image Segmentation: Problem Definition (Sparse Multinomial Logistic Regression - SMLR)

1) Background: In this application, we tackle the problem of image segmentation for hyperspectral images, such as those captured from remote sensing instruments. Hyperspectral image data sets often bring up difficult processing problems, and particularly in segmentation. Learning high-dimensional distributions from a limited number of training samples (ill-posed problem) is still an active area of research.

Discriminative approaches, which learn the class distributions in high-dimensional spaces by inferring the boundaries between classes in the feature space, effectively tackle the aforementioned difficulties. Support vector machines (SVMs) are among the state-of-the-art discriminative techniques that can be applied to solve ill-posed classification problems [22], [23]. Conversely, multinomial logistic regression (MLR) [24] is an alternative approach to deal with ill-posed problems, which has the advantage of learning the class distributions themselves.

MLR, as a discriminative classifier, models the posterior distributions instead of the joint probability distributions. Further, the ability to deal with large input spaces efficiently, and to produce sparse solutions, is important in high-dimension ill-posed classification. Effective sparse MLR (SMLR) methods fulfil these requirements, and can be found in literature [25], [26]. Plus, different versions have already been implemented in hyperspectral image classification as well [25], [27], [28], [29].

More information on the issue at hand, problem formulations and definitions can be found in [25], [27], [26], [24]. In fact, we follow the same approach for the learning step. In this document, however, we will only focus on how to solve the learning problem using *Proximal Algorithms*, most notably *ADMM* and the *Generic Proximal Algorithm*, previously defined in sections II-A and II-B respectively.

Moreover, [25] already uses a version of ADMM, though different from that which is presented in this document. An experimental comparison in terms of convergence performance is given for both ADMM approaches - in Appendix D. We will also adopt the same name for the resulting algorithm: LORSAL (MLR via variable splitting and augmented Lagrangian).

2) Problem Formulation: So, let $\mathcal{S} \equiv \{1, \cdots, n\}$ denote a set of integers indexing the n pixels of a hyperspectral image; let $\mathcal{L} \equiv \{1, \cdots, K\}$ be a set of K labels; let $\boldsymbol{X} = \{\boldsymbol{X}_1, \cdots, \boldsymbol{X}_n\} \in \mathbb{R}^{d \times n}$ denote an image of d-dimensional feature vectors; let $\boldsymbol{y} = (y_1, \cdots, y_n) \in \mathcal{L}^n$ be an image of labels; and let $\mathcal{D}_L \equiv \{(\boldsymbol{X}_1, y_1), \cdots, (\boldsymbol{X}_L, y_L)\} \in (\mathbb{R}^d \times \mathcal{L})^L$ be a training set where L denotes the total number of available labelled samples. With the aforementioned definitions in place, the goal is to assign a label $y_i \in \mathcal{L}$ to each pixel $i \in \mathcal{S}$, based on the vector $\boldsymbol{X}_i$, resulting in an image of class labels $\boldsymbol{y}$. We call this labelling.

In segmentation the goal is to compute, based on the observed image $\boldsymbol{X}$, a partition $\mathcal{S} = \cup_i \mathcal{S}_i$ such that the pixels in each element of the partition share some common properties. Hence, segmentation is when some spatial prior $\boldsymbol{P}(\boldsymbol{y})$ is being considered on the $\boldsymbol{y}$ labels, whilst traditional classification does not assume this.

Further, we can define $\boldsymbol{Y}' \in \{0, 1\}^{(K-1) \times L}$ as being a matrix where its columns $\boldsymbol{Y}'_i$ represent the labellings provided in vector $\boldsymbol{y}$ (1 for the selected class, 0 for the others in the (pixel's) column). In addition, other models may be used to describe the feature space, in such manner, we can have $\phi(\boldsymbol{X}_i) \equiv [\phi_1(\boldsymbol{X}_i), \cdots, \phi_l(\boldsymbol{X}_i)]^* \in \mathbb{R}^l$ defining a vector of l fixed functions of the input, often termed *features*. Moreover, $\boldsymbol{W} \equiv [\boldsymbol{W}_1, \cdots, \boldsymbol{W}_K]^* \in \mathbb{R}^{l \times K}$ denotes the logistic regressors. However, since the distributions ahead in (53) do not depend on translations of the regressors $\boldsymbol{W}_K$, we can take $\boldsymbol{W}_K = \boldsymbol{0}$ and remove it from $\boldsymbol{W}$, such that $\boldsymbol{W} \equiv [\boldsymbol{W}_1, \cdots, \boldsymbol{W}_{K-1}]^* \in \mathbb{R}^{l \times (K-1)}$.

Notice however that $\phi(.)$ may be linear, with $\phi(\boldsymbol{X}_i) = [1, \boldsymbol{X}_{i1}, \cdots, \boldsymbol{X}_{id}]^*$, or otherwise nonlinear. In the nonlinear case, kernels are a relevant example and can be expressed by $\phi(\boldsymbol{X}_i) = [1, Kr_{\boldsymbol{X}_i, \boldsymbol{X}_1}, \cdots, Kr_{\boldsymbol{X}_i, \boldsymbol{X}_l}]^*$, where $Kr_{\boldsymbol{X}_i, \boldsymbol{X}_j} \equiv Kr(\boldsymbol{X}_i, \boldsymbol{X}_j)$ and $Kr(.,.)$ is some symmetric kernel function; kernels can improve data separability in the transformed space. In our report, we present results only for the Gaussian radial basis function (RBF) kernel (widely used in hyperspectral image classification), given by $Kr(\boldsymbol{X}, \boldsymbol{Z}) = \exp(-\|\boldsymbol{X} - \boldsymbol{Z}\|^2 / (2\rho^2))$. Now, if we denote γ as the dimension of $\phi(\boldsymbol{X})$, then we have $\gamma = d + 1$ for the linear case and $\gamma = L + 1$ for the RBF kernel (L being the dimension of the training set).

Therefore, modelling the posterior densities using an MLR, we get the following

$$\boldsymbol{P}(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{W}) = \exp(\boldsymbol{W}^* \phi(\boldsymbol{X})) \circ \\ 1 \Big/ \left(\boldsymbol{1}^*_{(k-1)} \exp(\boldsymbol{W}^* \phi(\boldsymbol{X})) + 1 \right)_{k=1}^{(K-1)}, \quad (53)$$

where $\boldsymbol{P}(\boldsymbol{Y}|\boldsymbol{W}, \boldsymbol{X}) \in \mathbb{R}^{(K-1) \times L}$ is a matrix with the densities for every class vs. pixel, such that probability densities for every pixel can be defined by vector column $\boldsymbol{P}_i(\boldsymbol{Y}_i|\boldsymbol{W}, \boldsymbol{X}_i)$; also, in (53), we have that $\circ$ is applying element-wise matrix product; $\exp(.)$ is element-wise; $(\boldsymbol{1}^*_{(k-1)} \exp(\boldsymbol{W}^* \phi(\boldsymbol{X})) + 1)_{k=1}^{(K-1)} \in \mathbb{R}^{(K-1) \times L}$ is a matrix where its columns (per pixel) have the same value - repeated $(K-1)$ times; and $\boldsymbol{1}^*_{(K-1)}$ is the transpose of a vector column of ones with size $(K-1)$.

Additionally, we can define the general Softmax function, which will be used in LORSAL (ADMM), as

$$\boldsymbol{P}(\boldsymbol{Y}|\boldsymbol{v}) = \exp(\boldsymbol{v}^*) \circ 1 \Big/ \left(\boldsymbol{1}^*_{(k-1)} \exp(\boldsymbol{v}^*) + 1 \right)_{k=1}^{(K-1)}, \quad (54)$$

where $\boldsymbol{P}(\boldsymbol{Y}|\boldsymbol{v}) \in \mathbb{R}^{(K-1) \times L}$, notice that $\boldsymbol{v}$ is transposed to fit the structure of ADMM, so some of the resulting expressions for LORSAL will be transposed as well. All the other operations in (53) apply.

Hence, assuming the data set $\boldsymbol{X}$, for every i-th pixel, is conditionally independent of the features given the labels, we have for the log-likelihood function

$$\ell(\boldsymbol{W}, \boldsymbol{Y}') = \sum_{i=1}^{L} \log \boldsymbol{Y}'^*_i \boldsymbol{P}_i(\boldsymbol{Y}_i|\boldsymbol{W}, \boldsymbol{X}_i) \quad (55)$$

$$= \sum_{i=1}^{L} \left(\sum_{k=1}^{K} \boldsymbol{Y}'_{ki} \boldsymbol{W}^*_k \phi(\boldsymbol{X}_i) - \log \sum_{k=1}^{K} \exp(\boldsymbol{W}^*_k \phi(\boldsymbol{X}_i)) \right), \quad (56)$$

where log takes in a single real value, and $\boldsymbol{Y}'_{ki}$ is an element of matrix $\boldsymbol{Y}'$ for indexes k and i.

In view of the above, densities $\boldsymbol{P}_i(\boldsymbol{Y}_i|\boldsymbol{X}_i)$ are modelled as MLRs, whose regressors are learned - either via *ADMM* or *Generic Proximal Algorithm*. On the other hand, prior $\boldsymbol{P}(\boldsymbol{y})$ on the labellings $\boldsymbol{y}$ will be described by a multilevel logistic (MLL) prior Markov Random Field, which encourages neighbouring pixels to have the same label. The process for the priors is out of this report's scope, but can be seen in detail in [25].

We are now ready to define the learning (optimization) problem, using the SMLR approach:

$$\hat{\boldsymbol{W}} = \underset{\boldsymbol{W} \in \mathbb{R}^{\gamma \times (K-1)}}{\operatorname{argmin}} \Phi(\boldsymbol{W}) \quad (57)$$

$$= \underset{\boldsymbol{W} \in \mathbb{R}^{\gamma \times (K-1)}}{\operatorname{argmin}} -\ell(\boldsymbol{W}, \boldsymbol{Y}') + \lambda_{SMLR} \|\boldsymbol{W}\|_1 \quad (58)$$

where $\|.\|_1$ is the L_1-norm, and λ_{SMLR} is the tunable regularization parameter. Indeed, $\|\boldsymbol{W}\|_1$ corresponds to a Laplacian prior promoting sparsity on $\boldsymbol{W}$, forcing many components to be zero. Therefore, selecting just a few kernel functions, and as a result enhancing the generalization capacity of the MLR regressor matrix.

Lastly, notice that though this is a convex problem, it is a difficult one because $\ell(\boldsymbol{W}, \boldsymbol{Y}')$ is nonquadratic, and the term $\|\boldsymbol{W}\|_1$ is not smooth.

B. Hyperspectral Image Segmentation: Applying ADMM

So, now using LORSAL (i.e. ADMM) to solve the optimization problem (58), we have the following assignment of variables and functions:

$$\boldsymbol{z} \in \mathbb{R}^{\gamma\times(K-1)}, \qquad \boldsymbol{z} = \boldsymbol{W}, \tag{59}$$
$$\boldsymbol{v}^{(1)} \in \mathbb{R}^{L\times(K-1)}, \qquad \boldsymbol{v}^{(1)} = \boldsymbol{H}^{(1)}\boldsymbol{W}, \tag{60}$$
$$\boldsymbol{H}^{(1)} \in \mathbb{R}^{L\times\gamma}, \qquad \boldsymbol{H}^{(1)} = \phi(\boldsymbol{X})^*, \tag{61}$$
$$\boldsymbol{v}^{(2)} \in \mathbb{R}^{\gamma\times(K-1)}, \qquad \boldsymbol{v}^{(2)} = \boldsymbol{H}^{(2)}\boldsymbol{W}, \tag{62}$$
$$\boldsymbol{H}^{(2)} \in \mathbb{R}^{\gamma\times\gamma}, \qquad \boldsymbol{H}^{(2)} = \boldsymbol{I}_d, \tag{63}$$
$$g_1 : \mathbb{R}^{L\times(K-1)} \to \mathbb{R}, \qquad g_1(.) = -\ell(\ .,\boldsymbol{Y}'), \tag{64}$$
$$g_2 : \mathbb{R}^{\gamma\times(K-1)} \to \mathbb{R}, \qquad g_2(.) = \lambda_{SMLR}\|\cdot\|_1. \tag{65}$$

Moreover, now we have for step 4 of Algorithm 1 the following:

$$\boldsymbol{W}_{k+1} = (\phi(\boldsymbol{X})\phi(\boldsymbol{X})^* + \boldsymbol{I}_d)^{-1}\Big[\phi(\boldsymbol{X})(\boldsymbol{v}_k^{(1)} + \boldsymbol{d}_k^{(1)}) + (\boldsymbol{v}_k^{(2)} + \boldsymbol{d}_k^{(2)})\Big]. \tag{66}$$

Further, notice that $\boldsymbol{R_X} = \phi(\boldsymbol{X})\phi(\boldsymbol{X})^*$ is the covariance matrix in the feature space, since we are using a (Gaussian) RBF kernel for feature representation. Hence, by definition, we know that $\boldsymbol{R_X}$ is a real symmetric matrix, and positive semi-definite as well. Taking this into account, we can factorize $(\boldsymbol{R_X} + \boldsymbol{I}_d)$ into a more efficient structure, in order to compute its inverse.

Thus, using eigen-decomposition we know that $\boldsymbol{R_X} = \boldsymbol{U_{R_X}}\boldsymbol{\Lambda_{R_X}}\boldsymbol{U}^*_{\boldsymbol{R_X}}$, where $\boldsymbol{U_{R_X}}$ is the unitary matrix with the (orthonormal) eigen-vectors, and $\boldsymbol{\Lambda_{R_X}}$ is the diagonal matrix containing the eigen-values. So now we write

$$(\boldsymbol{R_X} + \boldsymbol{I}_d)^{-1} = \boldsymbol{U_{R_X}}(\boldsymbol{\Lambda_{R_X}} + \boldsymbol{I}_d)^{-1}\boldsymbol{U}^*_{\boldsymbol{R_X}}, \tag{67}$$

which is helpful in the computation of the inverse, since the matrix might be high-dimensional, depending on the dimension of the feature space (i.e. dimension γ). However, since this matrix does not depend on any changing values, it can be pre-computed and then used repeatedly.

Further, it becomes important to define how to compute the proximity operator for g_1 and g_2.

Firstly, starting with g_1, we can define the operator as

$$prox_{g_1/\mu}(.) = \underset{\boldsymbol{t}\in\mathbb{R}^{\gamma\times(K-1)}}{\operatorname{argmin}} \frac{\mu}{2}\|\ . - \boldsymbol{t}\|^2 + g_1(\boldsymbol{t}), \tag{68}$$

however, the above optimization problem is still a difficult one to solve, since $g_1(.) = -\ell(\ .,\boldsymbol{Y}')$ although convex and smooth, it is non-quadratic and often large. In fact, there is no closed-form solution to the problem. Yet, we can approximate the solution, in a way that is computationally efficient. We can tackle the problem by replacing $-\ell(\ .,\boldsymbol{Y}')$ with successive iterations over a quadratic upper bound, therefore applying the MM (Majorize-Minimization) algorithm to the proximity operator in (68), [30]. For the m-th iteration of MM algorithm, the bound is given by

$$-\ell(\boldsymbol{t},\boldsymbol{Y}') \leq -\ell(\boldsymbol{t}_m,\boldsymbol{Y}') + (\boldsymbol{t}-\boldsymbol{t}_m)\,\boldsymbol{g}'(\boldsymbol{t}_m,\boldsymbol{Y}') + \frac{1}{2}(\boldsymbol{t}-\boldsymbol{t}_m)\,\boldsymbol{B}'\,(\boldsymbol{t}-\boldsymbol{t}_m)^* = Q'(\boldsymbol{t}|\boldsymbol{t}_m), \tag{69}$$

where $Q'(\boldsymbol{t}|\boldsymbol{t}_m)$ is the bound, otherwise known as surrogate function in the MM algorithm. Moreover, $\boldsymbol{B}' = 1/2[\boldsymbol{I}_d - \boldsymbol{1}\boldsymbol{1}^*/K]$ is the second-order (Hessian) upper bound on $-\ell(\boldsymbol{t}_m,\boldsymbol{Y}')$, and $\boldsymbol{g}'(.,\boldsymbol{Y}') = (\boldsymbol{P}(\boldsymbol{Y}|.) - \boldsymbol{Y}')$ is the gradient of $-\ell(\boldsymbol{t}_m,\boldsymbol{Y}')$ - these are modified versions of the equations in [24], [25], [26]. In the previous formulations, symbol $\boldsymbol{1}$ denotes a vector column of ones (dimension K), and $\boldsymbol{P}(\boldsymbol{Y}|.)$ denotes the general Softmax function, in (54).

So, straightaway we can combine the surrogate function in (69) and the operator in (68), following the MM algorithm principle. Hence, for the m-th iteration we have

$$\boldsymbol{t}_{m+1} = \big[prox_{g_1/\mu}(.)\big]_{m+1} \tag{70}$$
$$= \underset{\boldsymbol{t}\in\mathbb{R}^{\gamma\times(K-1)}}{\operatorname{argmin}} \frac{\mu}{2}\|\ . - \boldsymbol{t}\|^2 + Q'(\boldsymbol{t}|\boldsymbol{t}_m), \tag{71}$$

such that replacing $Q'(\boldsymbol{t}|\boldsymbol{t}_m)$ by the upper bound expression found in (69), and without the terms that not depend on $\boldsymbol{t}$, we therefore have

$$\big[prox_{g_1/\mu}(.)\big]_{m+1} = \underset{\boldsymbol{t}\in\mathbb{R}^{\gamma\times(K-1)}}{\operatorname{argmin}} \frac{\mu}{2}\|\ . - \boldsymbol{t}\|^2 + \boldsymbol{t}\left(\boldsymbol{g}'(\boldsymbol{t}_m,\boldsymbol{Y}') - \boldsymbol{B}'\boldsymbol{t}_m^*\right) + \frac{1}{2}\boldsymbol{t}\boldsymbol{B}'\boldsymbol{t}^*. \tag{72}$$

Now the formulation above defines a quadratic problem. To find the minimum we take the first-order derivative, equate to zero vector, and solve for $\boldsymbol{t}$

$$\boldsymbol{0} = \mu\,(\ . - \boldsymbol{t})^* + \boldsymbol{g}'(\boldsymbol{t}_m,\boldsymbol{Y}') - \boldsymbol{B}'\boldsymbol{t}_m^* + \boldsymbol{B}'\boldsymbol{t}^*, \tag{73}$$

which after some algebraic manipulation becomes

$$\boldsymbol{t}_{m+1} = \big[prox_{g_1/\mu}(.)\big]_{m+1} = \big[\mu\ . - \boldsymbol{g}'(\boldsymbol{t}_m,\boldsymbol{Y}')^* + \boldsymbol{t}_m\boldsymbol{B}'\big]\left(\mu\boldsymbol{I}_d + \boldsymbol{B}'\right)^{-1}, \tag{74}$$

which can be computed several times for better precision, by feeding the result for the m-th iteration back into the expression - MM algorithm approximation. Nonetheless, in reality, for this application, we will only need one iteration to get a good performance.

For the sake of clarity, in step 7 of Algorithm 1, for $\boldsymbol{v}_k^{(1)}$ update, equation (74) is used, with MM algorithm. Accordingly, $\boldsymbol{t}_{m+1}$ and $\boldsymbol{t}_m$, in (74), become $(\boldsymbol{v}_k^{(1)})_{m+1}$ and $(\boldsymbol{v}_k^{(1)})_m$ respectively. Initial values for the MM, in the k-th iteration of Algorithm 1, should be $\boldsymbol{v}_{k-1}^{(1)}$; and for the $k=1$ iteration we should get a warm start with $\boldsymbol{v}_1^{(1)} = \phi(\boldsymbol{X})^*\boldsymbol{W}_{init}$.

Secondly, we now move on to define the proximity operator for $g_2(.)$, which is straightforward and has been described in literature before [7], [31], [25]. In fact, it translates into a well known operator, the soft-thresholding operator - i.e. $soft-thres(.,.)$. Hence, we have

$$prox_{g_2/\mu}(.) = soft{-}thres(\lambda_{SMLR}/\mu,.\ ) : (t_c)_{c=1}^{C} \in \mathbb{R} \to sgn\,(t_c)\,.\,(|t_c| - \lambda_{SMLR}/\mu)^+, \tag{75}$$

where the above expression is to be understood component-wise regarding the input of the operator - matrix or vector. Thus, C represents the cardinality of input $\boldsymbol{t}$, c the index of each element t_c of $\boldsymbol{t}$, sgn is the sign function, $(.)^+$ is the positive part, $|.|$ is the absolute value operator, and λ_{SMLR}/μ is considered the input constant of this operator.

Lastly, the initial estimate of $\boldsymbol{W}$, i.e. $\boldsymbol{W}_{init}$, can be computed in the first iteration through the standard *Tikhonov Regularization* method, otherwise known as *Ridge Regression* in statistics. Thus, we have the problem

$$\boldsymbol{W}_{init} = \underset{\boldsymbol{W}\in\mathbb{R}^{\gamma\times(K-1)}}{\text{argmin}} \ \|\boldsymbol{Y}^{tik} - \boldsymbol{H}^{(1)}\boldsymbol{W}\|^2 + \beta\|\boldsymbol{W}\|^2, \quad (76)$$

where β is the regularization parameter, obviously $\boldsymbol{H}^{(1)} = \phi(\boldsymbol{X})^*$, and whose solution is therefore

$$\boldsymbol{W}_{init} = (\boldsymbol{R}_{\boldsymbol{X}} + \beta\boldsymbol{I}_d)^{-1}\phi(\boldsymbol{X})\boldsymbol{Y}^{tik}, \quad (77)$$

where β was chosen to be $\beta = 10^{-5}$, for yielding a good initial result. Moreover, in this case, $\boldsymbol{Y}^{tik}$ entries are defined as 10 if pixel is in the class, and -10 if the pixel is outside the class - instead of 1, 0 respectively, as defined for $\boldsymbol{Y}'$.

In terms of algorithm parameters, μ and α are tunable, as in the previous application SR-TV III-B.

In relation to algorithm convergence, it can be proven using Theorem 1, in section II-A, such that in this case: one has that $\boldsymbol{G} = [\phi(\boldsymbol{X}) \ \ \boldsymbol{I}_d]^*$ is obviously full column rank, and thus invertible, even though $\boldsymbol{H}^{(1)} = \phi(\boldsymbol{X})^*$ may not be invertible, such is assured by $\boldsymbol{H}^{(2)} = \boldsymbol{I}_d$; also, both assigned functions g_1 and g_2 belong to set $\Gamma_0(\mathcal{X})$; and the set of minimizers is nonempty. Hence, LORSAL (i.e. ADMM) convergence for problem (58) is assured.

Finally, computational cost per iteration is determined by the same value found in [25]. It can be shown that this corresponds to $O\left(K\gamma(L+\gamma)\right)$ per iteration. Further, if we now assume that the Gaussian RBF kernel is being used, and so $\gamma = L+1$, then we have $O\left(K\gamma^2\right)$, which is precisely the value found in [25].

C. Hyperspectral Image Segmentation: Applying Forward-Backward (from Generic Proximal Algorithm)

Here, we now apply the Forward-Backward for SMLR, i.e. the SMLR_FB algorithm. There is not much point in applying the *Chambolle-Pock*, since it is preferable to use ∇f_1 when available for better convergence, check section II-B for more detail. Therefore, we now solve the optimization problem (58), for $J = 0$, with the following assignment

$$\boldsymbol{z} \in \mathbb{R}^{\gamma\times(K-1)}, \qquad \boldsymbol{z} = \boldsymbol{W}, \quad (78)$$

$$f_1 : \mathbb{R}^{\gamma\times(K-1)} \to \mathbb{R}, \qquad f_1(.) = -\ell(\ .,\boldsymbol{Y}'), \quad (79)$$

$$f_2 : \mathbb{R}^{\gamma\times(K-1)} \to \mathbb{R}, \qquad f_2(.) = \lambda_{SMLR}\|\ .\ \|_1. \quad (80)$$

Now, in order to implement FB we need to define $\nabla f_1(.)$ and $prox_{\tau f_2}(.)$.

Firstly, the gradient of f_1 has been extensively documented in literature, and for $\boldsymbol{W}$ is given by

$$\begin{aligned}\nabla f_1(\boldsymbol{W}) &= \nabla(-\ell(\boldsymbol{W},\boldsymbol{Y}')) = -\boldsymbol{g}(\boldsymbol{W},\boldsymbol{Y}') && (81)\\ &= \sum_{i=1}^{\mathcal{S}} \left(\boldsymbol{P}_i(\boldsymbol{Y}_i|\boldsymbol{W},\boldsymbol{X}_i) - \boldsymbol{Y}'_i\right) \otimes \phi(\boldsymbol{X}_i) && (82)\\ &= \phi(\boldsymbol{X})\left[\boldsymbol{P}(\boldsymbol{Y}|\boldsymbol{W},\boldsymbol{X}) - \boldsymbol{Y}'\right]^*, && (83)\end{aligned}$$

where $\boldsymbol{g}(\boldsymbol{W},\boldsymbol{Y}')$ has been defined in [24] [25] [26], and $\otimes$ is the Kronecker product.

Secondly, as previously defined in section IV-B, the proximity operator for $f_2(.)$ is also defined by the soft-thresholding operator:

$$\begin{aligned}&prox_{\tau f_2}(.) = soft{-}thres(\tau\lambda_{SMLR},.\ ) :\\ &\qquad (t_c)_{c=1}^C \in \mathbb{R} \to sgn\,(t_c)\,.\,(|t_c| - \tau\lambda_{SMLR})^+ . \quad (84)\end{aligned}$$

Furthermore, we shall use the same warm-start method as defined in the previous section for the first estimate of $\boldsymbol{W}$, i.e. $\boldsymbol{W}_{init}$ - see *Ridge Regression* in (77).

Once again, regarding algorithm convergence and parameter selection, we must resort to the theorems exposed in section II-B (for *Generic Proximal Algorithm*). In this case, since $f_1(.) \neq 0$, we must use the conditions in Theorem 2. In view of $J = 0$, we therefore define the first condition to be

$$\tau\frac{\beta}{2} \leq 1 \Leftrightarrow \tau \leq \frac{2}{\beta} \Rightarrow \tau = 0.99\frac{2}{\beta}, \quad (85)$$

and at the same time we solved it for τ, giving us the optimum step value - the highest value within the convergence restriction. Again, β is the Lipschitz constant defined for ∇f_1, where $f_1(.) = -\ell(\boldsymbol{W},\boldsymbol{Y})$, which is given by

$$\beta = \frac{1}{4}\|\phi(\boldsymbol{X})\phi(\boldsymbol{X})^*\| = \frac{1}{4}\|\boldsymbol{R}_{\boldsymbol{X}}\|, \quad (86)$$

where proof is given in Appendix B. Further, in the *Forward-Backward* implementation of Algorithm 2, σ, dual variables $\boldsymbol{u}^{(j)}$, operators $\boldsymbol{H}^{(j)}$ and $prox_{\sigma g_j^*}(.)$, are not used since $J = 0$. On the contrary, relaxation step ρ is still tunable, with $\rho \in]0,1]$, though recommended values tend to be near 1, as will be shown. Moreover, algorithm convergence is guaranteed since all conditions in II-B hold up here: $f_1, f_2 \in \Gamma_0(\mathcal{X})$; f_1 is differentiable and its gradient is β-Lipschitz continuous; the set of minimizers is nonempty.

In terms of computational complexity, the cost per iteration can be shown to be $O\left(K\gamma L\right)$. Further assuming that we are using an RBF kernel, with $\gamma = L+1$, we then have that $O(K\gamma^2)$, which is the same value found for LORSAL, in section IV-B.

D. Hyperspectral Image Segmentation (SMLR): Practical Results

So now, in this section, we will be presenting the convergence curves for both LORSAL (i.e. *ADMM*) and SMLR_FB (i.e. *Forward-Backward* for SMLR), all sharing the same objective function (58).

The results pertain to a *Hyperspectral Image Segmentation* problem, for $K = 3$ classes (labels) of material mixtures, using USGS_1995_Library.mat data library (from United States

Geological Survey), which contains the spectral information about the materials. Number of samples per class (label) k in the training set is $L_k = 100$, such that $L = 300$. Further, each class is composed of 10 different materials, with an angle of $10°$ between pairs (spectral components). In addition, material mixtures for each class (label) were built using Dirichlet statistical mixtures. Hence defining the spectral *feature* vectors. Moreover, label images $\boldsymbol{y}$ obey an MLL prior $\boldsymbol{P}(\boldsymbol{y})$ with a second order neighbourhood, while image size is $n = 128 \times 128$.

Algorithms ran until an error of 1% was reached for their objective function relative error, or a maximum of 200,000 iterations was attained. The optimal value was provided by LORSAL running 200,000 iterations.

Further, results in this section were obtained using a PC running a MATLAB 2010a package, with OS Windows 7 (64 bits). The PC has the following specifications: 8 GB of RAM, Intel Core i7 at 3.4 Ghz.

In relation to algorithm parameters, the initial ADMM μ used was borrowed from the previous implementation of LORSAL, i.e. $\mu = \lambda_{SMLR}/10$, [25]. It seems to provide both good results and convergence, enough for the comparison in this section. Plus, ADMM's dynamic μ assignment only provides marginal gains in terms of convergence in this example. Further, α was defined with value $\alpha = 1.65$, since good convergence results were obtained.

On the other hand, the only parameter up for selection in SMLR_FB is ρ, which is best served with $\rho = 1$, as will be revealed ahead.

Meanwhile, λ_{SMLR} is defined as $\lambda_{SMLR} = 0.001$, also taken from the previous implementation of LORSAL, which had already been optimized for problem (58) [25]. Both algorithms, LORSAL and SMLR_FB, will be using it.

Moreover, a comparison between the original LORSAL implementation, which is also ADMM, and the new LORSAL, which is presented in this document, is revealed in Appendix D. It can be seen that both are very similar in terms of convergence towards the same solution, though the implementation provided in this report is faster - which also implements the α relaxation step, defined in section II-A, something not available in the traditional version of LORSAL [25].

1) Comparison between ADMM and Forward-Backward: In this section we present the convergence curve results for SMLR_FB and LORSAL. The results can be seen in Fig.5; and the input parameters, for the algorithms, follow those described in the previous section.

It is obvious that LORSAL solution is far superior in terms of convergence. Conversely, SMLR_FB seems to have a very low convergence rate, even with optimized parameters.

Although the curve seems static, a closer look reveals that convergence for SMLR_FB is actually happening, but at very slow rates indeed. Something easily seen in next section's results (see Fig.6).

Lastly, more results concerning this scenario are revealed in Appendix E. In there we can see the image segmentation results, the original data containing the real material segmentation, and the noisy spectral measurement to be fed into the algorithm for segmentation. Further, in the captions

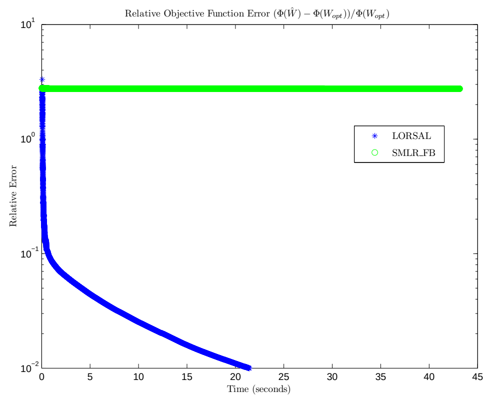


Fig. 5. Relative error of objective function $\Phi(.)$ vs. time, in one entire run until solution is obtained - convergence curve. For USGS_1995_Library.mat data and 3 possible classes, $\lambda_{SMLR} = 0.001$, $\mu = 0.001/10$, $\alpha = 1.65$, $\rho = 1$. Algorithms are run until 1% objective function $\Phi(.)$ relative error is reached. Each algorithm is presented: *ADMM* (LORSAL), *Forward-Backward* (SMLR_FB).

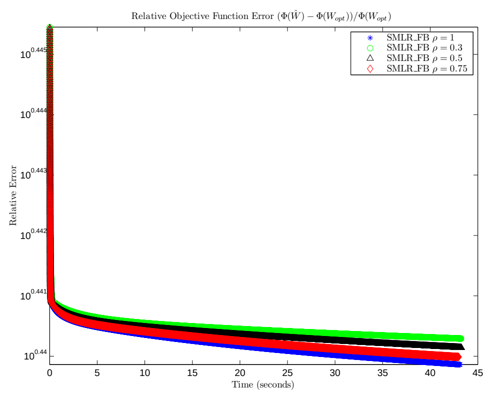


Fig. 6. Relative error of objective function $\Phi(.)$ vs. time, in one entire run until solution is obtained - convergence curve. For USGS_1995_Library.mat data and 3 possible classes, $\lambda_{SMLR} = 0.001$. Algorithms are run until 1% objective function $\Phi(.)$ relative error is reached. For *Forward-Backward* (SMLR_FB) each ρ value is presented: $\rho = 1$, $\rho = 0.75$, $\rho = 0.5$, $\rho = 0.3$.

of the resulting classification figures, for both LORSAL and SMLR_FB, accuracy values for training and testing data are provided. As can be seen, very accurate values were achieved.

2) Best choice of ρ for Forward-Backward (FB) in SMLR: Now, this section presents the results for the best ρ in SMLR_FB, see Fig.6. This is the only parameter that is tunable for SMLR_FB.

Further, it can be seen that the best result in terms of convergence comes from choosing $\rho = 1$.

So, just like in *Proximal Algorithm I*, which uses gradient ∇f_1 of *Generic Proximal Algorithm* (Algorithm 2), results and experience points to a value of $\rho = 1$. This can be checked in Appendix F, which reveals parameter selection results for the previous problem SR-TV, in (10).

V. Conclusion

In this report, our work focused on the implementation of *Proximal Algorithms* with focus on applications to inverse problems in imaging. We studied ill-posed problems, which often resort to nonsmooth regularization, and therefore are a prime case for the use of *Proximal Algorithms*.

The first application was Super-Resolution Total Variation Deburring (SR-TV), meanwhile, the second focused on Hyperspectral Image Segmentation (SMLR).

Two literature relevant algorithms were presented, *Alternating Direction Method of Multipliers* (ADMM) and the *Generic Proximal Algorithm* by L. Condat. The latter possesses specific implementations, which are well know, such as *Forward-Backward* and *Chambolle-Pock*. Moreover, ADMM is a *Second-Order method* (with curvature information), while all implementations of *Generic Proximal Algorithm* are *First-Order methods*.

For both algorithms, in all applications, we used optimum values for the tunable parameters therein. Also, the appropriate and best implementations of each algorithm were provided, for each application. These were guaranteed convergence, by related theorems, and computational cost analysis was also produced.

In fact, per iteration, ADMM will almost always register the same or more computational cost, something confirmed experimentally in SR-TV (check Appendix C). However, regarding convergence rates, it was proven experimentally that ADMM tends to have superior performance, in both applications. Typically, the more ill-posed the problem, the more advantageous the convergence rate offered by ADMM - SR-TV results are obvious in this respect, and in the SMLR problem, it was dominated by ADMM convergence rates.

Further, we conclude that the usage of *Second-Order Proximal Algorithms* (e.g. ADMM) is clearly beneficial in terms of convergence, against *First-Order* ones, if the problem at hand allows for their implementation.

By and large, ADMM offers great flexibility, with its variable splitting process. Further, improvement of ADMM convergence rates is obtained with over-relaxation (α step), dynamic μ assignment, and selection of initial μ. In addition, specific problem structure exploitation boosts performance - such is the case of function and variable assignment, and simultaneously matrix factorization (e.g. DFT decomposition).

Generally, the downside of ADMM might be increased memory usage, due to the expanding nature of auxiliary variables. Though in most problems this should not be an issue, usually since processing power seems to be the bottleneck.

Lastly, the *Generic Proximal Algorithm*, as stated by its convergence theorems in section II-B, needs to have a bounded norm of the sum of its linear operators. Conversely, ADMM does not present this dependency, which is definitely an advantage.

APPENDIX A
OPERATOR $prox_{\sigma g_1^*}(.)$, FOR CHAMBOLLE-POCK (CP) IN SR-TV

In here, we demonstrate how to get expression (50) for proximity operator $prox_{\sigma g_1^*}(.)$, in Chambolle-Pock (CP) implementation for SR-TV, section III-D.

The proximity operator can be introduced according to its definition

$$prox_{\sigma g_1^*}(.) = \underset{\boldsymbol{\theta}\in\mathbb{R}^{2N}}{\operatorname{argmin}} \frac{1}{2\sigma}\|. - \boldsymbol{\theta}\|^2 + g_1^*(\boldsymbol{\theta}), \tag{87}$$

where $g_1(.)$ was defined in section III-D as

$$g_1(.) = 1/2\| \, . - \boldsymbol{y}_m\|^2. \tag{88}$$

On the other hand, using the (Fenchel) conjugation property [7], we can write an alternative expression based on $prox_{g_1/\sigma}$ as

$$prox_{\sigma g_1^*}(.) = \; . - \sigma prox_{g_1/\sigma}\left(\frac{.}{\sigma}\right), \tag{89}$$

moreover, proximity operator $prox_{g_1/\sigma}\,(\,./\sigma)$ according to the definition can be written as

$$prox_{g_1/\sigma}\,(\,./\sigma) = \underset{\boldsymbol{\theta}\in\mathbb{R}^{2N}}{\operatorname{argmin}} \frac{\sigma}{2}\left\|\frac{.}{\sigma} - \boldsymbol{\theta}\right\|^2 + g_1(\boldsymbol{\theta}) \tag{90}$$

$$= \underset{\boldsymbol{\theta}\in\mathbb{R}^{2N}}{\operatorname{argmin}} \frac{\sigma}{2}\left\|\frac{.}{\sigma} - \boldsymbol{\theta}\right\|^2 + \frac{1}{2}\left\|\boldsymbol{\theta} - \boldsymbol{y}_m\right\|^2, \tag{91}$$

and now to find the minimum above we take the first-order derivative, equate to zero vector, and solve for $\boldsymbol{\theta}$

$$\mathbf{0} = \sigma\left(\boldsymbol{\theta} - \frac{.}{\sigma}\right) + (\boldsymbol{\theta} - \boldsymbol{y}_m), \tag{92}$$

after some algebraic manipulation we get

$$prox_{g_1/\sigma}\,(\,./\sigma) = \boldsymbol{\theta}_{min} = \frac{1}{1+\sigma}\left[\,. + \boldsymbol{y}_m\right]. \tag{93}$$

Now, replacing the above result (93) into expression (89) we have

$$prox_{\sigma g_1^*}(.) = \; . - \frac{\sigma}{1+\sigma}\left[\,. + \boldsymbol{y}_m\right] \tag{94}$$

$$= \frac{1}{1+\sigma}\left[\,. - \sigma\boldsymbol{y}_m\right].\square \tag{95}$$

APPENDIX B
β-LIPSCHITIZ CONSTANT OF SMLR_FB

In here, we reveal how to obtain the expression defined in (86), the β-Lipschitz constant, for gradient $\nabla f_1(.)$. So, according to the definition β is defined as

$$\|\nabla f_1(\boldsymbol{X}) - \nabla f_1(\boldsymbol{X}')\| \leq \beta\|\boldsymbol{X} - \boldsymbol{X}'\|, \quad \forall \boldsymbol{X}, \boldsymbol{X}' \in \mathbb{R}^{\gamma\times K}, \beta > 0, \tag{96}$$

notice that $\boldsymbol{X}$ and $\boldsymbol{X}'$ share their dimension with $\boldsymbol{W}$, the regressor matrix.

Now, starting with the left term of (96), and using the Taylor series expansion upon $\boldsymbol{X}'$ for function ∇f_1, we get

$$\|\nabla f_1(\boldsymbol{X}) - \nabla f_1(\boldsymbol{X}')\| = = \left\|\int_0^1 \nabla^2 f_1\left(t\boldsymbol{X} + (1-t)\boldsymbol{X}'\right)^*\left(\boldsymbol{X} - \boldsymbol{X}'\right)dt\right\|, \tag{97}$$

notice that only one term of the series was used (i.e $\nabla f_1(\boldsymbol{X}')$), along with the remainder, which is represented on the right of the above equation. Also, $\nabla^2 f_1$ is obviously the first-order derivative of ∇f_1. Further, we can write the following

$$\left\| \int_0^1 \nabla^2 f_1 \left(t\boldsymbol{X} + (1-t)\boldsymbol{X}'\right)^* \left(\boldsymbol{X} - \boldsymbol{X}'\right) dt \right\| \leq \quad (98)$$

$$\leq \int_0^1 \left\| \nabla^2 f_1 \left(t\boldsymbol{X} + (1-t)\boldsymbol{X}'\right)^* \left(\boldsymbol{X} - \boldsymbol{X}'\right) \right\| dt \leq \quad (99)$$

$$\leq \|\boldsymbol{X} - \boldsymbol{X}'\| \int_0^1 \left\| \nabla^2 f_1 \left(t\boldsymbol{X} + (1-t)\boldsymbol{X}'\right) \right\| dt \leq \quad (100)$$

$$\leq L_c \|\boldsymbol{X} - \boldsymbol{X}'\|, \quad (101)$$

where L_c is just a constant serving as an upper bound for $\int_0^1 \left\| \nabla^2 f_1 \left(t\boldsymbol{X} + (1-t)\boldsymbol{X}'\right) \right\| dt$. In fact, given the structure of expression (96) we know that $L_c = \beta$. So, such upper bound L_c can be given by

$$L_c = \|\boldsymbol{B}\|, \quad (102)$$

where $\boldsymbol{B}$ is the upper bound function for the second-order derivative of f_1. $\boldsymbol{B}$ has already been introduced, and can be found in [24], [25], [26]. It is defined by $\boldsymbol{B} = 1/2\left[\boldsymbol{I}_d - \boldsymbol{1}\boldsymbol{1}^*/K\right] \otimes \boldsymbol{X}\boldsymbol{X}^*$ - where $\boldsymbol{X}\boldsymbol{X}^* = \boldsymbol{R}_{\boldsymbol{X}}$ is the covariance matrix. Hence, given the definition of the Euclidean norm (L_2-norm), and defining $\boldsymbol{B}_a = 1/2\left[\boldsymbol{I}_d - \boldsymbol{1}\boldsymbol{1}^*/K\right]$, we can now write the following

$$\|\boldsymbol{B}\| = \lambda_{max}\left(\boldsymbol{B}^*\boldsymbol{B}\right) \quad (103)$$

$$= \lambda_{max}\left((\boldsymbol{B}_a \otimes \boldsymbol{R}_{\boldsymbol{X}})(\boldsymbol{B}_a \otimes \boldsymbol{R}_{\boldsymbol{X}})\right) \quad (104)$$

$$= \lambda_{max}\left(\boldsymbol{B}_a\boldsymbol{B}_a \otimes \boldsymbol{R}_{\boldsymbol{X}}\boldsymbol{R}_{\boldsymbol{X}}\right) \quad (105)$$

$$= \lambda_{max}\left(\boldsymbol{U}_{\boldsymbol{B}_a}\|\boldsymbol{\Lambda}_{\boldsymbol{B}_a}\|^2\boldsymbol{U}^*_{\boldsymbol{B}_a} \otimes \boldsymbol{U}_{\boldsymbol{R}_{\boldsymbol{X}}}\|\boldsymbol{\Lambda}_{\boldsymbol{R}_{\boldsymbol{X}}}\|^2\boldsymbol{U}^*_{\boldsymbol{R}_{\boldsymbol{X}}}\right) \quad (106)$$

$$= \lambda_{max}[(\boldsymbol{U}_{\boldsymbol{B}_a} \otimes \boldsymbol{U}_{\boldsymbol{R}_{\boldsymbol{X}}}) \left(\|\boldsymbol{\Lambda}_{\boldsymbol{B}_a}\|^2 \otimes \|\boldsymbol{\Lambda}_{\boldsymbol{R}_{\boldsymbol{X}}}\|^2\right) (\boldsymbol{U}_{\boldsymbol{B}_a} \otimes \boldsymbol{U}_{\boldsymbol{R}_{\boldsymbol{X}}})^*] \quad (107)$$

$$= \lambda_{max}(\|\boldsymbol{\Lambda}_{\boldsymbol{B}_a}\|^2 \otimes \|\boldsymbol{\Lambda}_{\boldsymbol{R}_{\boldsymbol{X}}}\|^2), \quad (108)$$

where we know that $\boldsymbol{B}_a$ and $\boldsymbol{R}_{\boldsymbol{X}}$ are real symmetric matrices, $\boldsymbol{1}$ is a column vector of ones with dimension K, $\otimes$ is the Kronecker product, $\lambda_{max}(.)$ is the maximum eigen-value operator, $\boldsymbol{U}$ and $\boldsymbol{\Lambda}$ are the matrices of the (orthonormal) eigen-vectors and eigen-values respectively, and $\|\boldsymbol{\Lambda}\|^2$ represents the square value of the diagonal entries.

Now, it becomes relevant to obtain the structure of $\|\boldsymbol{\Lambda}_{\boldsymbol{B}_a}\|^2$. It can be easily shown that matrix $\boldsymbol{1}\boldsymbol{1}^*$ has rank 1, and its eigen-values are K (once) and 0 ($K-1$ times). Naturally, it follows that $\boldsymbol{1}\boldsymbol{1}^*/K$ has eigen-values 1 (once) and 0 ($K-1$ times). Further, eigen-values for $[\boldsymbol{I}_d - \boldsymbol{1}\boldsymbol{1}^*/K]$ are clearly 0 (once) and 1 ($K-1$ times). Now, to get the eigen-values of $\|\boldsymbol{\Lambda}_{\boldsymbol{B}_a}\|^2$ we must first multiply them by $1/2$, and then square them; which gets us the eigen-values 0 (once) and $1/4$ ($K-1$ times). The latter being the maximum eigen-value of $\|\boldsymbol{\Lambda}_{\boldsymbol{B}_a}\|^2$.

Therefore, it becomes clear that the result for the $\lambda_{max}(.)$ operator in (108) is in fact given by

$$\frac{1}{4}\lambda_{max}(\|\boldsymbol{\Lambda}_{\boldsymbol{R}_{\boldsymbol{X}}}\|^2) = \frac{1}{4}\lambda_{max}(\boldsymbol{R}^*_{\boldsymbol{X}}\boldsymbol{R}_{\boldsymbol{X}}) = \frac{1}{4}\|\boldsymbol{R}_{\boldsymbol{X}}\|.\Box, \quad (109)$$

the same expression found in (86) for β, if we assume the feature space to be represented using a (Gaussian) RBF kernel, where $\boldsymbol{X}$, $\boldsymbol{X}^*$ becomes $\phi(\boldsymbol{X})$, $\phi(\boldsymbol{X})^*$ respectively.

Appendix C
Additional Results for Man image (1024x1024), $KS = 5$, $Ds = 3$

Fig.7, Fig.8, Fig.9, and Fig.10 are figures of this appendix.

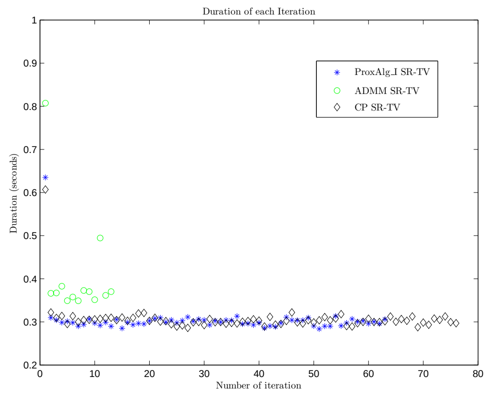


Fig. 7. Duration per iteration execution, in one entire run until solution is obtained. Early iterations register superior values due to initializations. For Man image (1024x1024), $Ds = 3$, $KS = 5$, $\lambda_{TV} = 0.2$. Algorithms are run until 1% objective function $\Phi(.)$ relative error is reached. Each algorithm is presented: *ADMM*, *Proximal Algorithm I* (ProxAlg_I), *Chambolle-Pock* (CP).

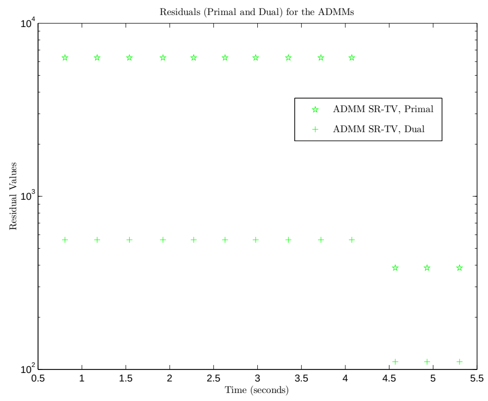


Fig. 8. Residues for ADMM, in one entire run until solution is obtained. Calculated only every 10 iterations to avoid computational burden. μ has the same update window. For Man image (1024x1024), $Ds = 3$, $KS = 5$, $\lambda_{TV} = 0.2$. Algorithms are run until 1% objective function $\Phi(.)$ relative error is reached.

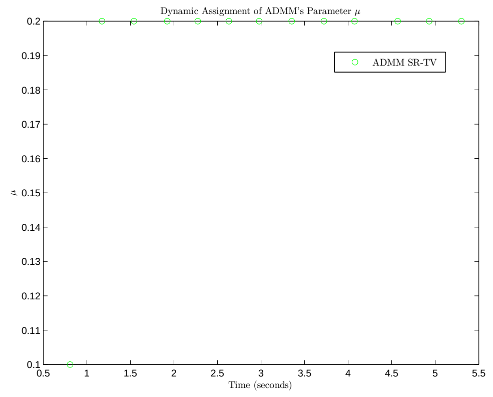
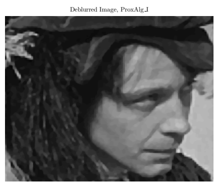


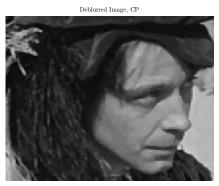
Fig. 9. Dynamic μ assignment variation, in one entire run until solution is obtained. Updated every 10 iterations to avoid computational burden. For Man image (1024x1024), $Ds = 3$, $KS = 5$, $\lambda_{TV} = 0.2$. Algorithms are run until 1% objective function $\Phi(.)$ relative error is reached.

(a) Man (1024x1024) Image; $Ds = 3$; $KS = 5$; $\lambda_{TV} = 0.2$. Man Blurred/ Down-Sampled Image.

(b) Man (1024x1024) Image; $Ds = 3$; $KS = 5$; $\lambda_{TV} = 0.2$. Man Deblurred Image, by ADMM.



(c) Man (1024x1024) Image; $Ds = 3$; $KS = 5$; $\lambda_{TV} = 0.2$. Man Deblurred Image, by ProxAlg_I.



(d) Man (1024x1024) Image; $Ds = 3$; $KS = 5$; $\lambda_{TV} = 0.2$. Man Deblurred Image, by CP.

Fig. 10. First image is the blurred and down-sampled image, with (1024/3 x 1024/3) size, and Gaussian Blur of kernel lateral size of 5 pixels. The others are different deblurred images, for each algorithm. For Man image (1024x1024), $Ds = 3$, $KS = 5$, $\lambda_{TV} = 0.2$. Algorithms are run until 1% objective function $\Phi(.)$ relative error is reached. Zoom into the figure for better comparison.

Appendix D
Comparison between new and old LORSAL

Fig.11 is the figure of this appendix.

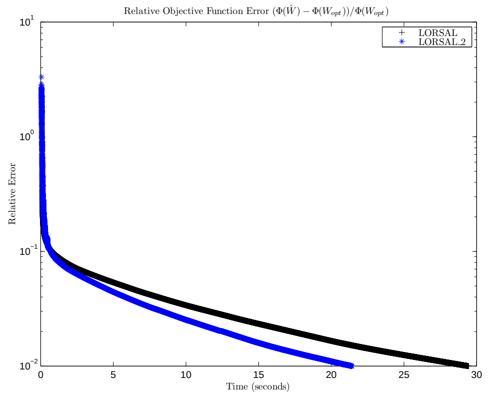


Fig. 11. Relative error of objective function $\Phi(.)$ vs. time, in one entire run until solution is obtained - convergence curve. For USGS_1995_Library.mat data and 3 possible classes, $\lambda_{SMLR} = 0.001$, $\mu = 0.001/10$. Algorithms are run until 1% objective function $\Phi(.)$ relative error is reached. Each algorithm is presented: *ADMM Old Execution* (LORSAL), *ADMM New Execution* (LORSAL_2). In this case, LORSAL_2 is the implementation offered in this report, and uses $\alpha = 1.65$. LORSAL_2 is faster in terms of convergence, which is also a result of using an α relaxation step - defined in section II-A.

Appendix E
Image Segmentation Results: Original, Noisy Data and Classification

Fig.12, Fig.13, Fig.14, and Fig.15 belong to this appendix.

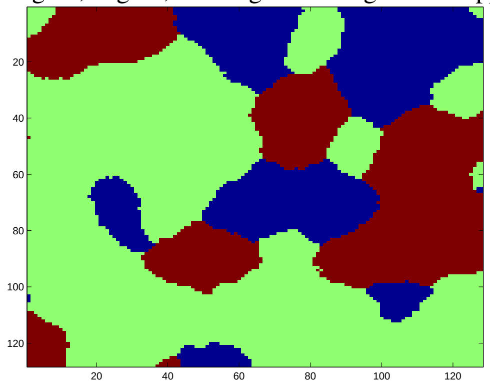


Fig. 12. Original Data (Labels vs Pixels), without noise. For USGS_1995_Library.mat data and 3 possible classes.

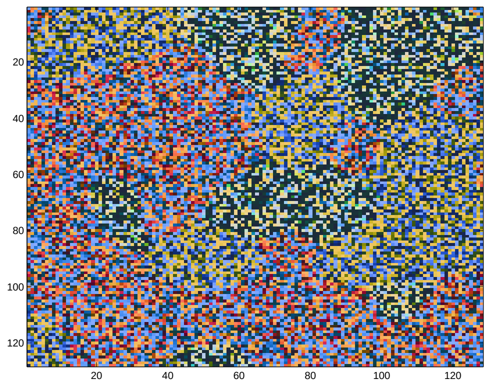


Fig. 13. Noisy Data (Labels vs Pixels), testing data. For USGS_1995_Library.mat data and 3 possible classes.

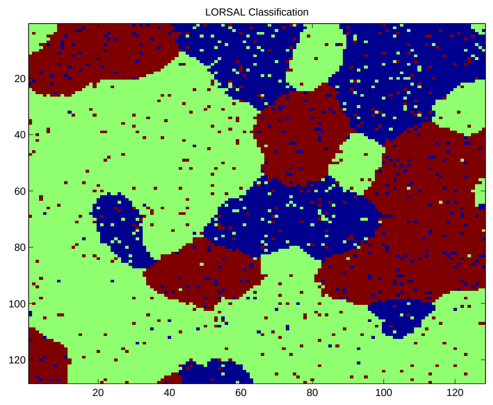


Fig. 14. Segmentation performed (Labels vs Pixels). For USGS_1995_Library.mat data and 3 possible classes, $\lambda_{SMLR} = 0.001$, $\mu = 0.001/10$, $\alpha = 1.65$. Algorithms are run until 1% objective function $\Phi(.)$ relative error is reached. For ADMM (LORSAL), testing training accuracy was 0.9967, testing data accuracy was 0.9485.

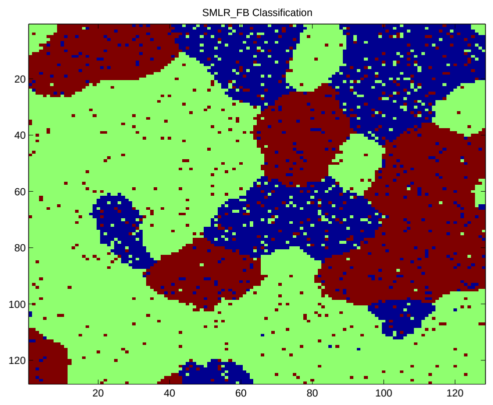


Fig. 15. Segmentation performed (Labels vs Pixels). For USGS_1995_Library.mat data and 3 possible classes, $\lambda_{SMLR} = 0.001$, $\rho = 1$. Algorithms are run until 1% objective function $\Phi(.)$ relative error is reached. For Forward-Backward (SMLR_FB), testing training accuracy was 0.9999, testing data accuracy was 0.9387.

Acknowledgment

The author would like to thank Prof. José Bioucas-Dias (IST - IT) and Prof. Mário Figueiredo (IST - IT), and Instituto de Telecomunicações at IST (Instituto Superior Técnico - University of Lisbon), Lisbon.

References

[1] N. Parikh and S. Boyd, "Proximal algorithms," *Found. Trends Optim.*, vol. 1, no. 3, pp. 127–239, Jan. 2014. [Online]. Available: http://dx.doi.org/10.1561/2400000003

[2] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jan. 2011. [Online]. Available: http://dx.doi.org/10.1561/2200000016

[3] M. Figueiredo and J. Bioucas-Dias, "Restoration of poissonian images using alternating direction optimization," *Image Processing, IEEE Transactions on*, vol. 19, no. 12, pp. 3133–3145, Dec 2010.

[4] T. Goldstein and S. Osher, "The split bregman method for l1-regularized problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 2, pp. 323–343, 2009. [Online]. Available: http://dx.doi.org/10.1137/080725891

[5] D. O'Connor and L. Vandenberghe, "Primal-dual decomposition by operator splitting and applications to image deblurring," *SIAM Journal on Imaging Sciences*, vol. 7, no. 3, pp. 1724–1754, 2014. [Online]. Available: http://dx.doi.org/10.1137/13094671X

[6] A. Chambolle and T. Pock, "A first-order primal-dual algorithm for convex problems with applications to imaging," *Journal of Mathematical Imaging and Vision*, vol. 40, no. 1, pp. 120–145, 2011. [Online]. Available: http://dx.doi.org/10.1007/s10851-010-0251-1

[7] P. Combettes and J.-C. Pesquet, "Proximal splitting methods in signal processing," in *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, ser. Springer Optimization and Its Applications, H. H. Bauschke, R. S. Burachik, P. L. Combettes, V. Elser, D. R. Luke, and H. Wolkowicz, Eds. Springer New York, 2011, vol. 49, pp. 185–212. [Online]. Available: http://dx.doi.org/10.1007/978-1-4419-9569-8_10

[8] L. Condat, "A primal-dual splitting method for convex optimization involving lipschitzian, proximable and linear composite terms," *Journal of Optimization Theory and Applications*, vol. 158, no. 2, pp. 460–479, 2013. [Online]. Available: http://dx.doi.org/10.1007/s10957-012-0245-9

[9] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, 1st ed. Springer Publishing Company, Incorporated, 2011.

[10] T. F. Chan, G. H. Golub, and P. Mulet, "A nonlinear primal-dual method for total variation-based image restoration," *SIAM Journal on Scientific Computing*, vol. 20, no. 6, pp. 1964–1977, 1999. [Online]. Available: http://dx.doi.org/10.1137/S1064827596299767

[11] L. Condat, "A generic proximal algorithm for convex optimization;application to total variation minimization," *Signal Processing Letters, IEEE*, vol. 21, no. 8, pp. 985–989, Aug 2014.

[12] R. H. Chan, M. Tao, and X. Yuan, "Constrained total variation deblurring models and fast algorithms based on alternating direction method of multipliers," *SIAM Journal on Imaging Sciences*, vol. 6, no. 1, pp. 680–697, 2013. [Online]. Available: http://dx.doi.org/10.1137/110860185

[13] D. Gabay and B. Mercier, "A dual algorithm for the solution of nonlinear variational problems via finite element approximation," *Computers & Mathematics with Applications*, vol. 2, no. 1, pp. 17 – 40, 1976. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0898122176900031

[14] M. Almeida and M. Figueiredo, "Deconvolving images with unknown boundaries using the alternating direction method of multipliers," *Image Processing, IEEE Transactions on*, vol. 22, no. 8, pp. 3074–3086, Aug 2013.

[15] M. Afonso, J. Bioucas-Dias, and M. Figueiredo, "Fast image recovery using variable splitting and constrained optimization," *Image Processing, IEEE Transactions on*, vol. 19, no. 9, pp. 2345–2356, Sept 2010.

[16] J. Eckstein and D. Bertsekas, "On the douglas-rachford splitting method and the proximal point algorithm for maximal monotone operators," *Mathematical Programming*, vol. 55, no. 1-3, pp. 293–318, 1992. [Online]. Available: http://dx.doi.org/10.1007/BF01581204

[17] R. Nishihara, L. Lessard, B. Recht, A. Packard, and M. I. Jordan, "A General Analysis of the Convergence of ADMM," *ArXiv e-prints*, Feb. 2015.

[18] W. Deng and W. Yin, "On the global and linear convergence of the generalized alternating direction method of multipliers," *Journal of Scientific Computing*, pp. 1–28, 2015. [Online]. Available: http://dx.doi.org/10.1007/s10915-015-0048-x

[19] T. Goldstein, B. O'Donoghue, S. Setzer, and R. Baraniuk, "Fast alternating direction optimization methods," *SIAM Journal on Imaging Sciences*, vol. 7, no. 3, pp. 1588–1623, 2014. [Online]. Available: http://dx.doi.org/10.1137/120896219

[20] N. Pustelnik, C. Chaux, and J. Pesquet, "Parallel proximal algorithm for image restoration using hybrid regularization," *Image Processing, IEEE Transactions on*, vol. 20, no. 9, pp. 2450–2462, Sept 2011.

[21] P. L. Combettes and J.-C. Pesquet, "A proximal decomposition method for solving convex variational inverse problems," *Inverse Problems*, vol. 24, no. 6, p. 065014, 2008. [Online]. Available: http://stacks.iop.org/0266-5611/24/i=6/a=065014

[22] A. Plaza, J. A. Benediktsson, J. W. Boardman, J. Brazile, L. Bruzzone, G. Camps-Valls, J. Chanussot, M. Fauvel, P. Gamba, A. Gualtieri, M. Marconcini, J. C. Tilton, and G. Trianni, "Recent advances in techniques for hyperspectral image processing," *Remote Sensing of Environment*, vol. 113, Supplement 1, pp. S110 – S122, 2009, imaging Spectroscopy Special Issue. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0034425709000807

[23] M. Chi and L. Bruzzone, "Semisupervised classification of hyperspectral images by svms optimized in the primal," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 45, no. 6, pp. 1870–1880, June 2007.

[24] D. Bhning, "Multinomial logistic regression algorithm," *Annals of the Institute of Statistical Mathematics*, vol. 44, no. 1, pp. 197–200, 1992. [Online]. Available: http://dx.doi.org/10.1007/BF00048682

[25] J. Li, J. Bioucas-Dias, and A. Plaza, "Hyperspectral image segmentation using a new bayesian approach with active learning," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 49, no. 10, pp. 3947–3960, Oct 2011.

[26] B. Krishnapuram, L. Carin, M. Figueiredo, and A. Hartemink, "Sparse multinomial logistic regression: fast algorithms and generalization bounds," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 6, pp. 957–968, June 2005.

[27] J. Borges, A. Marcal, and J. Bioucas-Dias, "Evaluation of bayesian hyperspectral image segmentation with a discriminative class learning," in *Geoscience and Remote Sensing Symposium, 2007. IGARSS 2007. IEEE International*, July 2007, pp. 3810–3813.

[28] J. Borges, J. Bioucas-Dias, and A. Maral, "Fast sparse multinomial regression applied to hyperspectral data," in *Image Analysis and Recognition*, ser. Lecture Notes in Computer Science, A. Campilho and M. Kamel, Eds. Springer Berlin Heidelberg, 2006, vol. 4142, pp. 700–709. [Online]. Available: http://dx.doi.org/10.1007/11867661_63

[29] J. Li, J. Bioucas-Dias, and A. Plaza, "Semisupervised hyperspectral image segmentation using multinomial logistic regression with active learning," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 48, no. 11, pp. 4085–4098, Nov 2010.

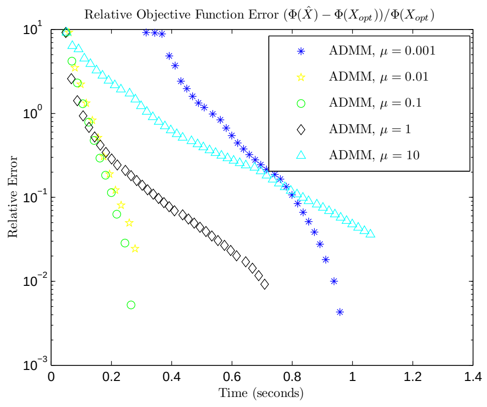
[30] D. R. Hunter and K. Lange, "A tutorial on mm algorithms," *The American Statistician*, vol. 58, no. 1, pp. 30–37, 2004. [Online]. Available: http://dx.doi.org/10.1198/0003130042836

[31] D. Donoho, "De-noising by soft-thresholding," *Information Theory, IEEE Transactions on*, vol. 41, no. 3, pp. 613–627, May 1995.

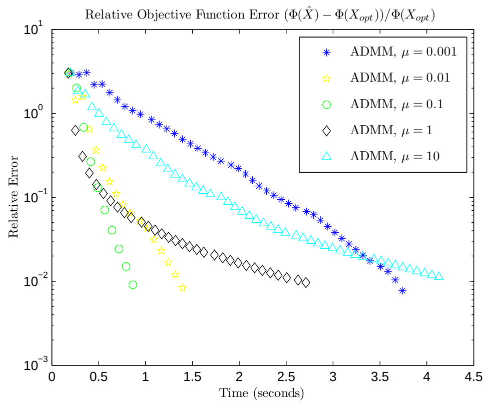
Appendix F
Parameter Selection for the Super-Resolution Total Variation (SR-TV) Problem: ADMM, CP and ProxAlg_I

Here we present algorithm parameter selection for the SR-TV problem.
Values in this section do not necessarily match those in III-F, some due to different conditions, others due to convergence curve fluctuations, which sometimes marginally affect convergence times. However, generally, the results are roughly the same, and corroborate the analysis in III-F about the relation between the different algorithms. Moreover, it is easy to see that much of the parameters are problem specific for each algorithm, and are valid for different deblurring scenarios simultaneously.

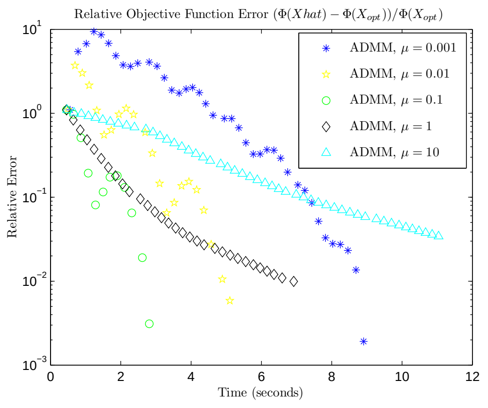
A. ADMM: Selection of initial μ (Step Parameter)



(a) Cameraman (256x256) Image; $Ds = 1$; $KS = 3$. This particular experiment was run with different conditions of section III-F: we used $\lambda_{TV} = 0.005$ and a precision of 1e-4. Though considerations for parameter selection should remain roughly the same.



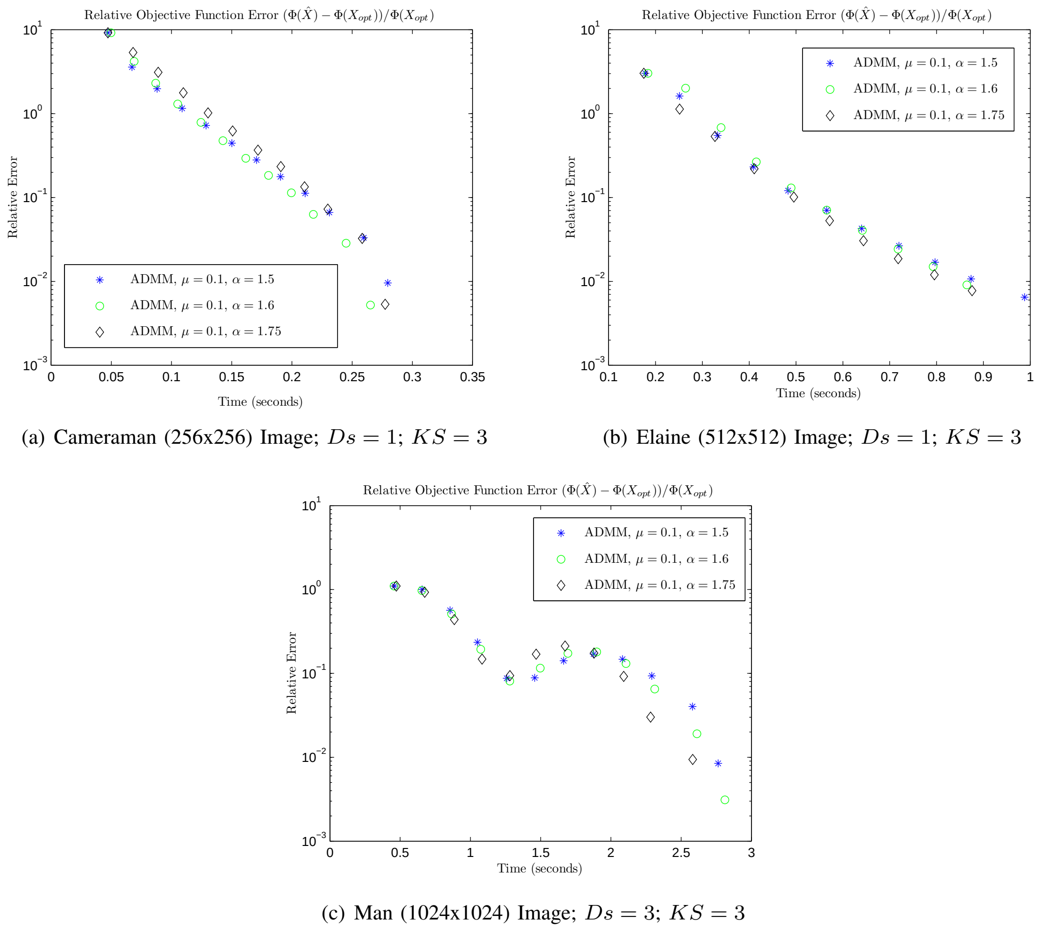
(b) Elaine (512x512) Image; $Ds = 1$; $KS = 3$



(c) Man (1024x1024) Image; $Ds = 3$; $KS = 3$

Fig. 16. Results for initial μ (ADMM step parameter) selection. The best initial μ value is clearly $\mu = 0.1$.

B. ADMM: Selection of α (Relaxation Parameter)



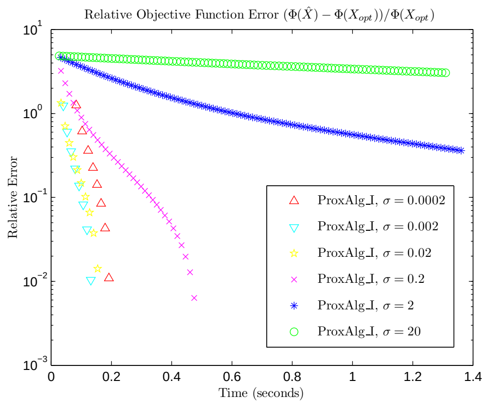
(a) Cameraman (256x256) Image; $Ds = 1$; $KS = 3$

(b) Elaine (512x512) Image; $Ds = 1$; $KS = 3$

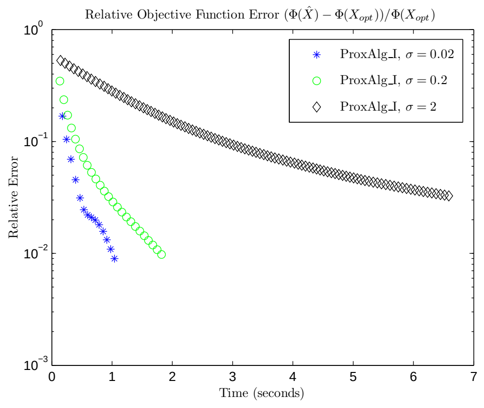
(c) Man (1024x1024) Image; $Ds = 3$; $KS = 3$

Fig. 17. Results for α (relaxation constant) selection. The values are more or less the same in terms of performance, within the interval recommended in II-A for α.

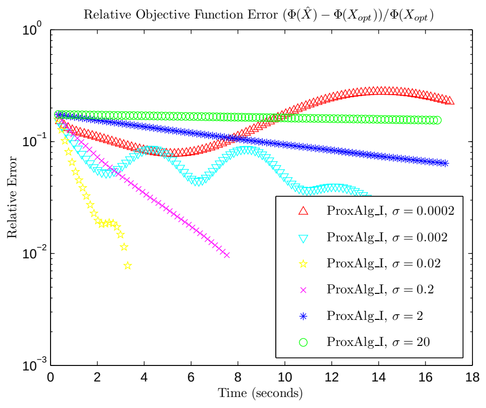
C. ProxAlg_I: Selection of σ (Step Parameter)



(a) Cameraman (256x256) Image; $Ds = 1$; $KS = 3$



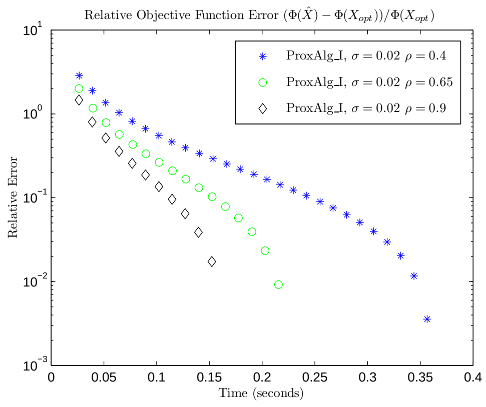
(b) Elaine (512x512) Image; $Ds = 1$; $KS = 3$



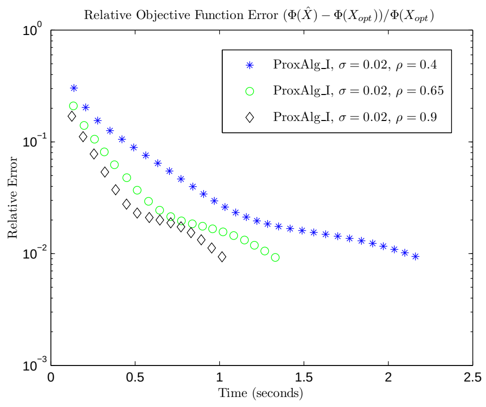
(c) Man (1024x1024) Image; $Ds = 3$; $KS = 3$

Fig. 18. Results for σ (step parameter) selection. The convergence curves clearly indicate that a value near $\sigma = 0.02$ is usually the best option.

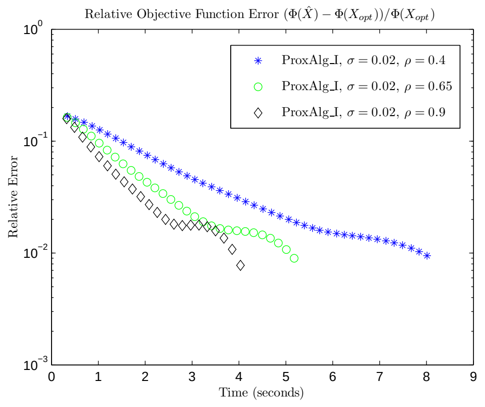
D. *ProxAlg_I: Selection of ρ (Relaxation Parameter)*



(a) Cameraman (256x256) Image; $Ds = 1$; $KS = 3$



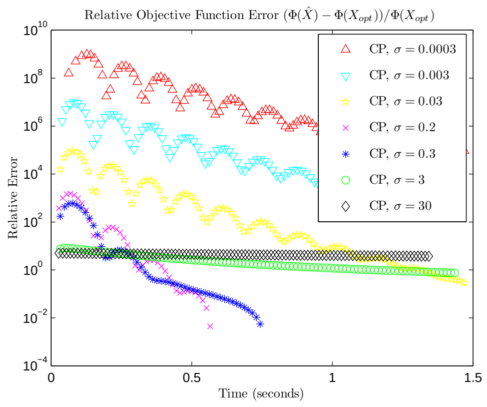
(b) Elaine (512x512) Image; $Ds = 1$; $KS = 3$



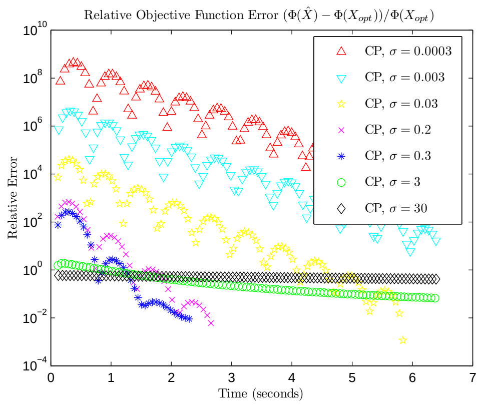
(c) Man (1024x1024) Image; $Ds = 3$; $KS = 3$

Fig. 19. Results for ρ (relaxation parameter) selection. The convergence curves clearly indicate that a value near $\rho = 1$ is the best option, as was stated by the implementation of *Generic Proximal Algorithm* into *ProxAlg_I*, in section III-C.

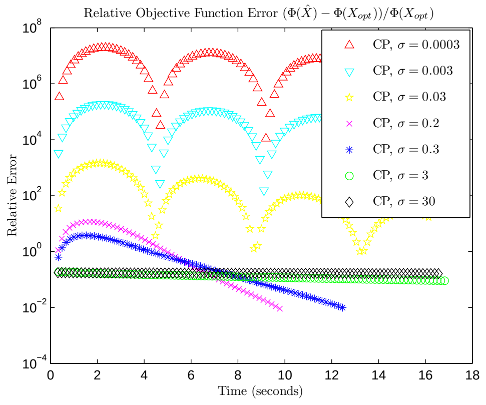
E. *CP: Selection of σ (Step Parameter)*



(a) Cameraman (256x256) Image; $Ds = 1$; $KS = 3$



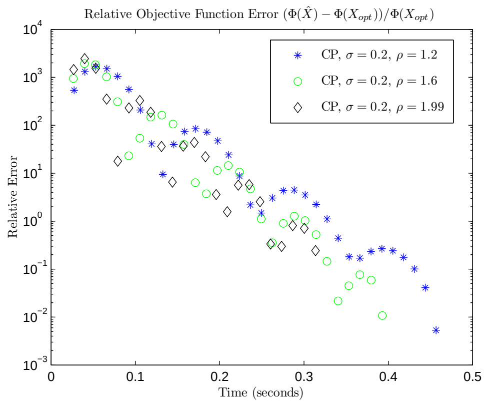
(b) Elaine (512x512) Image; $Ds = 1$; $KS = 3$



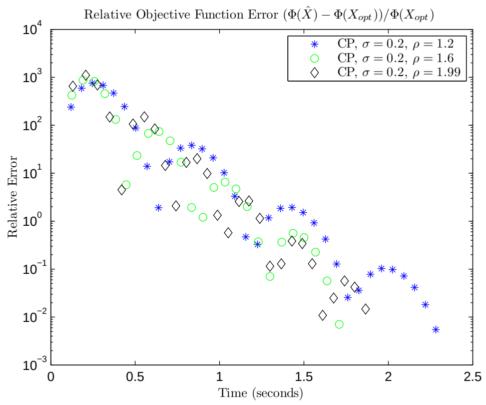
(c) Man (1024x1024) Image; $Ds = 3$; $KS = 3$

Fig. 20. Results for σ (step parameter) selection. The best values for σ in the case of *Chambolle-Pock* (CP) are between $\sigma = 0.2$ and $\sigma = 0.3$. Further, as can be seen, there is some harmonic fluctuation in the curves, which affects convergence times, sometimes benefiting $\sigma = 0.2$ or $\sigma = 0.3$. It is natural to have this harmonic fluctuation, since the is no guarantee of monotonicity in the convergence, according to II-B.

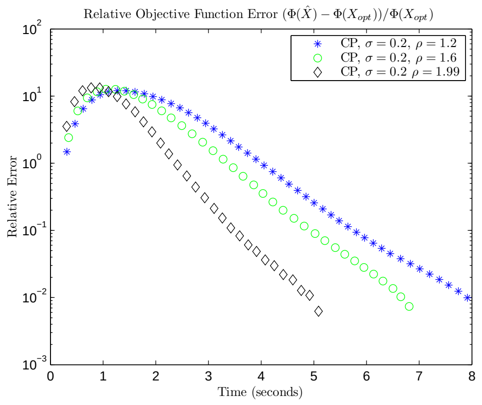
F. CP: Selection of ρ (Relaxation Parameter)



(a) Cameraman (256x256) Image; $Ds = 1$; $KS = 3$



(b) Elaine (512x512) Image; $Ds = 1$; $KS = 3$



(c) Man (1024x1024) Image; $Ds = 3$; $KS = 3$

Fig. 21. Results for ρ (relaxation parameter) selection. The best value for ρ in the case of *Chambolle-Pock* (CP) is usually a value near $\rho = 2$, as can be seen above, and as recommended in the theoretical introduction, in section II-B. Once more we register some harmonic fluctuation, which sometimes marginally affects the results.