Dual Block Coordinate Forward-Backward Algorithm with Application to Deconvolution and Deinterlacing of Video Sequences

Feriel Abboud, Emilie Chouzenoux, Jean-Christophe Pesquet, Jean-Hugues Chenot, Louis Laborelli

Dual Block-Coordinate Forward-Backward Algorithm with Application to Deconvolution and Deinterlacing of Video Sequences

Feriel Abboud[*,†], Emilie Chouzenoux[*,‡], Jean-Christophe Pesquet[‡], Jean-Hugues Chenot[†] and Louis Laborelli[†]

July 29, 2022

Abstract

Optimization methods play a central role in the solution of a wide array of problems encountered in various application fields, such as signal and image processing. Especially when the problems are highly dimensional, proximal methods have shown their efficiency through their capability to deal with composite, possibly non smooth objective functions. The cornerstone of these approaches is the proximity operator, which has become a quite popular tool in optimization. In this work, we propose new dual forward-backward formulations for computing the proximity operator of a sum of convex functions involving linear operators. The proposed algorithms are accelerated thanks to the introduction of a block coordinate strategy combined with a preconditioning technique. Numerical simulations emphasize the good performance of our approach for the problem of jointly deconvoluting and deinterlacing video sequences.

Keywords: Proximity operator, Duality, Block-coordinate approach, Video processing, Deconvolution, Deinterlacing.

1 Introduction

A large number of problems in image processing can be expressed as inverse problems whose solution is defined as the minimizer of a cost function which combines a data fidelity term, that describes the processing leading to the observed data, with some regularization terms accounting for prior information. In image and video restoration problems, optimal solutions are usually reached using models that involve nonsmooth functions, for which proximity operators appear as the most suitable tools [1]. Proximal methods indeed allow to consider the minimization of a sum of functions, the differentiable ones being dealt with through their gradient, whereas the nonsmooth functions are processed by evaluating their proximity operator [2]. Among the existing proximal methods, the class of primal-dual algorithms provides appealing strategies making it possible to split the considered objective function into simpler terms that are handled separately, without requiring any linear operator inversion. This allows the computational complexity to be reduced, especially when the processed data and linear operators are of high

∗ University Paris-Est, LIGM, UMR CNRS 8049, 77454 Marne-la-Vallée, France
† Institut National de l'Audiovisuel. 94366 Bry sur Marne, France
‡ Center for Visual Computing, CentraleSupelec, University Paris-Saclay, Grande Voie des Vignes, 92295 Châtenay-Malabry

dimensions. However, some of these methods encounter several limitations due to slow convergence and high memory requirements issues. These effects are especially critical when one has to deal with huge datasets as in the case of video processing [3, 4].

This paper addresses the problem of computing the proximity operator of a sum of convex possibly nonsmooth functions composed with arbitrary linear operators, in the context of large scale optimization problems. This problem has attracted a large interest and has been widely investigated in the literature via deterministic and stochastic approaches. Among deterministic methods, one can mention the dual parallel algorithm in [5], that converges strongly to the sought proximity operator, in the case of convex proper lower-semicontinuous functions composed with bounded linear operators. Its particular case is the Dykstra-like algorithm [6] when the problem reduces to evaluating the proximity operator of a sum of two convex functions. It is also worth mentioning the work in [7] which proposes a parallel splitting version of the Alternating Direction Method of Multipliers [8].

An appealing idea in the context of optimization is to adopt a block coordinate strategy [9], in such a way that two successive iterations deal with different blocks of variables. The block selection rule can be either deterministic (e.g. cyclic, quasi-cyclic, greedy) [10, 11] or random [12, 13, 14]. Based on this idea, various stochastic algorithms have been initially proposed in machine learning area, usually known as *dual ascent algorithms*. One can mention the stochastic dual coordinate ascent algorithm [15] where the functions are assumed to be Lipschitz continuous or smooth with a Lipschitz gradient, and its variant [16] where the selection rule of the blocks is arbitrary and the smoothness of the objective function is required. Another stochastic algorithm is the communication efficient distributed dual coordinate ascent algorithm [12] which has been designed in order to distribute block processing over multiple cores or remote machines. Nevertheless, the convergence guaranties shown for these dual ascent algorithms only concern decay properties on the dual of the objective function, the variables being assumed to be scalar. In the deterministic case, an accelerated FISTA-like method is proposed in [17], where the authors investigate a similar problem. They provide convergence guaranties for primal iterates as well, when each involved function deals with a variable belonging to $\mathbb{R}^2$.

The main contribution of this paper is the proposal of new primal-dual algorithms, based on the forward-backward iterations similarly to [5, 18], and combined with a block coordinate strategy where preconditioning matrices are introduced. The proposed algorithms can be used for computing the proximity operator of a sum of convex functions involving linear operators. In addition, we show that they benefit from convergence guaranties on both primal and dual sequences for arbitrary linear operators. Finally, the effectiveness of our algorithms in dealing with large-scale optimization problems is demonstrated for the deconvolution and deinterlacing of video sequences.

The paper is structured as follows: Section 2 introduces some optimization tools that will be needed throughout the paper as well as the considered optimization problem. In Section 3, we derive new algorithms by introducing a block-coordinate strategy. Section 4 investigates the convergence properties of our algorithms, and Section 5 addresses their application to deconvolution and super-resolution of interlaced video sequences. Finally, some conclusions are drawn in Section 6.

2 Problem Statement

2.1 Optimization Background

Let us first introduce some definitions and notations that will be used throughout this paper.

Definition 1 *Let ψ be a function from $\mathbb{R}^N$ to $]-\infty,+\infty]$. The domain of ψ is $\mathrm{dom}\,\psi = \{x \in$*

$\mathbb{R}^N : \psi(x) < +\infty\}$. *The function* ψ *is said proper if and only if its domain* dom ψ *is nonempty.*

Definition 2 *Let* E *be a subset of* $\mathbb{R}^N$. *The indicator function* ι_E *of set* E *is given by*

$$\iota_E(x) = \begin{cases} 0 & \text{if } x \in E, \\ +\infty & \text{otherwise.} \end{cases} \tag{1}$$

Definition 3 *Let* $\Gamma_0(\mathbb{R}^N)$ *denote the set of convex proper lower-semicontinuous functions from* $\mathbb{R}^N$ *to* $]-\infty, +\infty]$. *Let* $\psi \in \Gamma_0(\mathbb{R}^N)$ *and* $B \in \mathbb{R}^{N\times N}$ *be a symmetric positive definite matrix. The proximity operator of* ψ *at* $\widetilde{x} \in \mathbb{R}^N$ *relative to the metric induced by* B *is denoted by* $\mathrm{prox}_{B,\psi}(\widetilde{x})$ *and defined as [1]:*

$$\mathrm{prox}_{B,\psi}(\widetilde{x}) \;=\; \underset{x\in\mathbb{R}^N}{\mathrm{argmin}} \;\; \psi(x) + \frac{1}{2}\|x - \widetilde{x}\|_B^2, \tag{2}$$

where the weighted norm $\|\cdot\|_B = \langle \cdot \,|\, B\,\cdot\rangle^{1/2}$ *is used,* $\langle \cdot \,|\, \cdot \rangle$ *being the usual scalar product of* $\mathbb{R}^N$. *When* B *is equal to the identity matrix of* $\mathbb{R}^N$, *one retrieves the classical proximity operator.*

Definition 4 *Let* ψ *and* φ *be functions from* $\mathbb{R}^N$ *to* $]-\infty, +\infty]$. *The infimal convolution of* ψ *and* φ *is*

$$\psi \,\square\, \varphi : \mathbb{R}^N \to [-\infty, +\infty] : x \to \inf_{y\in\mathbb{R}^N} \left(\psi(y) + \varphi(x-y)\right). \tag{3}$$

The Moreau envelope of ψ *of parameter* $\gamma > 0$ *is*

$${}^{\gamma}\psi = \psi \,\square\, \left(\frac{1}{2\gamma}\|\cdot\|^2\right). \tag{4}$$

Definition 5 *The conjugate of a function* ψ *is denoted by* ψ^* *and defined as follows:*

$$\psi^* : \mathbb{R}^N \to [-\infty, +\infty] : x \to \sup_{\nu\in\mathbb{R}^N} \left(\langle \nu | x\rangle - \psi(\nu)\right). \tag{5}$$

Definition 6 *Let* $A \in \mathbb{R}^{N\times N}$ *and* $B \in \mathbb{R}^{N\times N}$ *be symmetric matrices.* $A \succeq B$ *(resp.* $A \succ B$*) if, for every* $x \in \mathbb{R}^N \setminus \{0\}$,

$$x^\top A x \geqslant x^\top B x \qquad (\textit{resp. } x^\top A x > x^\top B x). \tag{6}$$

Definition 7 *Let* $\psi \in \Gamma_0(\mathbb{R}^N)$. *The Moreau subdifferential of* ψ *at* $x \in$ domψ *is defined as*

$$\partial\psi(x) = \left\{t \in \mathbb{R}^N : \; \forall y \in \mathbb{R}^N, \quad \psi(y) - \langle y - x | t\rangle \geqslant \psi(x)\right\}. \tag{7}$$

Definition 8 *A function* ψ *satisfies the Kurdyka-Łojasiewicz inequality if for every* $\xi \in \mathbb{R}$ *and for every bounded subset* $E \in \mathbb{R}^N$, *there exist three constants* $\kappa > 0, \zeta > 0$ *and* $\theta \in [0, 1[$ *such that [19]*

$$\big(\forall t \in \partial\psi(y)\big) \qquad \|t\| \geqslant \kappa |\psi(y) - \xi|^\theta, \tag{8}$$

for every $y \in E$ *such that* $|\psi(y) - \xi| \leqslant \zeta$ *(with the convention* $0^0 = 0$*).*

2.2 Minimization problem

In this paper, similarly to the work in [5], we are interested in computing the proximity operator of a function g at $\widetilde{x} \in \mathbb{R}^N$, where g is defined as

$$\big(\forall x \in \mathbb{R}^N\big) \quad g(x) = f(x) + h(Ax), \tag{9}$$

with $f \in \Gamma_0(\mathbb{R}^N)$, $h \in \Gamma_0(\mathbb{R}^M)$ and $A \in \mathbb{R}^{M\times N}$ is a linear operator. This reads:

$$\begin{aligned} \text{Find } \widehat{x} &= \operatorname{prox}_g(\widetilde{x}), \\ &= \operatorname*{argmin}_{x\in\mathbb{R}^N} \; f(x) + h(Ax) + \frac{1}{2}\|x-\widetilde{x}\|^2. \end{aligned} \tag{10}$$

Let us assume that

Assumption 1 $\operatorname{ri}\big(A(\operatorname{dom} f)\big) \cap \operatorname{ri}(\operatorname{dom} h) \neq \varnothing$,

where $\operatorname{ri}(S)$ denotes the relative interior of a set S. Then, the dual problem of (10) can be expressed as:

$$\text{Find } \widehat{y} = \operatorname*{argmin}_{y\in\mathbb{R}^M} \; \varphi(-A^\top y + \widetilde{x}) + h^*(y), \tag{11}$$

where $\varphi = f^* \,\square\, \frac{1}{2}\,\|\,.\,\|^2$ is the Moreau envelope of parameter 1 of f^* in the standard Euclidean metric. The latter function has a nonexpansive (i.e., 1-Lipschitzian) gradient.
One can apply the preconditioned forward-backward algorithm to the dual problem (11) by performing at each iteration $n \in \mathbb{N}$ a gradient step on the smooth function φ at y_n and a proximal step on the convex function h^*, as described in Algorithm 1:

Algorithm 1 Preconditioned forward-backward algorithm

Initialization:

Let $y_0 \in \mathbb{R}^M$

For every $n \in \mathbb{N}, \gamma_n \in]0, +\infty[$

for $n = 0, 1, \ldots$ **do**

$\widetilde{y}_n = y_n - \gamma_n B^{-1} \nabla\big(\varphi \circ (-A^\top \cdot + \widetilde{x})\big)(y_n)$

$y_{n+1} = \operatorname{prox}_{\gamma_n^{-1}B, h^*}(\widetilde{y}_n)$

end for

where $B \in \mathbb{R}^{M\times M}$ is a symmetric positive definite matrix with $B \succeq AA^\top$ and

$$\forall n \in \mathbb{N}, \quad \gamma_n \in [\epsilon, 2-\epsilon] \quad \text{with } \epsilon \in]0, 1]. \tag{12}$$

Note that

$$\begin{aligned} \nabla\big(\varphi \circ (-A^\top \cdot + \widetilde{x})\big) &= -A\nabla\varphi(-A^\top \cdot + \widetilde{x}) \\ &= -A \operatorname{prox}_f(-A^\top \cdot + \widetilde{x}). \end{aligned} \tag{13}$$

Then, by setting

$$(\forall n \in \mathbb{N}) \qquad x_n = \operatorname{prox}_f(\widetilde{x} - A^\top y_n), \tag{14}$$

and using the Moreau decomposition

$$\operatorname{prox}_{B, h_j^*} = \mathrm{Id} - B^{-1}\operatorname{prox}_{B^{-1}, h_j}(B\,\cdot), \tag{15}$$

the gradient step of Algorithm 1 can be formulated by means of x_n using (13) and (14), whereas, the proximity operator of h^* can be rewritten in terms of h thanks to Moreau decomposition formula (15). This leads to the following algorithm:

Algorithm 2 Dual forward-backward algorithm

Initialization:

Let $y_0 \in \mathbb{R}^M$

For every $n \in \mathbb{N}, \gamma_n \in]0, +\infty[$

for $n = 0, 1, \ldots$ **do**

$x_n = \mathrm{prox}_f(\widetilde{x} - A^\top y_n)$

$\widetilde{y}_n = y_n + \gamma_n B^{-1} A x_n$

$y_{n+1} = \widetilde{y}_n - \gamma_n B^{-1} \mathrm{prox}_{\gamma_n B^{-1}, h}(\gamma_n^{-1} B \widetilde{y}_n)$

end for

It can be shown that the sequences $(x_n)_{n\in\mathbb{N}}$ and $(y_n)_{n\in\mathbb{N}}$ generated by Algorithm 2 converge to the solutions to the primal and dual problems $\widehat{x}$ and $\widehat{y}$ respectively [18]. Moreover the following relation is satisfied

$$\widehat{x} = \mathrm{prox}_f(\widetilde{x} - A^\top \widehat{y}). \tag{16}$$

Remark 1 *Let us introduce the variables*

$$(\forall n \in \mathbb{N}) \quad p_n = \mathrm{prox}_{\gamma_n B^{-1}, h}(\gamma_n^{-1} B \widetilde{y}_n), \tag{17}$$

$$\begin{aligned} q_n &= -A^\top y_{n+1} + \widetilde{x} - \gamma_n A^\top B^{-1} p_n, \\ &= \widetilde{x} - A^\top(\gamma_n B^{-1} A x_n + y_n), \end{aligned} \tag{18}$$

and let us notice that, if $\gamma_n \equiv \gamma$ satisfies (12) , then the following recursive relation is fulfilled:

$$q_{n+1} = q_n + \gamma A^\top B^{-1}(p_n - A x_{n+1}). \tag{19}$$

Algorithm 2 can then be rewritten as

Algorithm 3 Dykstra-like formulation of Algorithm 2

Initialization:

Let $y_0 \in \mathbb{R}^M, x_0 = \mathrm{prox}_f(\widetilde{x} - A^\top y_0)$ and $q_0 = \widetilde{x} - A^\top(\gamma B^{-1} A x_0 + y_0)$

For every $n \in \mathbb{N}, \gamma \in]0, +\infty[$

for $n = 0, 1, \ldots$ **do**

$p_n = \mathrm{prox}_{\gamma B^{-1}, h}(\gamma^{-1} B y_n + A x_n)$

$y_{n+1} = y_n + \gamma B^{-1}(A x_n - p_n)$

$x_{n+1} = \mathrm{prox}_f(q_n + \gamma A^\top B^{-1} p_n)$

$q_{n+1} = q_n + \gamma A^\top B^{-1}(p_n - A x_{n+1})$

end for

In particular, as pointed out in [18], if $A = I_N$ and $\gamma B^{-1} = I_M$ we retrieve the same iterative structure as the Dykstra-like algorithm which was proposed in [6] and whose convergence was proved for another initialization strategy.

Note that Algorithm 2 does not exploit the potential separability of the function h, thereby, one has to deal with the full linear operator A at each iteration, which may be very costly when the size of A is large.

3 Proposed optimization method

Now, we will derive new algorithms based on Algorithm 2 when h in (9) is a separable function:

$$(\forall x \in \mathbb{R}^N) \qquad h(Ax) = \sum_{j=1}^{J} h_j(A_j x), \tag{20}$$

where, for every $j \in \{1, \ldots, J\}$, A_j is a non null matrix in $\mathbb{R}^{M_j \times N}$ with $\sum_{j=1}^{J} M_j = M$, $h_j \in \Gamma_0(\mathbb{R}^{M_j})$, and

$$A = \begin{bmatrix} A_1 \\ \vdots \\ A_J \end{bmatrix}. \tag{21}$$

Then, problem 10 becomes:

$$\begin{aligned} \text{Find } \widehat{x} &= \operatorname{prox}_g(\widetilde{x}), \\ &= \operatorname*{argmin}_{x \in \mathbb{R}^N} f(x) + \sum_{j=1}^{J} h_j(A_j x) + \frac{1}{2}\|x - \widetilde{x}\|^2. \end{aligned} \tag{22}$$

According to (21), the dual problem reads:

$$\text{Find } \widehat{y} = \operatorname*{argmin}_{y=(y^j)_{1 \leqslant j \leqslant J} \in \mathbb{R}^M} \varphi\Big(\widetilde{x} - \sum_{j=1}^{J} A_j^\top y^j\Big) + \sum_{j=1}^{J} h_j^*(y^j). \tag{23}$$

Note that the dual variable y is now decomposed into J blocks of variables $(y^j)_{1 \leqslant j \leqslant J}$. The application of the variable metric block-coordinate forward-backward algorithm in [9] to the dual problem (23) yields the new Algorithm 4 where, at each iteration $n \in \mathbb{N}$, a block of index j_n is activated and its associated dual variable $y_n^{j_n}$ is updated by performing a proximal step on the function h_{j_n}, in the metric induced by a preconditioning matrix B_{j_n} satisfying (24). Note that the dual variable $y_n^{j_n}$ is the only one to be processed at the n-th iteration, whereas the other dual variables of index $j \neq j_n$ are kept intact during this iteration.

Algorithm 4 Dual block forward-backward algorithm

Initialization:

Let $(y_0^j)_{1\leqslant j\leqslant J} \in \mathbb{R}^M$

For every $n \in \mathbb{N}, \gamma_n \in]0, +\infty[$

for $n = 0, 1, \ldots$ **do**

$x_n = \mathrm{prox}_f(\widetilde{x} - A^\top y_n)$

$j_n \in \{1, \ldots, J\}$

$\widetilde{y}_n^{j_n} = y_n^{j_n} + \gamma_n B_{j_n}^{-1} A_{j_n} x_n$

$y_{n+1}^{j_n} = \widetilde{y}_n^{j_n} - \gamma_n B_{j_n}^{-1} \mathrm{prox}_{\gamma_n B_{j_n}^{-1}, h_{j_n}}\left(\gamma_n^{-1} B_{j_n} \widetilde{y}_n^{j_n}\right)$

$y_{n+1}^j = y_n^j, \quad j \in \{1, \ldots, J\} \setminus \{j_n\}$

end for

where γ_n fulfils (12), and

$$(\forall j \in \{1, \ldots, J\}) \quad B_j \succ O_{M_j} \text{ with } B_j \succeq A_j A_j^\top. \tag{24}$$

The simplest (non-preconditioned) version of Algorithm 4 is obtained by choosing

$$(\forall j \in \{1, \ldots, J\}) \qquad B_j = \beta_j I_{M_j}, \tag{25}$$

where β_j is the squared norm of the associated linear operator A_j, i.e., $\beta_j = \|A_j\|^2$.

Remark 2

1. *The pair* $\left(\widehat{x}, (\widehat{y}^j)_{1\leqslant j\leqslant J}\right)$ *is a solution to the primal and dual problems if and only if*

$$\begin{cases} -\sum_{j=1}^J A_j^\top \widehat{y}^j \in \partial f(\widehat{x}) + \widehat{x} - \widetilde{x} \\ \qquad\qquad \Leftrightarrow \ \widehat{x} = \mathrm{prox}_f\Big(\widetilde{x} - \sum_{j=1}^J A_j^\top \widehat{y}^j\Big), & (26) \\ (\forall j \in \{1, \ldots, J\}) \qquad \widehat{y}^j \in \partial h_j(A_j \widehat{x}). & (27) \end{cases}$$

   When the functions $(h_j)_{1\leqslant j\leqslant J}$ *are differentiable, the second optimality condition can be used to define a dual residue, which is exploited for the blocks selection rule in some recent dual coordinate ascent strategies [20].*

2. *If relation* (25) *is satisfied,* $f = \theta\|\cdot\|_1$ *with* $\theta \in]0, +\infty[$*, and* $(\forall j \in \{1, \ldots, J\})$ $h_j = \iota_{\{b^j\}}$ *with* $b^j \in \mathbb{R}^{M_j}$*, we recover an algorithm similar to the one studied in [21].*

3.1 Simplified form of preconditioned dual block forward-backward

In Algorithm 4, the update of the primal variable x_n involves all the dual variables $(y_n^{j_n})_{1\leqslant j\leqslant J}$ and the whole matrix $A^\top$, whereas only one block j_n is being processed. To overcome this limitation, we introduce a new variable $(z_n)_{n\in\mathbb{N}}$ that takes into account only the updated dual

variable.
To do so, let us define $(z_n)_{n\in\mathbb{N}}$ such that

$$z_n = -A^\top y_n = -\sum_{i=1}^{J} A_i^\top y_n^i. \tag{28}$$

Then we have

$$\begin{aligned} z_{n+1} &= -\sum_{i=1}^{J} A_i^\top y_{n+1}^i = -\sum_{\substack{i=1\\ i\neq j_n}}^{J} A_i^\top y_{n+1}^i - A_{j_n}^\top y_{n+1}^{j_n}, \\ &= -\sum_{i=1}^{J} A_i^\top y_n^i - A_{j_n}^\top y_{n+1}^{j_n} + A_{j_n}^\top y_n^{j_n}, \\ &= z_n - A_{j_n}^\top (y_{n+1}^{j_n} - y_n^{j_n}). \end{aligned} \tag{29}$$

Hence, Algorithm 4 becomes:

Algorithm 5 Simplified dual block forward-backward algorithm

Initialization:

Let $(y_0^j)_{1\leqslant j\leqslant J} \in \mathbb{R}^M$

For every $n \in \mathbb{N}, \gamma_n \in]0, +\infty[$

for $n = 0, 1, \ldots$ **do**

$x_n = \mathrm{prox}_f(\widetilde{x} + z_n)$

$j_n \in \{1, \ldots, J\}$

$\widetilde{y}_n^{j_n} = y_n^{j_n} + \gamma_n B_{j_n}^{-1} A_{j_n} x_n$

$y_{n+1}^{j_n} = \widetilde{y}_n^{j_n} - \gamma_n B_{j_n}^{-1} \mathrm{prox}_{\gamma_n B_{j_n}^{-1}, h_{j_n}} \left(\gamma_n^{-1} B_{j_n} \widetilde{y}_n^{j_n}\right)$

$y_{n+1}^{j} = y_n^{j}, \quad j \in \{1, \ldots, J\} \setminus \{j_n\}$

$z_{n+1} = z_n - A_{j_n}^\top (y_{n+1}^{j_n} - y_n^{j_n})$

end for

with the initialization

$$z_0 = -\sum_{j=1}^{J} A_j^\top y_0^j. \tag{30}$$

This simplified form of Algorithm 4 is more efficient in the sense that the updating step involves only the selected block j_n, thereby, this version reduces the complexity and memory requirements.

3.2 Particular case when $f = 0$

A special interesting case is obtained when f is the null function. Then, the update of the primal variable in Algorithm 5 reduces to

$$x_n = \widetilde{x} + z_n. \tag{31}$$

Thus, by changing the initialization (30) to $x_0 = \widetilde{x} - \sum_{j=1}^{J} A_j^\top y_0^j$ and after some simplifications, Algorithm 5 becomes:

Algorithm 6 Dual block forward-backward algorithm when $f = 0$

Initialization:

Let $(y_0^j)_{1\leqslant j\leqslant J} \in \mathbb{R}^M$

For every $n \in \mathbb{N}, \gamma_n \in]0, +\infty[$

for $n = 0, 1, \ldots$ **do**

$j_n \in \{1, \ldots, J\}$

$\widetilde{y}_n^{j_n} = y_n^{j_n} + \gamma_n B_{j_n}^{-1} A_{j_n} x_n$

$y_{n+1}^{j_n} = \widetilde{y}_n^{j_n} - \gamma_n B_{j_n}^{-1} \mathrm{prox}_{\gamma_n B_{j_n}^{-1}, h_{j_n}} \big(\gamma_n^{-1} B_{j_n} \widetilde{y}_n^{j_n}\big)$

$y_{n+1}^{j} = y_n^{j}, \quad j \in \{1, \ldots, J\} \setminus \{j_n\}$

$x_{n+1} = x_n - A_{j_n}^\top (y_{n+1}^{j_n} - y_n^{j_n})$

end for

3.3 Parallel dual block forward-backward

Algorithm 5 can be compared with its parallel variant proposed in [22, Example 5.6] given by:

Algorithm 7 Parallel dual block forward-backward algorithm [22]

Initialization:

Let $(y_0^j)_{1\leqslant j\leqslant J} \in \mathbb{R}^M$

For every $n \in \mathbb{N}, \gamma_n \in]0, +\infty[$

for $n = 0, 1, \ldots$ **do**

$x_n = \mathrm{prox}_f(\widetilde{x} + z_n)$

for $j = 1, \ldots, J$ **do**

$\widetilde{y}_n^{j} = y_n^{j} + \gamma_n B_{j}^{-1} A_{j} x_n$

$y_{n+1}^{j} = \widetilde{y}_n^{j} - \gamma_n B_{j}^{-1} \mathrm{prox}_{\gamma_n B_{j}^{-1}, h_{j}} \big(\gamma_n^{-1} B_{j} \widetilde{y}_n^{j}\big)$

end for

$z_{n+1} = z_n - \sum_{j=1}^J A_j^\top (y_{n+1}^j - y_n^j)$

end for

where z_0 is defined according to (30) and the preconditioning matrices $(B_j)_{1\leqslant j\leqslant J}$ are such that

$$\forall j \in \{1, \ldots, J\} \quad B_j \succeq \beta I_{M_j}, \tag{32}$$

with $\beta = \sum_{j=1}^{J} \|A_j\|^2$.

Some similarities existing between Algorithms 5 and 7 can be observed. However, in Algorithm 7, the dual variables $(y_n^j)_{1\leqslant j\leqslant J}$ are updated in parallel and the update of x_n has to be performed from all these dual variables. Conversely, in Algorithm 5, the dual variables are updated sequentially, and after any update of each of them, the primal variable is also updated. When no parallel implementation is used, this second solution can be expected to be more efficient.

In addition, conditions (32) imposed on the matrices $(B_j)_{1\leqslant j\leqslant J}$ in Algorithm 7 appear to be

more restrictive than those imposed in Algorithm 5 (see conditions (24)). Since the preconditioning matrices $(B_j)_{1\leqslant j\leqslant J}$ usually play an important role in the convergence speed, more freedom in their choice should also be beneficial to the algorithm performance.

A variant of the above parallel algorithm dealing with the case when $f = 0$ can be derived from the parallel block forward-backward algorithm proposed in [5] which, in the absence of error terms and relaxation factor, reads:

Algorithm 8 Parallel dual block forward-backward algorithm when $f = 0$ [5]

Initialization:

Let $(y_0^j)_{1\leqslant j\leqslant J} \in \mathbb{R}^M$

For every $n \in \mathbb{N}, \gamma_n \in]0, +\infty[$

for $n = 0, 1, \ldots$ **do**

  for $j = 1, \ldots, J$ **do**

    $\widetilde{y}_n^j = y_n^j + \gamma_n B_j^{-1} A_j x_n$

    $y_{n+1}^j = \widetilde{y}_n^j - \gamma_n B_j^{-1} \mathrm{prox}_{\gamma_n B_j^{-1}, h_j}\big(\gamma_n^{-1} B_j \widetilde{y}_n^j\big)$

  end for

  $x_{n+1} = x_n - \sum_{j=1}^{J} A_j^\top (y_{n+1}^j - y_n^j)$

end for

where

$$\forall j \in \{1, \ldots, J\}\ B_j = \beta \omega_j^{-1} I_{M_j},\ \text{with}\ \beta = \max_{j\in\{1,\ldots,J\}} \|A_j\|^2,$$

$$\text{and}\ (\omega_j)_{1\leqslant j\leqslant J} \in]0,1]^J\ \text{are such that}\ \sum_{j=1}^{J} \omega_j = 1.$$

Algorithms 6 and 8 exhibit several similarities, however, as mentioned hereabove, the main difference lies in the update rule of the dual variables. Another advantage of Algorithm 6 is that it leads to less restrictive conditions on the matrices $(B_j)_{1\leqslant j\leqslant J}$. Indeed, for Algorithm 8, we have

$$(\forall j \in \{1, \ldots, J\}) \quad B_j \succeq \omega_j B_j = \beta I_{M_j} \succeq \|A_j\|^2 I_{M_j} \succeq A_j A_j^\top.$$

3.4 Proximity operator in a general metric

In practice, one may be interested in more general problems of the form [23]:

$$\text{Find}\ \widehat{x} = \operatorname*{argmin}_{x\in\mathbb{R}^N}\ f(x) + \sum_{j=1}^{J} h_j(A_j x) + \frac{1}{2}\|x - \widetilde{x}\|_C^2. \tag{33}$$

where $C \in \mathbb{R}^{N\times N}$ is a symmetric strictly positive definite matrix. Algorithms can be deduced from Algorithms 5 and 6 by simply replacing the Euclidean metric of $\mathbb{R}^N$ by the metric induced by C (while keeping the standard Euclidean metric for the spaces $\mathbb{R}^{M_j}$ with $j \in \{1, \ldots, J\}$). By noticing that in the new metric, the adjoints of operators $(A_j)_{1\leqslant j\leqslant J}$ are replaced by $(C^{-1}A_j^\top)_{1\leqslant j\leqslant J}$, Algorithm 5 yields:

Algorithm 9 Dual block forward-backward algorithm in a general metric

Initialization:

Let $(y_0^j)_{1\leqslant j\leqslant J} \in \mathbb{R}^M$

For every $n \in \mathbb{N}, \gamma_n \in]0, +\infty[$

for $n = 0, 1, \ldots$ **do**

$x_n = \mathrm{prox}_{C,f}(\widetilde{x} + z_n)$

$j_n \in \{1, \ldots, J\}$

$\widetilde{y}_n^{j_n} = y_n^{j_n} + \gamma_n B_{j_n}^{-1} A_{j_n} x_n$

$y_{n+1}^{j_n} = \widetilde{y}_n^{j_n} - \gamma_n B_{j_n}^{-1} \mathrm{prox}_{\gamma_n B_{j_n}^{-1}, h_{j_n}}\big(\gamma_n^{-1} B_{j_n} \widetilde{y}_n^{j_n}\big)$

$y_{n+1}^{j} = y_n^{j}, \quad j \in \{1, \ldots, J\} \setminus \{j_n\}$

$z_{n+1} = z_n - C^{-1} A_{j_n}^\top (y_{n+1}^{j_n} - y_n^{j_n})$

end for

where

$$z_0 = -C^{-1} \sum_{j=1}^{J} A_j^\top y_0^j \quad \text{and } \forall j \in \{1, \ldots, J\} \quad B_j \succeq A_j C^{-1} A_j^\top.$$

Similarly, a new algorithm can be derived from Algorithm 6 for computing the sought proximity operator in the metric induced by the matrix C when $f = 0$. This is achieved by simply substituting the adjoints operators of $(A_j)_{1\leqslant j\leqslant J}$ with $(C^{-1}A_j^\top)_{1\leqslant j\leqslant J}$.

4 Convergence analysis

We will need some additional assumptions in order to establish the convergence of the preconditioned dual block forward-backward algorithm 5:

Assumption 2

1. *For every $j \in \{1, \ldots, J\}$, the restriction of h_j^* on its domain is continuous.*
2. *The sequence $(j_n)_{n\in\mathbb{N}}$ follows a quasi-cyclic rule, i.e., there exists $K \geqslant J$ such that, for every $n \in \mathbb{N}$, $\{1, \ldots, J\} \subset \{j_n, \ldots, j_{n+K-1}\}$.*
3. *The functions f and $(h_j)_{1\leqslant j\leqslant J}$ are semi-algebraic.*

The following result can then be established:

Proposition 1 *Suppose that Assumptions 1 and 2 hold. Let $(x_n)_{n\in\mathbb{N}}$ and $\big(y_n = (y_n^j)_{1\leqslant j\leqslant J}\big)_{n\geqslant 1}$ be sequences generated by Algorithm 5. If $(y_n)_{n\geqslant 1}$ is bounded, then $(x_n)_{n\in\mathbb{N}}$ converges to the solution to the primal problem (22) and $(y_n)_{n\geqslant 1}$ converges to a solution to the dual one (23).*

Proof. We have seen that our algorithm amounts to applying a block-coordinate forward-backward approach to the function:

$$\Phi\colon (y^j)_{1\leqslant j\leqslant J} \mapsto \varphi\Big(-\sum_{j=1}^{J} A_j^\top y^j + \widetilde{x}\Big) + \sum_{j=1}^{J} h_j^*(y^j). \tag{34}$$

Since $\|\cdot\|^2$ is a semi-algebraic function and semi-algebraicity is preserved under standard operations such as sum, infimum, conjugate, and inf-convolution, it can be deduced from Assumption

2.3 that Φ is semi-algebraic. It follows from [9, Theorem 3.1] that the sequence $\left(y_n\right)_{n\geqslant 1}$ generated by Algorithm 5 converges to a critical point $\widehat{y}$ of Φ. Since Φ is a convex function, such a critical point is a (global) minimizer of Φ. By using now (14) and the continuity of the proximity operator, it follows that the sequence $(x_n)_{n\in\mathbb{N}}$ converges to a solution $\widehat{x}$ satisfying (16). As already mentioned, $\widehat{x}$ is then the solution to (22). □

Remark 3

1. *The boundedness of sequence $(y_n)_{n\geqslant 1}$ is satisfied if Φ is a coercive function. This happens, in particular, if all the functions $(h_j^*)_{1\leqslant j\leqslant J}$ are coercive, that is when, for every $j \in \{1,\ldots,J\}$, $0 \in \operatorname{int}(\operatorname{dom} h_j)$ [24, Proposition 14.16].*

2. *The quasi-cyclic rule (also sometimes called essentially cyclic rule) provides much more flexibility than the cyclic one. In particular, some of the (blocks of) variables may be activated more frequently than others, and the order in which the variables are swept can be randomly chosen.*

Some more accurate convergence rate results can also be provided. In particular, we give below conditions for which the linear convergence of the proposed algorithm is secured.

Proposition 2 *Suppose that Assumptions 1 and 2 hold and that $\widehat{x}$ and $\widehat{y}$ are the limits of the sequences $(x_n)_{n\in\mathbb{N}}$ and $\left(y_n = (y_n^j)_{1\leqslant j\leqslant J}\right)_{n\geqslant 1}$, respectively. assuming that $(y_n)_{n\in\mathbb{N}}$ is bounded, there exist $\alpha \in]0,+\infty[$ and $\lambda \in]0,+\infty[$ such that, for every $n \geqslant 1$,*

$$\|x_n - \widehat{x}\| \leqslant \lambda \|A\| n^{-\alpha} \tag{35}$$
$$\|y_n - \widehat{y}\| \leqslant \lambda n^{-\alpha}. \tag{36}$$

In addition, if one of the following conditions is met:

1. *Φ, as defined by (34), is strongly convex,*

2. *f is Lipschitz differentiable and A is surjective[1],*

3. *for every $j \in \{1,\ldots,J\}$, h_j is Lipschitz differentiable,*

4. *Φ is a piecewise polynomial function of degree 2,*

5. *f is a quadratic function and, for every $j \in \{1,\ldots,J\}$, h_j^* is a piecewise polynomial function of degree 2,*

then, there exist $\tau \in [0,1[$ and $\lambda' \in]0,+\infty[$ such that, for every $n \geqslant 1$,

$$\|x_n - \widehat{x}\| \leqslant \lambda' \|A\| \tau^n \tag{37}$$
$$\|y_n - \widehat{y}\| \leqslant \lambda' \tau^n. \tag{38}$$

Proof. As shown by [9, Theorem 3.2], the convergence rate of the Dual Forward-Backward algorithm depends on the Łojasiewicz exponent of function Φ defined by (34) at $\widehat{y}$. Then, (36) corresponds to the worst case upper bound. It then follows from (14), (16), and the nonexpansiveness of the proximity operator [24] that, for every $n \geqslant 1$,

$$\begin{aligned}\|x_n - \widehat{x}\| &= \|\operatorname{prox}_f(\widetilde{x} - A^\top y_n) - \operatorname{prox}_f(\widetilde{x} - A^\top \widehat{y})\| \\ &\leqslant \|A^\top (y_n - \widehat{y})\| \\ &\leqslant \|A\| \|y_n - \widehat{y}\|, \end{aligned} \tag{39}$$

[1] It is sometimes said that A is full row rank.

which yields (35).

If Φ is a strongly convex function [25] or Φ is a piecewise polynomial function of degree 2 [25], the Łojasiewicz exponent of function Φ is equal to 1/2. It then follows from [9, Theorem 3.2] that (38) holds. The decay behavior of $(x_n)_{n\geqslant 1}$ in (37) is then deduced as previously.
If f is Lipschitz differentiable, then $f + \frac{1}{2}\|\cdot\|^2$ is also Lipschitz differentiable, and its conjugate φ is thus strongly convex [24]. Since A is surjective,

$$(y^j)_{1\leqslant j\leqslant J} \mapsto \varphi\Big(-\sum_{j=1}^{J} A_j^\top y^j + \widetilde{x}\Big)$$

is strongly convex. The strong convexity of Φ is then guaranteed.
Similarly, if Condition 3 holds, then, for every $j \in \{1,\ldots,J\}$, h_j^* is strongly convex, hence Φ.
Finally, if Condition 5 holds, $f + \frac{1}{2}\|\cdot\|^2$ is a quadratic function and so is its conjugate φ. Since functions $(h_j^*)_{1\leqslant j\leqslant J}$ are assumed to be piecewise polynomial functions of degree 2, Φ is a piecewise polynomial function of degree 2. □

5 Application to video restoration

5.1 Observation Model

In this section, we consider the problem of jointly deblurring and deinterlacing video sequences. Interlacing scan has been the main format for TV recording, broadcasting, and displaying [26], where each frame is formed by merging two successive fields resulting from even (resp. odd) horizontal lines of the first (resp. second) image. However, with the increased popularity of HD flat LCD and plasma screens, that benefit from high brightness and contrast, the human visual system becomes more sensitive to interlacing artefacts [27]. Hence, the need for high image quality has become essential to meet the actual customer's demand [28].
The degradation model is expressed as

$$(\forall t \in \{1,\ldots,T\}) \quad y_t = \mathrm{S}_t\,(h * \overline{x}_t) + w_t, \tag{40}$$

where $(y_t)_{1\leqslant t\leqslant T} \in \mathbb{R}^{TL}$ denotes the interlaced frame sequence, $(\overline{x}_t)_{1\leqslant t\leqslant T} \in \mathbb{R}^{TN}$ is the sought progressive video sequence with T the number of time frames, and L (resp. N) the number of pixels in each image of the interlaced (resp. progressive) sequence. The operator S_t is a row decimation operator where $\mathrm{S}_t = \mathrm{S}_\mathrm{o}$ for odd frames and $\mathrm{S}_t = \mathrm{S}_\mathrm{e}$ for even frames. $h \in \mathbb{R}^P$ corresponds to a convolution kernel accounting for spatial blur, and $(w_t)_{1\leqslant t\leqslant T} \in \mathbb{R}^{TL}$ is an unknown additive noise.
Note that in deinterlacing problems, the number of rows in the progressive video sequence $(\overline{x}_t)_{1\leqslant t\leqslant T}$ is equal to twice that of the fields in the interlaced video sequence $(y_t)_{1\leqslant t\leqslant T}$, thereby we have $N = 2L$.

5.2 Optimization problem

An estimate of the original sequence can be obtained by finding a solution to the following penalized least squares problem:

$$\underset{x\in\mathbb{R}^{TN}}{\text{minimize}} \;\; F(x) = \Phi(x) + \Psi(x), \tag{41}$$

where Φ denotes the data fidelity term given by

$$(\forall x \in \mathbb{R}^{TN}) \quad \Phi(x) = \frac{1}{2}\sum_{t=1}^{T} \|\mathrm{S}_t(h * x_t) - y_t\|^2, \tag{42}$$

and Ψ is a regularization function introducing prior informations on the sought video sequence defined as

$$(\forall x \in \mathbb{R}^{TN}) \quad \Psi(x) = \sum_{t=1}^{T} \Theta_t(x_t) + \iota_{[x_{\min}, x_{\max}]^{TN}}(x) + \mathrm{M}(x). \tag{43}$$

The indicator function $\iota_{[x_{\min}, x_{\max}]^{TN}}$ imposes a range $[x_{\min}, x_{\max}]$ on the pixel values of the images composing the video sequence. Θ_t is a spatial regularization term that manages each image $x_t \in \mathbb{R}^N$ independently, while M accounts for a temporal regularization function.

5.2.1 Spatial Regularization

For every $t \in \{1, \dots, T\}$, Θ_t incorporates prior information on each image $x_t \in \mathbb{R}^N$ and is defined as $\Theta_t(x_t) = \eta\, \mathrm{sltv}(x_t)$ where $\eta \geqslant 0$ and "sltv" denotes the semi-local total variation from [29]:

$$(\forall z \in \mathbb{R}^N) \quad \mathrm{sltv}(z) = \sum_{\ell \in \Omega} \chi\left(Dz - V_\ell Dz\right). \tag{44}$$

Hereabove, $D \in \mathbb{R}^{2N \times N}$ is the concatenation of the horizontal and vertical gradient operators:

$$D = \begin{bmatrix} \nabla_h \\ \nabla_v \end{bmatrix}, \quad \text{with} \quad \nabla_h \in \mathbb{R}^{N \times N}, \quad \nabla_v \in \mathbb{R}^{N \times N}, \tag{45}$$

$\Omega = \{1, \dots, 6\}$ and $(V_\ell)_{\ell \in \{1,\dots,6\}} \in \mathbb{R}^{2N \times 2N}$ represent shift operators as illustrated in Fig. 1. Moreover, $\chi : \mathbb{R}^{2N} \to \mathbb{R}$ is given by

$$\chi\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \sum_{n=1}^{N} \sqrt{((z_1)_n)^2 + ((z_2)_n)^2}. \tag{46}$$

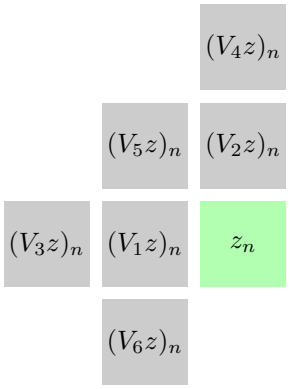


Figure 1: Shift operators $(V_\ell)_{\ell \in \{1,\dots,6\}}$ applied to a given pixel position $n \in \{1, \dots, N\}$.

Note that (44) can be rewritten as

$$(\forall z \in \mathbb{R}^N) \quad \mathrm{sltv}(z) = \sum_{\ell \in \Omega} \chi\left(L_\ell z\right) \text{ with } L_\ell = (I_{2N} - V_\ell)D. \tag{47}$$

5.2.2 Temporal Regularization

In (43), M represents a nonsmooth temporal regularization term similar to the one proposed in [30] that takes into account temporal redundancies. It is given by

$$(\forall x \in \mathbb{R}^{TN}) \quad \mathrm{M}(x) = \sum_{t=1}^{T} \sum_{\ell \in \mathcal{V}_t} \beta_{\ell,t} \|x_t - M_{\ell \to t} x_\ell\|_1, \tag{48}$$

where $\|\cdot\|_1$ denotes the $\mathbf{l}_1$ norm, in addition, for every t and ℓ, $\beta_{\ell,t}$ are positive weights, the index set $\mathcal{V}_t$ defines the neighborhood of the current image x_t (i.e., $\ell \in \mathcal{V}_t$ is such that $|\ell - t|$ is small), and $M_{\ell\to t} \in \mathbb{R}^{N\times N}$ is a linear operator modelling the sought motion fields between the current image x_t and the neighboring image x_ℓ of the video. The matrices $M_{\ell\to t}$ are related to some vertical and horizontal shift matrices $u_{\ell\to t} \in \mathbb{R}^{N_1\times N_2}$ and $v_{\ell\to t} \in \mathbb{R}^{N_1\times N_2}$ respectively, with N_1 (resp. N_2) corresponding to the height (resp. width) of the images (i.e., $N_1N_2 = N$), such a way that, $(\forall i \in \{1,\dots,N_1\})$ $(\forall j \in \{1,\dots,N_2\})$

$$M_{\ell\to t}x_\ell(i,j) \approx x_\ell\left(i - u_{\ell\to t}(i,j)\,,\, j - v_{\ell\to t}(i,j)\right). \tag{49}$$

More precisely, we set

$$u_{\ell\to t} = \overline{u}_{\ell\to t} + \delta^u_{\ell\to t} \quad \text{and} \quad v_{\ell\to t} = \overline{v}_{\ell\to t} + \delta^v_{\ell\to t}, \tag{50}$$

where $\overline{u}_{\ell\to t}$ and $\overline{v}_{\ell\to t}$ represent the integer part of $u_{\ell\to t}$ and $v_{\ell\to t}$ respectively, and $\delta^u_{\ell\to t}$, $\delta^v_{\ell\to t}$ are their decimal part. We propose to resort to the following bilinear interpolation in order to approximate (49):

$$\begin{aligned}
&(\forall i \in \{1,\dots,N_1\})\ (\forall j \in \{1,\dots,N_2\}) \quad M_{\ell\to t}x_\ell(i,j)\\
&= (1-\delta^u_{\ell\to t}(i,j))\,(1-\delta^v_{\ell\to t}(i,j))\,x_\ell\left(i-\overline{u}_{\ell\to t}(i,j)\,,\, j-\overline{v}_{\ell\to t}(i,j)\right)\\
&+ (1-\delta^u_{\ell\to t}(i,j))\,\delta^v_{\ell\to t}(i,j)x_\ell\left(i-\overline{u}_{\ell\to t}(i,j)\,,\, j-\overline{v}_{\ell\to t}(i,j)-1\right)\\
&+ \delta^u_{\ell\to t}(i,j)\,(1-\delta^v_{\ell\to t}(i,j))\,x_\ell\left(i-\overline{u}_{\ell\to t}(i,j)-1\,,\, j-\overline{v}_{\ell\to t}(i,j)\right)\\
&+ \delta^u_{\ell\to t}(i,j)\delta^v_{\ell\to t}(i,j)\,x_\ell\left(i-\overline{u}_{\ell\to t}(i,j)-1\,,\, j-\overline{v}_{\ell\to t}(i,j)-1\right).
\end{aligned} \tag{51}$$

Thus

$$M_{\ell\to t} = D_{1,\ell\to t}\,M_{1,\ell\to t} + D_{2,\ell\to t}\,M_{2,\ell\to t} + D_{3,\ell\to t}\,M_{3,\ell\to t} + D_{4,\ell\to t}\,M_{4,\ell\to t}, \tag{52}$$

where $D_{k,\ell\to t} \in \mathbb{R}^{N\times N}$ with $k \in \{1,\dots,4\}$ are diagonal matrices such that, for every $y \in \mathbb{R}^N$, for every $i \in \{1,\dots,N_1\}$ and for every $j \in \{1,\dots,N_2\}$:

$$\begin{aligned}
D_{1,\ell\to t}y(i,j) &= (1-\delta^u_{\ell\to t}(i,j))\ (1-\delta^v_{\ell\to t}(i,j))\ y(i,j),\\
D_{2,\ell\to t}y(i,j) &= (1-\delta^u_{\ell\to t}(i,j))\ \delta^v_{\ell\to t}(i,j)\ y(i,j),\\
D_{3,\ell\to t}y(i,j) &= \delta^u_{\ell\to t}(i,j)\ (1-\delta^v_{\ell\to t}(i,j))\ y(i,j),\\
D_{4,\ell\to t}y(i,j) &= \delta^u_{\ell\to t}(i,j)\ \delta^v_{\ell\to t}(i,j)\ y(i,j),
\end{aligned}$$

and $M_{k,\ell\to t} \in \{0,1\}^{N\times N}$, $k \in \{1,\dots,4\}$, are defined as

$$\begin{aligned}
M_{1,\ell\to t}y(i,j) &= y\left(i-\overline{u}_{\ell\to t}(i,j)\,,\, j-\overline{v}_{\ell\to t}(i,j)\right),\\
M_{2,\ell\to t}y(i,j) &= y\left(i-\overline{u}_{\ell\to t}(i,j)\,,\, j-\overline{v}_{\ell\to t}(i,j)-1\right),\\
M_{3,\ell\to t}y(i,j) &= y\left(i-\overline{u}_{\ell\to t}(i,j)-1\,,\, j-\overline{v}_{\ell\to t}(i,j)\right),\\
M_{4,\ell\to t}y(i,j) &= y\left(i-\overline{u}_{\ell\to t}(i,j)-1\,,\, j-\overline{v}_{\ell\to t}(i,j)-1\right).
\end{aligned}$$

The adjoint operator $(M_{k,\ell\to t})^\top$ is such that, for every $n \in \{1,\dots,N\}$, the n'-th component of $\left((M_{k,\ell\to t})^\top\, y\right)$ with $y \in \mathbb{R}^N$, corresponds to the sum of all the pixels located at $n \in \{1,\dots,N\}$ in the image y to which the pixel of index n' has been displaced in the resulting image $(M_{k,\ell\to t}y)$. Thereby, for every $n' \in \{1,\dots,N\}$

$$\left((M_{k,\ell\to t})^\top\,(D_{k,\ell\to t})^\top y\right)_{n'} = \sum_{n\in\mathcal{S}^k_{n',\ell\to t}} \left(D_{k,\ell\to t}\,y\right)_n, \tag{53}$$

where, for every $i \in \{1, \dots, N_1\}$ and for every $j \in \{1, \dots, N_2\}$,

$$\begin{aligned}
\mathcal{S}^1_{n',\ell\to t} &= \{n \mid i = i' + \overline{u}_{\ell\to t}(i,j)\ ;\ j = j' + \overline{v}_{\ell\to t}(i,j)\},\\
\mathcal{S}^2_{n',\ell\to t} &= \{n \mid i = i' + \overline{u}_{\ell\to t}(i,j)\ ;\ j = j' + \overline{v}_{\ell\to t}(i,j) + 1\},\\
\mathcal{S}^3_{n',\ell\to t} &= \{n \mid i = i' + \overline{u}_{\ell\to t}(i,j) + 1\ ;\ j = j' + \overline{v}_{\ell\to t}(i,j)\},\\
\mathcal{S}^4_{n',\ell\to t} &= \{n \mid i = i' + \overline{u}_{\ell\to t}(i,j) + 1; j = j' + \overline{v}_{\ell\to t}(i,j) + 1\},
\end{aligned}$$

and n (resp. n') is the index of the pixel located at (i,j) (resp. (i',j')) in the corresponding image.

According to (52), the norm of the motion compensation operator $M_{\ell\to t}$ reads

$$\|M_{\ell\to t}\| = \|\sum_{k=1}^{4} D_{k,\ell\to t} M_{k,\ell\to t}\| \ \leqslant \sum_{k=1}^{4} \|D_{k,\ell\to t} M_{k,\ell\to t}\|. \tag{54}$$

Note that, for every $k \in \{1, \dots, 4\}$, $M_{k,\ell\to t}$ is an $N \times N$ binary matrix. By definition, for every $n' \in \{1, \dots, N\}$, the n'-th column of this matrix has nonzero entries at the row indices $n \in \mathcal{S}^k_{n',\ell\to t}$. Therefore, since $D_{k,\ell\to t}$ is a diagonal matrix,

$$(M_{k,\ell\to t})^\top (D_{k,\ell\to t})^\top D_{k,\ell\to t} M_{k,\ell\to t}$$

is also diagonal with n'-th diagonal entry equals

$$\sum_{n\in\mathcal{S}^k_{n',\ell\to t}} D_{k,\ell\to t}(n,n)^2.$$

Thus, $\|D_{k,\ell\to t} M_{k,\ell\to t}\|$ can be easily computed according to

$$\|D_{k,\ell\to t} M_{k,\ell\to t}\| = \max_{n'\in\{1,\dots,N\}} \left(\sqrt{\sum_{n\in\mathcal{S}^k_{n',\ell\to t}} D_{k,\ell\to t}(n,n)^2} \right). \tag{55}$$

5.3 Minimization Strategy

A solution to Problem (41) is obtained by making use of PALM Algorithm recently proposed in [31] (see also [9] for recent extensions) which provides an asymptotically exact solution to (41). The images $(x_t)_{1\leqslant t\leqslant T}$ are processed sequentially, where at each iteration, an image x_t is updated with a forward-backward iteration that consists of gradient step on Φ with respect to x_t, and a proximal step on Ψ_t which represents the restriction of Ψ to the t-th image defined as: for every $x \in \mathbb{R}^{TN}$,

$$\begin{aligned}
(\forall z \in \mathbb{R}^N) \quad \Psi_t(z|x) = \eta \sum_{\ell\in\Omega} \chi\left(L_\ell z\right) \ + \ \iota_{[x_{\min}, x_{\max}]^N}(z)\\
+ \sum_{\ell\in\mathcal{V}_t} \beta_{\ell,t} \|z - M_{\ell\to t} x_\ell\|_1 + \sum_{\ell\in\mathcal{V}_t} \beta_{t,\ell} \|x_\ell - M_{t\to\ell} z\|_1,
\end{aligned} \tag{56}$$

where $(\beta_{\ell,t})_{\ell\in\mathcal{V}_t}$ and $(\beta_{t,\ell})_{\ell\in\mathcal{V}_t}$ are selected proportionally to the distance $|t-\ell|$ between the frame index of images x_t and x_ℓ. Thus, the number of terms in (56) is equal to

$$J = |\Omega| + 2\,|\mathcal{V}_t| + 1, \tag{57}$$

where $|\mathcal{Z}|$ denotes the cardinality of a set $\mathcal{Z}$. PALM algorithm reads:

Algorithm 10 PALM algorithm

Initialization:

Let $\left(x_t^0\right)_{1\leqslant t\leqslant T} \in \mathbb{R}^{TN}$

For every $k \in \mathbb{N}$ and $t \in \{1,\ldots,T\}$, $\sigma_t^k \in]0,+\infty[$

for $k = 0, 1, \ldots$ **do**

  for $t = 1,\ldots,T$ **do**

$$\breve{x}^{t,k} = \left(x_1^{k+1},\ldots,x_{t-1}^{k+1},\ x_t^k\ ,x_{t+1}^k\ldots,x_T^k\right)$$

$$\widetilde{x}_t^k = x_t^k - \sigma_t^k\left(\nabla_{x_t}\Phi(\breve{x}^{t,k})\right)$$

$$x_t^{k+1} = \mathrm{prox}_{(\sigma_t^k)^{-1}I_N,\Psi_t(\cdot|\breve{x}^{t,k})}\left(\widetilde{x}_t^k\right)$$

  end for

end for

where $\nabla_{x_t}\Phi$ denotes the gradient of Φ with respect to x_t and

$$0 < \sigma_t^k < 2\theta_t^{-1},$$

with θ_t the Lipschitz constant of $\nabla_{x_t}\Phi$ (i.e., $\theta_t = \|S_t H\|$ with $H \in \mathbb{R}^{N\times N}$ the Hankel-block Hankel matrix form of the convolution kernel h). Finally, $\mathrm{prox}_{(\sigma_t^k)^{-1}I_N,\Psi_t}$ represents the proximity operator of Ψ_t in the metric induced by $(\sigma_t^k)^{-1}I_N$. Since F is semi-algebraic and Φ is Lipschitz differentiable, the sequence $(x^k)_{k\in\mathbb{N}}$ generated by PALM algorithm is guaranteed to converge to the solution to Problem (41) [31].

As the proximity operator of (56) does not have a closed form expression and involves several linear operators, we resort to an inner iteration to estimate it by means of Algorithms 5, 6, and 8. Note that, when implementing Algorithm 5, function f in (22) is chosen equal to $\iota_{[x_{\min},x_{\max}]^N}$ since it does not involve any linear operator, whereas, in Algorithms 6 and 8, the latter function is regarded as some of the h_j functions, the corresponding A_j being the identity matrix. Let us emphasize that PALM algorithm is robust to computational errors in the proximal step [9], assuming that a sufficient decrease condition is satisfied. In practice, a rough stopping criterion on the inner loop will be used in order to avoid numerical instabilities.

5.4 Data-set Benchmark

We evaluate the performance of our methods (Figs. 2, 3, and 4) using a benchmark of four sequences of images (Figs. 5, 6, 7, 8, 9, and 10).

- Two synthetic video sequences **Foreman** and **Claire** of size $N = 352\times 288$ (resp. $N = 360\times 288$) composed of $T = 50$ frames. These video sequences were blurred with the horizontal convolution kernel shown in Fig. 5 which corresponds to a realistic model of the observed degradations in the context of old television archives, then interlaced and finally corrupted with a white Gaussian noise. The process results in a degraded video sequence with spatial dimension $L = 352\times 144$ (resp. $L = 360\times 144$). The videos are sourced from http://media.xiph.org/video/derf/.

- Two real interlaced sequences of size $L = 720\times 288$ supplied by INA from French broadcast archive programmes **Au théâtre ce soir** and **Tachan**. We extract $T = 50$ fields from each sequence and apply our method to recover progressive sharp video sequences with

resolution $N = 720 \times 576$.
For the deconvolution task, we use spatial convolution kernels shown in Fig. 8a and Fig. 8b, that are obtained using blind identification methods in [32] and [30].

These sequences are provided as RGB videos. We apply our method on their luminance component only, which represents a grayscale version of the original images, while the two chrominance components are processed with a median filter of size 3×3 on each component separately, in order to reduce the residual persistent noise. The motion matrices $(u_{\ell \to t}, v_{\ell \to t})$ involved in the temporal regularization term are computed from the luminance component of the degraded sequences using the method described in [33], and then spatially interpolated to reach the final resolution. The neighborhood $\mathcal{V}_t$ includes the previous and next frames of the image x_t. Moreover the set Ω involved in the semi-local total variation term is of size 6, so that the number of terms in (56) is equal to $J = 10$ or 11, namely:

- $(\forall j \in \{1, \ldots, 6\}) \quad h_j = \chi$ and $(A_j)_{1 \leqslant j \leqslant 6} = (L_\ell)_{\ell \in \Omega}$,
- $(\forall j \in \{7, 8\}) \quad h_j = \| \cdot \|_1$ and $A_j = I_N$,
- $(\forall j \in \{9, 10\}) \quad h_j = \| \cdot \|_1$ and $(A_j)_{9 \leqslant j \leqslant 10} = (M_{t \to \ell})_{\ell \in \mathcal{V}_t}$,
- For Algorithm 6 or 8, $\quad h_{11} = \iota_{[x_{\min}, x_{\max}]^N}$ and $A_{11} = I_N$.

5.5 Experimental Results

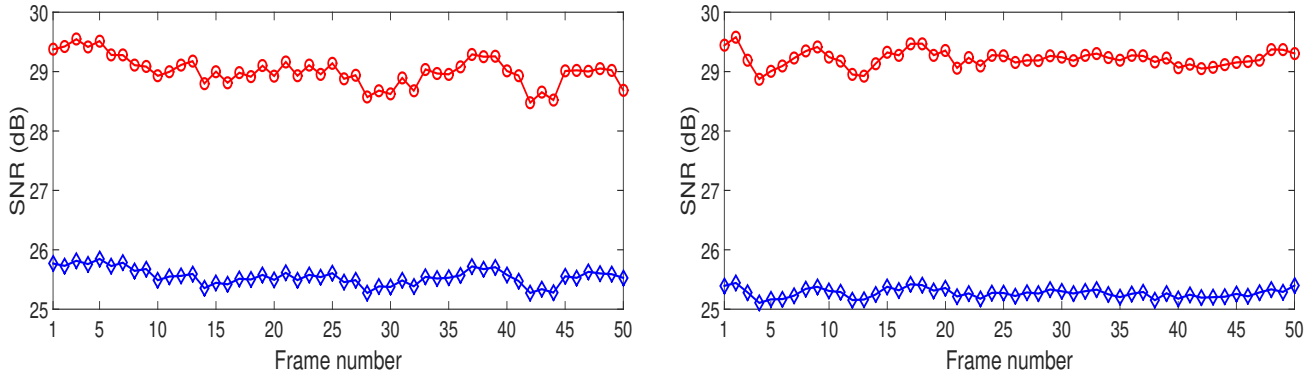
5.5.1 Restoration Quality

Table 1 presents the performance of our restoration method in terms of SNR, averaged SSIM [34], and MOVIE [35]. The latter is a video quality assessor, that takes into account both spatial and temporal aspects in the quality measurement. Moreover, the results in terms of SNR per frame are displayed in Figs 2a and 2b. The simulations are run using 100 iterations of PALM algorithm, which appears to be sufficient to reach the convergence of the method. Note that the values related to the degraded sequences are evaluated on a spatially interpolated version of them, with a final resolution equals to N. In addition, it should be mentioned that all the restoration results we obtained are similar in terms of visual quality, regardless of the chosen optimization algorithm.

Sequences		SNR (dB)	SSIM	MOVIE
Foreman	Degraded	25.54	0.78	4.34×10^{-4}
	Restored	28.95	0.90	3.73×10^{-4}
Claire	Degraded	25.27	0.85	1.97×10^{-3}
	Restored	29.21	0.96	1.77×10^{-3}

Table 1: Quality of our deinterlacing and deconvolution method.

Our reconstruction method achieves good quality results for all tested sequences. This can also be assessed by visual inspection on Figs 6 and 7, and for the real sequences on Figs 9 and 10, for which no ground truth is available. The motion compensation terms play a central role in the restoration quality, especially in the deinterlacing process. This is emphasized in the case of **Foreman** sequence, where the motion between two successive images is fast, which leads to a rough estimation of motion operators, at the price of a lower improvement of the restoration quality, especially in terms of MOVIE.



(a) **Foreman** sequence

(b) **Claire** sequence

Figure 2: SNR values per frame : Degraded (blue diamond), restored (red circle).

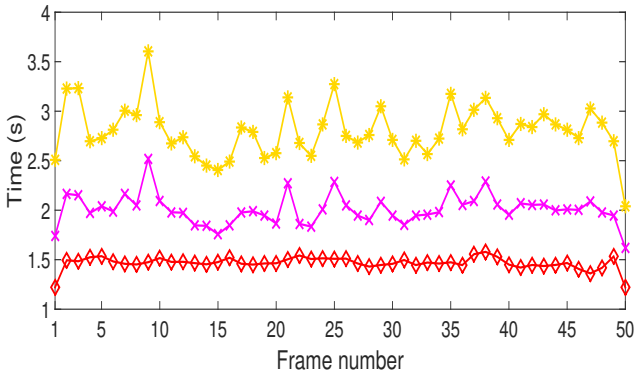
5.5.2 Convergence Speed

Let us analyse the convergence speed of the proposed algorithms. First, in order to investigate the impact of preconditioning strategies, we have carried out a number of tests regarding the preconditioning matrices related to the involved linear operators $(A_j)_{1\leqslant j\leqslant J}$. These evaluations are performed on the synthetic sequence **Foreman** using the minimization strategy described in section 5.3, combined with Algorithm 6. We work with diagonal preconditioning matrices in order to achieve a good trade-off between the convergence acceleration and the computation time.
The tested preconditioning matrices are

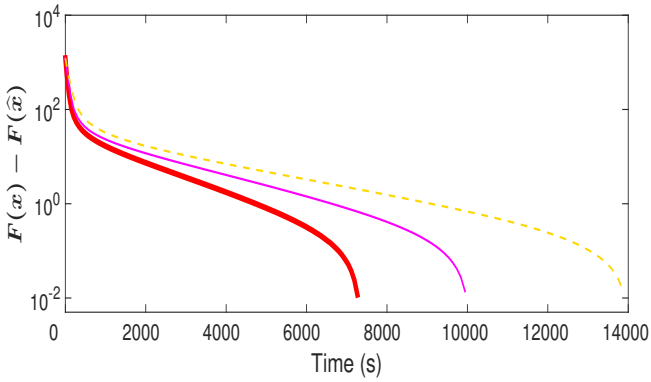
- $\forall j \in \{1,\dots,J\} \quad B_j = \|A_j\|^2 \, \mathrm{I}_{M_j}$, (58)
  and
- $\forall j \in \{1,\dots,J\} \quad B_j = \mathrm{Diag}\left(|A_j|\,|A_j^\top|\,1_{M_j}\right)$, (59)

where 1_{M_j} denotes the ones vector of $\mathbb{R}^{M_j}$.
In the non-preconditioned case (58), we need to supply the norms of the operators $(A_j)_{1\leqslant j\leqslant J}$. When it comes to motion compensation operators, this norm is either approximated using (54), or precomputed using the power iterative method in [36].

Fig. 3a presents the average execution time needed for computing the proximity operator of Ψ_t per image, by means of Algorithm 6. The latter is stopped when the relative decrease of the criterion gets below 10^{-5} which appears sufficient in practice to ensure the stability of the whole PALM algorithm. A Matlab 7 implementation is used with an Intel(R) Xeon(R) E5-2670 CPU @ 2.3 GHz. We get an acceleration of 50% using the preconditioning strategy (59) instead of the approximated version of the non-preconditioned case (58), while the acceleration is of 25% if the exact norms of the motion operators are used in (58). Note however that the exact computation of these norms is not a realistic strategy when processing long videos at standard or high resolution, since it calls upon an iterative and costly method. Figure 3b shows the variation of the cost function $F(x) - F(\widehat{x})$ with respect to the execution time, where F is defined in (41), and $F(\widehat{x})$ represents the minimum of F obtained at the end of the corresponding simulations. Figures 4a-4d illustrate the averaged time spent in computing the proximity operator of Ψ_t for all the images composing the video sequences over 100 iterations of PALM algorithm 10, using either Algorithm 5, 6, or 8. The preconditioning strategy (59)

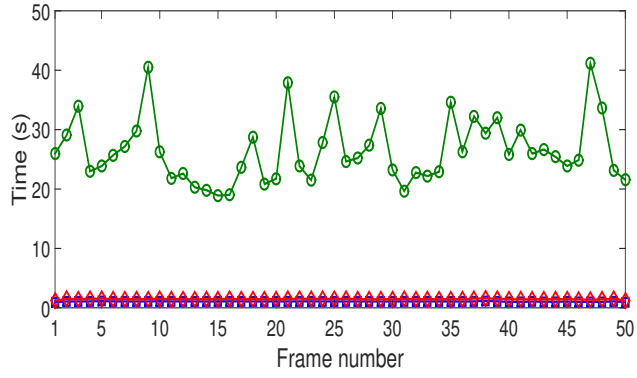


(a) Average execution time per frame for computing $\text{prox}_{(\sigma_t^k)^{-1}I_N,\Psi_t}$ using Algorithm 6 and **Foreman** sequence: preconditioning strategy (59) (red diamond), no preconditioning (see (58)) using exact norms (magenta cross), and no preconditioning strategy (58) using approximate norms (yellow astrick)).

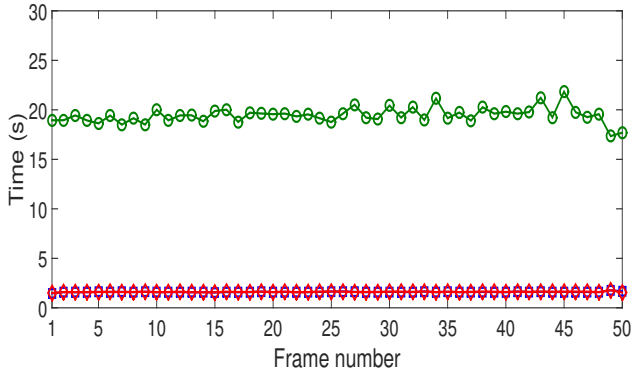


(b) Comparison of the preconditioning strategies in terms of execution time (s.): preconditioning strategy (59) (solid tick red), no preconditioning strategy (58) with exact norms (solid thin magenta) and no preconditioning strategy (58) with approximated norms (dashed thin yellow).

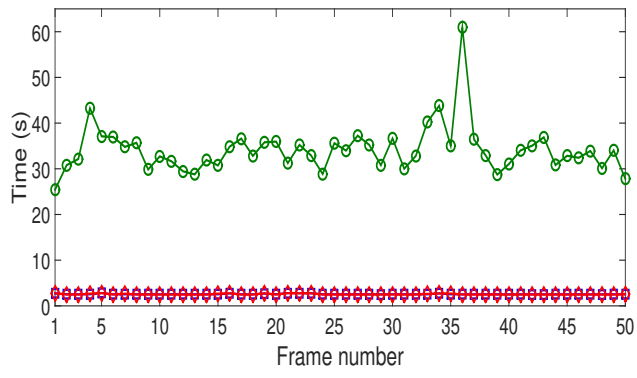
Figure 3: **Foreman** sequence : Convergence acceleration.



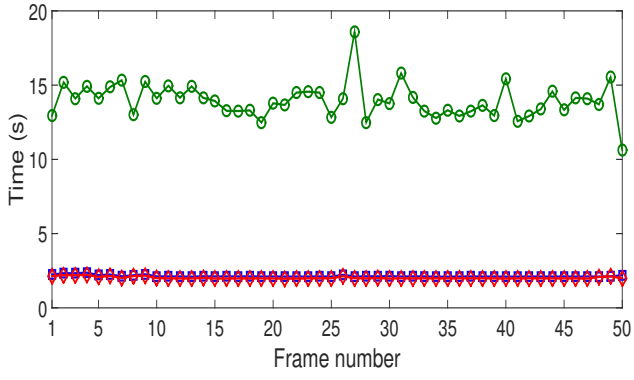
(a) **Foreman** sequence.



(b) **Claire** sequence.



(c) **Tachan** sequence.



(d) **Au théâtre ce soir** sequence.

Figure 4: Averaged execution time (in s.) per frame: Algorithm 5 (blue square), Algorithm 6 (red diamond) and Algorithm 8 (green circle).

is used for Algorithms 5 and 6. Depending on the video sequence, the best performance in terms of computation time are obtained either with Algorithm 5 or Algorithm 6 with small

differences between them. Moreover, the dual FB Algorithm 8 from [5] is up to 18 times slower to reach the stopping criterion. This emphasizes the gain provided by our algorithms in terms of acceleration.

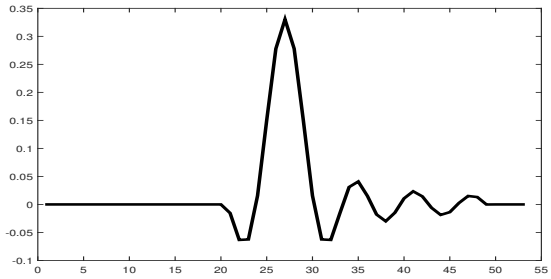


Figure 5: Synthetic spatial convolution kernel, $P = 53$.

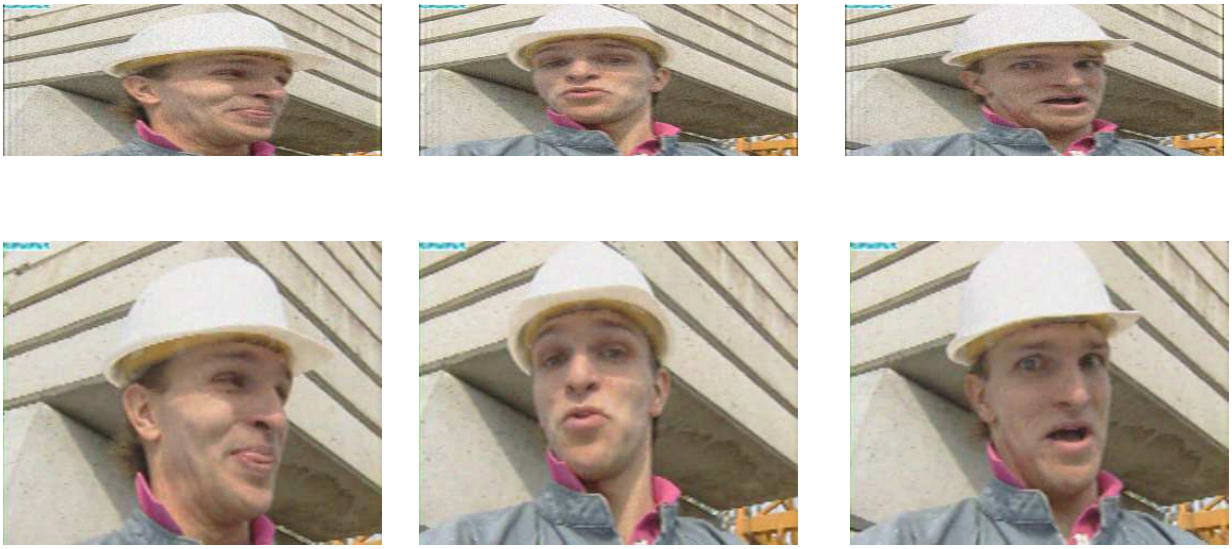
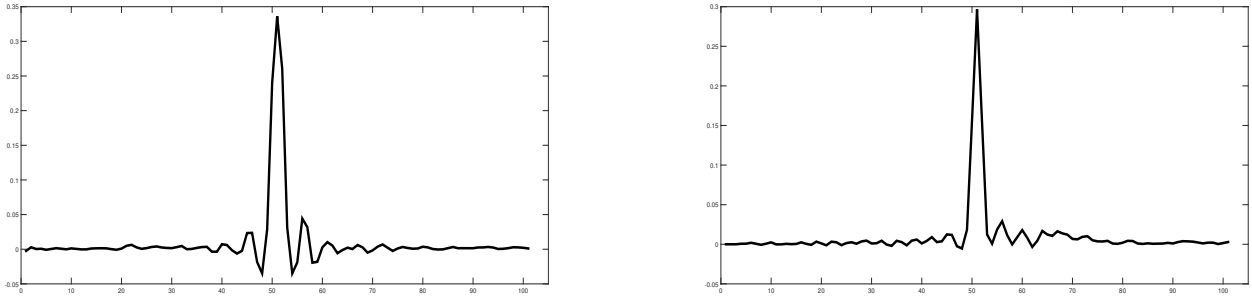


Figure 6: **Foreman** sequence: degraded low resolution fields (top), restored high resolution images (bottom).

Figure 7: **Claire** sequence: degraded low resolution fields (top), restored high resolution images (bottom).



(a) **Tachan** kernel.

(b) **Au théâtre ce soir** kernel.

Figure 8: Spatial convolution kernels for real sequences provided by INA, $P = 101$.

Figure 9: **Tachan** sequence: degraded low resolution fields (top), restored high resolution images (bottom).

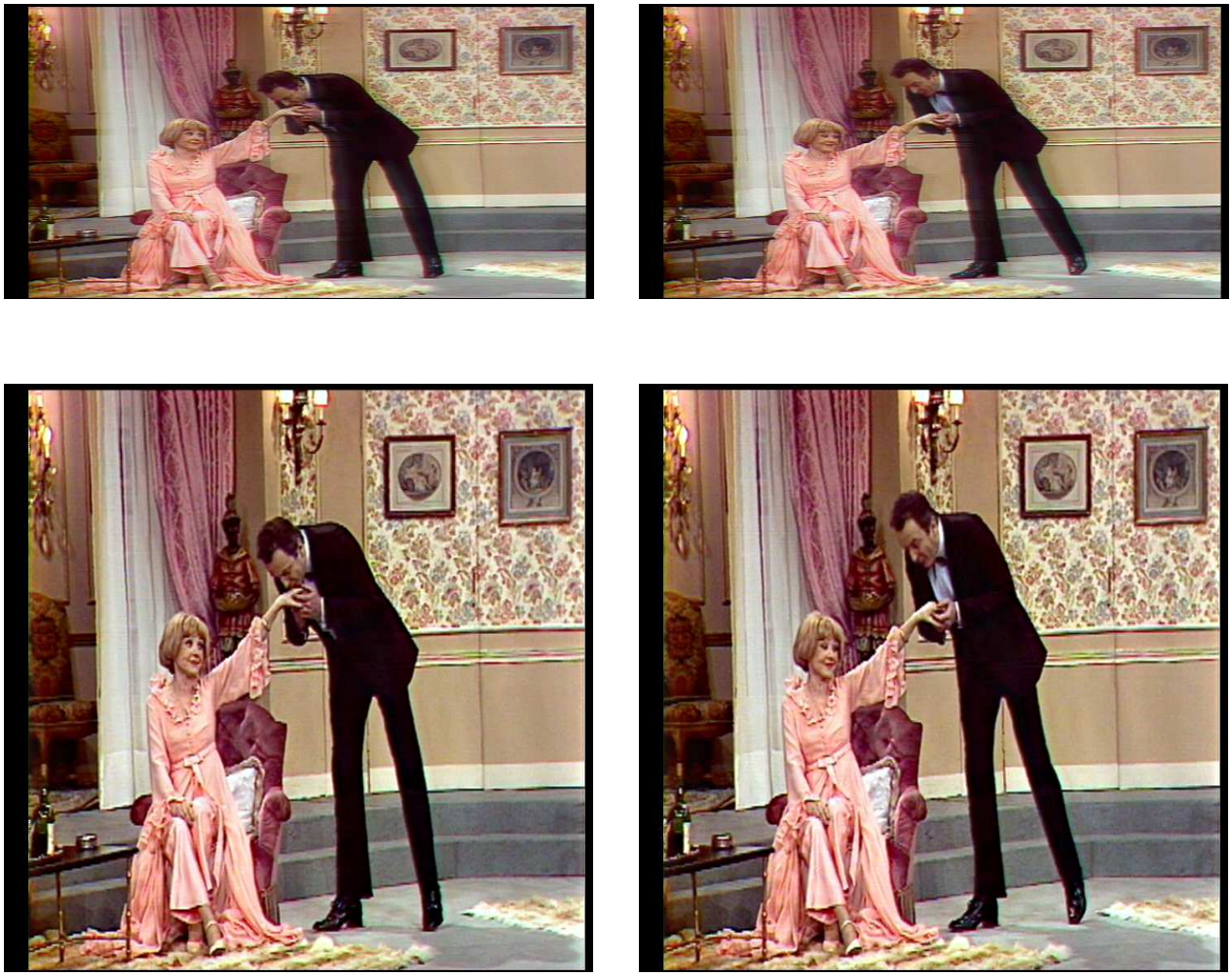


Figure 10: **Au théâtre ce soir** sequence: degraded low resolution fields (top), restored high resolution images (bottom).

6 Conclusion

We have proposed several primal-dual splitting algorithms for computing the proximity operator of convex composite functions. These algorithms can be applied in various areas. In our application, we have considered the joint problem of deconvolution and super-resolution enhancement of interlaced video sequences. The convergence of our approach has been theoretically analyzed and the experimental results provide an illustration of its good performance in terms of restoration quality and convergence speed.
In our future work, we intend to develop distributed versions of the proposed algorithms in order to solve large-scale optimization problems in a more computationally efficient manner, by exploiting the multi-core architectures of recent computer systems.

References

[1] P. L. Combettes and J.-C. Pesquet, "Proximal splitting methods in signal processing," in *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, H. H. Bauschke,

R. Burachik, P. L. Combettes, V. Elser, D. R. Luke, and H. Wolkowicz, Eds., pp. 185–212. Springer-Verlag, New York, 2010.

[2] P. L. Combettes and V. R. Wajs, "Signal recovery by proximal forward-backward splitting," *Multiscale Model. Simul.*, vol. 4, no. 4, pp. 1168–1200, 2005.

[3] Z. Xu, Z. Gan, and X. Zhu, "Compressed video super-resolution reconstruction based on regularized algorithm," in *6th IEEE Int. Conf. Signal Process. (ICSP 2006)*, Beijing, China, 16-20 Nov. 2006.

[4] X. Zhang, R. Xiong, S. Ma, and W. Gao, "A robust video super-resolution algorithm," in *28th IEEE Picture Coding Symp. (PCS 2010)*, Nagoya, Japan, 7-10 Dec. 2010, pp. 574–577.

[5] P. L. Combettes, D. Dũng, and B. C. Vũ, "Proximity for sums of composite functions," *J. Math. Anal. Appl.*, vol. 380, no. 2, pp. 680–688, Aug. 2011.

[6] H. H. Bauschke and P. L. Combettes, "A Dykstra-like algorithm for two monotone operators," *Pac. J. Optim.*, vol. 4, no. 3, pp. 383–391, 2008.

[7] X. Fu, B. He, X. Wang, and X. Yuan, "Block wise alternating direction method of multipliers with gaussian back substitution for multiple block convex programming," 2014, http://www.optimization-online.org/DB_FILE/2014/09/4544.pdf.

[8] 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. 8, no. 1, pp. 1–122, Jan. 2011.

[9] E. Chouzenoux, J.-C. Pesquet, and A. Repetti, "A block coordinate variable metric forward-backward algorithm," *To appear in J. Global Optim.*, 2016.

[10] P. Richtárik and M. Takác, "Efficient serial and parallel coordinate descent methods for huge-scale truss topology design," in *Int. Conf. Oper. Res. (OR 2011)*, 2011, pp. 27–32.

[11] A. Saha and A. Tewari, "On the finite time convergence of cyclic coordinate descent methods," *SIAM J. Optim.*, vol. 23, no. 1, pp. 576–601, 2013.

[12] M. Jaggi, V. Smith, M. Takác, J. Terhorst, S. Krishnan, T. Hofmann, and M. I. Jordan, "Communication-efficient distributed dual coordinate ascent," in *Adv. Neural Inf. Process. Syst. (NIPS 2014)*, pp. 3068–3076. Montréal, Canada, 8-13 Dec. 2014.

[13] P. L. Combettes and J.-C. Pesquet, "Stochastic quasi-Fejér block-coordinate fixed point iterations with random sweeping," *SIAM J. Optim.*, vol. 25, no. 2, pp. 1221–1248, Jul. 2015.

[14] J.-C. Pesquet and A. Repetti, "A class of randomized primal-dual algorithms for distributed optimization," *J. Nonlinear Convex Anal.*, vol. 16, no. 12, pp. 2453–2490, Dec. 2015.

[15] S. Shalev-Shwartz and T. Zhang, "Stochastic dual coordinate ascent methods for regularized loss minimization," *J. Mach. Learn. Res.*, vol. 14, no. 1, pp. 567–599, Feb. 2013.

[16] Z. Qu, P. Richtárik, and T. Zhang, "Randomized dual coordinate ascent with arbitrary sampling," in *Adv. Neural Inf. Process. Syst. (NIPS 2015)*, Montréal, Canada, 7-12 Dec. 2015, pp. 865–873.

[17] A. Chambolle and T. Pock, "A remark on accelerated block coordinate descent for computing the proximity operators of a sum of convex functions," *SMAI J. Comput. Math.*, vol. 1, pp. 29–54, 2015.

[18] P. L. Combettes, D. Dung, and B. C. Vũ, "Dualization of signal recovery problems," *Set-Valued Var. Anal.*, vol. 18, no. 3, pp. 373–404, Dec. 2010.

[19] S. Lojasiewicz, "Une propriété topologique des sous-ensembles analytiques réels," *Editions du centre National de la Recherche Scientifique*, pp. 87–89, 1963.

[20] D. Csiba, Z. Qu, and P. Richtárik, "Stochastic dual coordinate ascent with adaptive probabilities," in *32nd Int. Conf. on Mach. Learn. (ICML 2015)*, Lille, France, 1-6 July 2015, pp. 674–683.

[21] D. A. Lorenz, S. Wenger, F. Schöpfer, and M. A. Magnor, "A sparse Kaczmarz solver and a linearized Bregman method for online compressed sensing," in *IEEE Int. Conf. Image Process. (ICIP 2014)*, Paris, France, 27-30 Oct. 2014, pp. 1347–1351.

[22] P. L. Combettes and B. C. Vũ, "Variable metric forward-backward splitting with applications to monotone inclusions in duality," *Optimization*, vol. 63, no. 9, pp. 1289–1318, Sept. 2014.

[23] E. Chouzenoux, J.-C. Pesquet, and A. Repetti, "Variable metric forward-backward algorithm for minimizing the sum of a differentiable function and a convex function," *J. Optim. Theory App.*, vol. 162, no. 1, pp. 107–132, Jul. 2014.

[24] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, Springer, New York, 2011.

[25] J. Bolte, T. P. Nguyen, J. Peypouquet, and B. Suter, "From error bounds to the complexity of first-order descent methods for convex functions," Tech. Rep., 2015, http://www.optimization-online.org/DB_HTML/2015/10/5176.html.

[26] S. Mallat, "Super resolution bandlet upconversion for HD TV," Tech. Rep., 2006, http://www.di.ens.fr/∼mallat/papiers/whitepaper.pdf.

[27] S. H. Keller, F. Lauze, and M. Nielsen, "A total variation motion adaptive deinterlacing scheme," in *5th Int. Conf., Scale-Space 2005*, Hofgeismar, Germany, 7-9 Apr. 2005, pp. 408–418.

[28] S. H. Keller, *Video Upscaling Using Variational Methods*, Ph.D. thesis, The Image Group, Department of Computer Science Faculty of Science, University of Copenhagen, 2007.

[29] L. Condat, "Semi-local total variation for regularization of inverse problems," in *22nd IEEE Eur. Signal Process. Conf. (EUSIPCO 2014)*, Lisbon, Portugal, 1-5 Sep. 2014, pp. 1806–1810.

[30] F. Abboud, E. Chouzenoux, J.-C. Pesquet, J.-H. Chenot, and L. Laborelli, "A hybrid alternating proximal method for blind video restoration," in *22nd IEEE Eur. Signal Process. Conf. (EUSIPCO 2014)*, Lisbon, Portugal, 1-5 Sep. 2014, pp. 1811–1815.

[31] J. Bolte, S. Sabach, and M. Teboulle, "Proximal alternating linearized minimization for nonconvex and nonsmooth problems," *Math. Program.*, vol. 146, no. 1-2, pp. 459–494, Aug. 2014.

[32] D. Krishnan, T. Tay, and R. Fergus, "Blind deconvolution using a normalized sparsity measure," in *IEEE Conf. Comput. Vis. Pattern Recogn. (CVPR 2011)*, Colorado Springs, CO, USA, 21-25 Jun. 2011, pp. 233–240.

[33] C. Liu, W. T. Freeman, E. H. Adelson, and Y. Weiss, "Human-assisted motion annotation," in *IEEE Conf. Comput. Vis. Pattern Recogn. (CVPR 2008)*, Anchorage, Alaska, USA, 23-28 Jun. 2008, pp. 1–8.

[34] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. on Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[35] k. Seshadrinathan and A. C. Bovik, "Motion tuned spatio-temporal quality assessment of natural videos," *IEEE Trans. on Image Process.*, vol. 19, no. 2, pp. 335–350, Feb. 2010.

[36] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, USA, 3 edition, 1996.