Estimation of super-resolved video dynamics

Patrick Héas, Angélique Drémeau, Cédric Herzet

ESTIMATION OF SUPER-RESOLVED VIDEO DYNAMICS

P. HÉAS*, A. DRÉMEAU†, AND C. HERZET*

Abstract. In this work, we propose an efficient methodology for video super-resolution, that is, the recovery of a sequence of high-resolution images from its low-resolution counterpart. The optimization problem associated to video super-resolution has several specificities which makes it particularly challenging. A first barrier is the high-dimensionality of the problem, which derives from the extra temporal dimension and the unknown parametrization of the dynamical model characterizing the video. A second obstacle is the non-differentiability and the non-convexity of some of the terms of the cost function: the non-differentiability stems from the use of regularization terms of the state of the art (*e.g.*, to enforce sparsity) whereas the non-convexity appears as soon as the motion describing the video is unknown.

In this paper, we propose an overall algorithmic framework to address the video super-resolution problem. Our approach is based on fast gradient evaluation methods and modern optimization techniques for non-differentiable/non-convex problems. As a consequence, unlike previous work in the field, we show that there exists a provably-convergent method estimating both the high-resolution image sequence and the underlying motion with a complexity linear in the problem dimensions. We assess the proposed optimization methods on videos of the *MPI Sintel* data set, known to be a challenging optical-flow benchmark.

Key words. Sparse models, non-convex optimization, optimal control, video super-resolution.

1. Introduction. Super resolution (SR) aims at reconstructing high-resolution (HR) images from distorted low-resolution (LR) observations. This type of methodology dates back to the 70's with the pioneering work of Gerchberg [27] and Santis&Gori [17]. Since then, super resolution has been applied to a large variety of applicative domains, including infrared [29], medical [42], satellite and aerial [38, 45] imaging. We refer the reader to [34] for a pretty comprehensive overview of the works dealing with SR. In this paper, we focus on the particular problem of "video SR" where a sequence of HR images has to be recovered from the observation of its LR counterpart. So far, in comparison with the huge number of papers dealing with SR, only a few have focussed on this particular problem, see [44, 22, 37, 4, 20, 19, 48, 55, 24, 23, 14, 15, 26, 28, 16, 30].

Video SR has some specificities which make it particularly challenging. First, because it adds a temporal dimension to the SR problem, the number of variables involved in the HR reconstruction may become very large. There is thus a need for very efficient methodologies to deal with such high-dimensional problems. In [44], the authors modeled the dependence between the different images of the sequence as a Gaussian process and provided an efficient implementation in the Fourier domain. Other contributions relied on adaptive-filtering techniques, see [22, 20, 19, 48, 23, 28]. In this line of thought, most of the contributions cited above considered that the HR sequence is ruled by a state-space evolution model and the authors derived estimation procedures inspired by the well-known Kalman filter. The standard Kalman updates having a prohibitive complexity in the context of video SR, Elad and co-authors published a series of papers [22, 20, 19, 23] in which they proposed updates having a linear complexity in the problem dimensions. Their approach is based on some approximations of the model and/or Kalman updates (*e.g.*, uniform translational

*INRIA Centre Rennes - Bretagne Atlantique, campus universitaire de Beaulieu, 35042 Rennes, France (patrick.heas@inria.fr, cedric.herzet@inria.fr)

†ENSTA Bretagne, 2 rue Francois Verny, 29806 Brest, France (angelique.dremeau@ensta-bretagne.fr)

motion [23], noise-free evolution model [19], etc). A first contribution of this paper will be to emphasize that such linear complexity can be achieved without resorting to any approximation. In particular, we propose a methodology to compute the exact solution of the Kalman smoother with a complexity scaling linearly with the problem dimensions. Our approach is based on efficient gradient computation techniques, well-known in the domain of optimal control, see [6].

A second problem inherent to video SR stands in the precise characterization of the model connecting the different images of the sequence. In the context of video SR, this model is typically characterized by non-global motions. This is in contrast with many standard "multi-frame" SR models of the literature which assume global motions (*e.g.*, translation [1], affine [51] or projection [13]), well-suited to still image reconstruction. The imaging model in video SR thus takes a more-involved form and has to be considered with care. In particular, the estimation of the motion between two consecutive frames is usually tantamount to solving an optical-flow problem [4]. Embedding motion estimation in the SR reconstruction introduces new difficulties: *i)* it increases the problem dimensionality since two additional unknowns (the displacement in each direction) have to be estimated for each pixel of the HR images; *ii)* it typically introduces non-linearities in the image formation model. As a consequence, the estimation of the motion between successive frames has been overlooked in many contributions mentioned above, with the exception of [4, 26, 30]. Interestingly, several authors have emphasized the importance of accurate motion estimation in the video SR process and provided studies of the sensibility of adaptive filtering techniques to the latter, see [55, 14, 15, 16]. In this paper, we show that the motion can be included, without significant additional cost, to the Kalman smoothing process, by relying on the gradient computation techniques mentioned above.

Finally, a last problem of SR video stands in its ill-posedness. Indeed, typical setups impose the observation of (at most) one LR image for each frame of the HR sequence. If the motion between the different frame is supposed to be unknown, it is easy to see that the number of variables which have to be estimated is well beyond the number of observations. In order to tackle this difficulty, a well-known technique consists in resorting to prior information on the sought quantities. This type of approach has been used extensively (but not only) in the context of single-image SR, where an HR image has to be reconstructed from one single LR observation. First methodologies based on prior information dates back to the 70's [27, 17]. Since then, many type of priors have been studied, including Markov random fields [43], total variation [53], morphological [40] or sparse [52] models , etc. Among the most effective models in the literature, many of them impose the minimization of some non-differentiable functions. It is for example the case for SR techniques based on sparse representations where the decomposition coefficients of the sought quantity in a redundant dictionary are commonly penalized by an ℓ_1 norm, see *e.g.*, [25]. Another example is total variation where the ℓ_1 norm is applied to the gradient of the sought images/motions, see [53]. The introduction of non-differentiable functions in the SR reconstruction leads to new conundrums since standard optimization techniques for smooth problems can no longer be applied. We address this point in the paper as well.

Hereafter, we mainly focus on problems involving an ℓ_1 norm, although other non-differentiable convex functions could be processed using a procedure similar to the one exposed in this paper. We proceed in two steps. First, the motion connecting the HR frames is supposed to be known. In this case, our SR formulation leads

to a convex non-differentiable problem. We show that the latter can be solved iteratively by resorting to the fast implementation of the Kalman smoother, described previously, and simple soft-thresholding operations. Our approach relies on the well-known "Alternating Direction Method of Multipliers" [7], a fairly recent optimization techniques well-suited to large-scale non-differentiable problems. The proposed procedure is provably convergent to a global minimum of the SR problem and its overall complexity remains linear with the problem dimensions. In a second step, we consider the reconstruction of both the HR sequence and the inter-frame motion. We focus on a recent procedure for non-convex and non-differentiable optimization proposed by Attouch *et al.* in [2, 3] and particularized to standard (multi-frame) SR in [41]. The resulting procedure is iterative and provably convergent to a critical point of our cost function. Moreover, as previously, the complexity of the algorithm scales linearly with the problem dimensions.

The remainder of the paper is organized as follows. We introduce the notations used throughout the paper in section 2. In section 3, we present the video SR model considered in our subsequent derivations. In section 4, we express the video SR problem as a constrained optimization problem and provide the algorithmic building blocks to solve it efficiently. Different scenarios are considered. In section 4.1, we address the case where all the functions involved in the cost function are differentiable. In section 4.2, we assume that the inter-frame motion is known but the cost function contains some non-differentiable terms. Finally, in subsection 4.3, we provide an algorithmic framework for the general problem where both the HR sequence and the inter-frame motion have to be estimated, and the cost function is non-differentiable. The numerical evaluation of the proposed methods is presented in section 5 for different experimental setups.

2. Notations. The notational conventions adopted in this paper are as follows. Italic lowercase indicates a scalar quantity, as in a; boldface lowercase (resp. uppercase) indicates a vector (resp. matrix) quantity, as in $\mathbf{a}$ (resp. $\mathbf{A}$). The n-dimensional vector of zeroes and identity matrix will be written as $\mathbf{0}_n$ and $\mathbf{I}_n$. The i-th element of vector $\mathbf{a}$ is denoted $\mathbf{a}(i)$; similarly $\mathbf{A}(i,j)$ is the element of $\mathbf{A}$ located at row i and column j. The exponent * denotes the transpose operation. A subscript notation, as in $\mathbf{a}_t$, will refer to the member of some sequence $\{\mathbf{a}_t\}_{t=0}^T = \{\mathbf{a}_0, \mathbf{a}_1, \cdots, \mathbf{a}_T\}$.

Calligraphic letters, as $\mathcal{H}$, denote functions. The subscript notation $\mathcal{H}_i$ may either denote the i-th element of a set $\{\mathcal{H}_i\}_i$ or the i-th component of a multidimensional function $\mathcal{H} : \mathbb{R}^m \to \mathbb{R}^n$; the distinction between these two notations is usually clear from the context. The Jacobian matrix of $\mathcal{H} : \mathbb{R}^m \to \mathbb{R}^n$ evaluated at $\tilde{\mathbf{a}}$, denoted by $\nabla_{\mathbf{a}}\mathcal{H}(\tilde{\mathbf{a}})$, is defined as:

$$\nabla_{\mathbf{a}}\mathcal{H}(\tilde{\mathbf{a}}) = \begin{pmatrix} \partial_{\mathbf{a}(1)}\mathcal{H}_1(\tilde{\mathbf{a}}) & \cdots & \partial_{\mathbf{a}(m)}\mathcal{H}_1(\tilde{\mathbf{a}}) \\ \vdots & & \vdots \\ \partial_{\mathbf{a}(1)}\mathcal{H}_n(\tilde{\mathbf{a}}) & \cdots & \partial_{\mathbf{a}(m)}\mathcal{H}_n(\tilde{\mathbf{a}}) \end{pmatrix} \in \mathbb{R}^{n\times m},$$

where ∂_v is the partial derivative operator with respect to v. We use the notation $\nabla_{\mathbf{a}}\mathcal{H}^*(\tilde{\mathbf{a}})$ to denote the transpose of $\nabla_{\mathbf{a}}\mathcal{H}(\tilde{\mathbf{a}})$.

3. Model and General Problem. Let $\mathbf{x}_t \in \mathbb{R}^n$ be the image at time t of an HR video sequence rearranged into a n-dimensional vector, with $t \in \{0, \ldots, T\}$. Let

us suppose that we capture noisy and low-resolution (LR) observations $\mathbf{y}_t \in \mathbb{R}^m$ with $m \leq n$ of the HR sequence: $\forall t \in \{0, \ldots, T\}$,

$$\mathbf{y}_t = \mathcal{H}(\mathbf{x}_t) + \boldsymbol{\eta}_t, \tag{3.1}$$

where $\boldsymbol{\eta}_t \in \mathbb{R}^m$ stands for some noise and $\mathcal{H} : \mathbb{R}^n \to \mathbb{R}^m$ denotes a linear function, which is the composition of a low-pass filtering and a sub-sampling operation. We focus on the problem of recovering the HR sequence $\{\mathbf{x}_t\}_{t=0}^T$ from the LR observations $\{\mathbf{y}_t\}_{t=0}^T$.

Without any additional information, this problem is ill-conditioned since the number of unknowns (that is $(T+1)n$) is larger than the number of observations (that is $(T+1)m$). One way to circumvent this problem is to take into account the relation existing between the HR images at different time instants. More specifically, as part of a video, we can assume that two consecutive images obey the following dynamical model[1]:

$$\mathbf{x}_t = \mathcal{P}(\mathbf{x}_{t+1}, \mathbf{d}_{t+1}) + \boldsymbol{\epsilon}_{t+1}, \tag{3.2}$$

where $\mathcal{P} : \mathbb{R}^n \times \mathbb{R}^{2n} \to \mathbb{R}^n$ is a "warping" function characterized by a displacement $\mathbf{d}_{t+1} \in \mathbb{R}^{2n}$, and $\boldsymbol{\epsilon}_{t+1} \in \mathbb{R}^n$ is some noise. The choice of $\mathcal{P}$ is usually motivated by some conservation property, as for example the preservation of the pixel intensity along the displacement. One particular instance of function $\mathcal{P}$, that we will consider in the sequel, is based on the well-known "Displaced Frame Difference" model. More specifically, this model assumes that the s-th component of $\mathcal{P}(\mathbf{x}_t, \mathbf{d}_t)$ admits the following series representation:

$$\mathcal{P}_s(\mathbf{x}_t, \mathbf{d}_t) = \sum_{i \in \mathcal{V}(\chi(s)+\mathbf{d}_t(s))} \mathbf{x}_t(i)\psi_i(\chi(s) + \mathbf{d}_t(s)), \tag{3.3}$$

where $\chi : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a function returning the spatial position corresponding to index s, where $\mathcal{V}(\chi(s) + \mathbf{d}_t(s))$ denotes a subset of indices corresponding to the "neighborhood" of point $\chi(s) + \mathbf{d}_t(s)$ and where $\{\psi_i\}_{i=1}^n$ with $\psi_i : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a family of bi-dimensional polynomial interpolation functions. In this case, (3.2)-(3.3) models the fact that $\mathbf{x}_t$ can be seen as a displaced version of $\mathbf{x}_{t+1}$ plus some additive noise. Let us note that $\mathcal{P}$, as defined in (3.3), is linear in $\mathbf{x}_t$ and polynomial in $\mathbf{d}_t$; it is thus a bi-polynomial function. Let us also mention that $\mathcal{V}_s$ typically only contains a few elements that is $|\mathcal{V}_s| \ll n$, where $|\mathcal{V}_s|$ denotes the cardinality of $\mathcal{V}_s$; this observation will play an important role in the sequel for the analysis of the complexity of the proposed SR methodology.

The noise $\boldsymbol{\epsilon}_{t+1}$ in (3.2) accounts for all the modifications of the image $\mathbf{x}_t$ which cannot be inferred from $\mathbf{x}_{t+1}$ and $\mathbf{d}_{t+1}$. This includes pixel occlusions, interpolation errors or variations of the scene illumination. Notice that, in practice, the choice of $\mathcal{P}$ should be made such that the residual noise $\boldsymbol{\epsilon}_{t+1}$ is as small as possible. In particular, if $\boldsymbol{\epsilon}_{t+1} = 0$ (and $\mathbf{d}_{t+1}$ is known), $\mathbf{x}_t$ is entirely determined from $\mathbf{x}_{t+1}$ $\forall t$. Recovering the whole sequence $\{\mathbf{x}_t\}_{t=0}^T$ is then tantamount to recovering the last image $\mathbf{x}_T$. In such a case, the number of unknowns is therefore reduced to n and the recovery of the HR sequence from the LR images may be possible.

[1] We note that *backward* dynamical models such as (3.2) are common in the computer-vision literature. We therefore restrict our reasoning to the latter formulation. However, adapting the methodologies derived in section 4 to a *forward* dynamical model is straightforward.

Another option to decrease the ill-possedness of the video SR problem consists in restricting the family of signals to which the "initial condition"[2] $\mathbf{x}_T$ belongs. In this paper, we will study several restrictions. We will in particular consider the case where $\mathbf{x}_T$ is assumed to be sparse in some (possibly redundant) dictionary $\mathbf{D} \in \mathbb{R}^{n\times q}$, that is

$$\mathbf{x}_T = \mathbf{D}\mathbf{c} \quad \text{for some } \mathbf{c} \in \mathbb{R}^q \text{ such that } \|\mathbf{c}\|_0 \ll n, \tag{3.4}$$

where $\|.\|_0$ is the so-called "ℓ_0 norm", which returns the number of nonzero coefficients of its argument. Dealing with $\|.\|_0$ leads to combinatorial optimization problems. Hereafter, we will restrict our attention to the case where $\|\mathbf{c}\|_1 \ll n$. The ℓ_1 norm is a well-known surrogate to the ℓ_0 norm in the literature. In particular, if the sparsity of the sought vector is large enough, there exists an equivalence between the solution of the problems involving the ℓ_0 and ℓ_1 norms, see [25].

Finally, let us mention that the displacement $\mathbf{d}_t$ between two successive images is rarely known in practice. It must therefore be inferred from the received LR images $\{\mathbf{y}_t\}_{t=0}^T$. This may seem to be counterproductive since the estimation of $\mathbf{d}_t$ implies an increase of the number of unknowns of $2n$ elements per time step. One way to circumvent this problem consists again in constraining $\mathbf{d}_t$ to belong to a restricted family of signals. In the "multi-frame" SR setup, this restriction is usually performed by assuming that the displacements are global geometric transformations parametrized by a limited number of coefficients, as for example translations, affine or projection models. Although in specific settings, parametric models may approach reasonably well the displacements occurring in videos, a global transformation assumption appears to be abusive in general video SR problems. An alternative is to perform an implicit restriction by enforcing some non-negative function of $\mathbf{d}_t$ to be small. Following this approach, we will assume in the sequel that the sought displacement is such that $\mathcal{R}(\mathbf{G}^*\mathbf{d}_t)$ is "small", where $\mathbf{G} = [\mathbf{g}_1, \ldots, \mathbf{g}_h] \in \mathbb{R}^{2n\times h}$ is some linear "analysis" operator and $\mathcal{R} : \mathbb{R}^h \to \mathbb{R}$ is a nonnegative function. We note that this approach is commonly adopted in the computer-vision literature in which many options for $\mathcal{R}$ and $\mathbf{G}$ have been proposed, see [5]. In the sequel, we will focus on the following choice for $\mathbf{G}$ and $\mathcal{R}$: the elements of $\mathbf{G}^*\mathbf{d}_t$ will correspond to the spatial gradients of (an interpolation of) $\mathbf{d}_t$ at each point of the pixel grid; $\mathcal{R}(\mathbf{G}^*\mathbf{d}_t)$ takes the following form:

$$\mathcal{R}(\mathbf{G}^*\mathbf{d}_t) \triangleq \sum_{i=1}^{n} \mathbf{w}(i) \Big| \sum_{j\in\mathcal{S}_i} (\mathbf{g}_j^*\mathbf{d}_t)^2 \Big|^{\frac{p}{2}}, \quad p \geq 0, \tag{3.5}$$

where $\mathbf{w}$ is defined as a vector of weights and $\mathcal{S}_i$'s denote disjoint subsets of elements of $\{1, \ldots, h\}$. Index i represents a location on the pixel grid. The subset $\mathcal{S}_i$ typically gathers 4 elements corresponding to the 2 spatial gradients of the 2 components of motion $\mathbf{d}_t$ at the location indexed by i. For $p = 2$, these choices are equivalent to constraining the spatial gradient of the displacement field $\mathbf{d}_t$ by a quadratic penalization [49], whereas the case $p = 1$ corresponds to the weighted total variation (TV) approach suggested in [54].

In summary, (3.1)-(3.4) together with the definition of $\mathbf{G}$ and $\mathcal{R}$ specify our prior/observation model. In the next section, we will present a low-complexity methodology to recover the HR sequence from the collected observations $\{\mathbf{y}_t\}_{t=0}^T$ by properly exploiting this model. More specifically, we will assume that the unknowns of the problem include the HR sequence $\{\mathbf{x}_t\}_{t=0}^T$, the dynamical noise $\{\boldsymbol{\epsilon}_t\}_{t=1}^T$, the displacements

[2] We remind the reader that we consider a *backward* dynamical model.

$\{\mathbf{d}_t\}_{t=1}^T$ and the decomposition vector $\mathbf{c}$. All the other parameters of the problem will be supposed to be known, although they could easily be included as additional unknowns without introducing any conceptual problem in the proposed methodology.

4. Estimation of Super-resolved Dynamics. In this section, we expose our methodology to estimate the HR sequence by exploiting the model described in section 3. Our approach is based on the resolution of a constrained optimization problem. For the sake of clarity, we introduce the following shorthand notations: $\mathbf{x} \triangleq \{\mathbf{x}_t\}_{t=0}^T$, $\boldsymbol{\epsilon} \triangleq \{\boldsymbol{\epsilon}_t\}_{t=1}^T$ and $\mathbf{d} \triangleq \{\mathbf{d}_t\}_{t=1}^T$. Our SR reconstruction procedure is based on the following constrained optimization problem:

$$\underset{(\mathbf{x},\boldsymbol{\epsilon},\mathbf{d},\mathbf{c})}{\arg\min}\, \mathcal{J}(\mathbf{x},\boldsymbol{\epsilon},\mathbf{d},\mathbf{c}) \quad \text{s.t.} \quad \begin{cases} \mathbf{x}_t = \mathcal{P}(\mathbf{x}_{t+1},\mathbf{d}_{t+1}) + \boldsymbol{\epsilon}_{t+1}, & 0 \leq t \leq T-1, \\ \mathbf{x}_T = \mathbf{D}\mathbf{c}, \end{cases} \tag{4.1}$$

where

$$\mathcal{J}(\mathbf{x},\boldsymbol{\epsilon},\mathbf{d},\mathbf{c}) \triangleq \sum_{t=0}^{T} \|\mathcal{H}(\mathbf{x}_t) - \mathbf{y}_t\|_2^2 + \alpha_1 \sum_{t=1}^{T} \|\boldsymbol{\epsilon}_t\|_p^p + \alpha_2 \sum_{t=1}^{T} \mathcal{R}(\mathbf{G}^*\mathbf{d}_t) + \alpha_3 \|\mathbf{c}\|_p^p,$$

for some $\alpha_j \geq 0$, $j \in \{1,2,3\}$, and $p \geq 0$. Let us make a few comments about (4.1). The first constraint ensures that the images of the HR sequence verify the dynamical model (3.2); the second enforces that prior model (3.4) is satisfied. Each term in the cost function $\mathcal{J}(\mathbf{x},\boldsymbol{\epsilon},\mathbf{d},\mathbf{c})$ has a clear physical meaning: the first term penalizes the discrepancies between the predicted and the received observations; the second penalizes the noise on the dynamical model; the third enforces the displacement to have some regularity and the last one constrains $\mathbf{c}$ to have some desirable properties (depending on the choice of p). For example, setting $p \in [0,1]$ typically promotes the sparsity of $\mathbf{c}$, see [25].

Problem (4.1) involves a huge number of unknowns (namely $(4T+1)n + q$ variables if $\mathbf{x},\boldsymbol{\epsilon},\mathbf{d},\mathbf{c}$ have to be estimated). Hence, solving (4.1) may be critical even for reasonable problem sizes: for instance, considering images of $n = 2^8 \times 2^8$ pixels, a non-redundant dictionary, *i.e.*, $q = n$ and a sequence length $T = 2^4$, we have that the number of variables involved in the optimization problem grows up to roughly 2^{22}. Clearly, such a high-dimensional problem can only be addressed by specifically-dedicated procedures. This is the purpose of the rest of this section. Our approach is based on three main building blocks, presented in sections 4.1, 4.2 and 4.3.

First, in section 4.1, we consider the case where all the functions involved in (4.1) are differentiable. In such a case, we show that the gradient of the cost function associated to an (equivalent) unconstrained version of (4.1) can be evaluated efficiently by resorting to optimal control techniques [6]. More specifically, we emphasize that the complexity related to the evaluation of this gradient remains *linear* in the problem dimensions for many setups of practical interest. As a side remark, let us note that when $p = 2$ and $\mathbf{d}$ is known[3], (4.1) is equivalent to the MAP problem tackled by the Kalman smoother, and considered in several contributions of the literature [22, 20, 19, 48]. Unlike these contributions, which ground their reasoning on the standard Kalman updates, we show that the proposed methodology leads to the exact resolution of (4.1) with a tractable complexity and without resorting to any approximation.

[3]That is $\mathbf{d}$ is no longer considered as an optimization variable but set to a fixed value.

In section 4.2, we consider the case where $\mathbf{d}$ is known but $p = 1$. The corresponding optimization problem is then convex but no longer differentiable. Building on our derivations in section 4.1, we emphasize that this type of problem can be nicely addressed by resorting to the so-called "Alternating Direction of Multipliers Method" (ADMM) [7], a modern optimization technique proposed to handle large-scale non-differentiable optimization problems. More specifically, we show that the particularization of ADMM to the video SR problem involves the iterative application of the low-complexity Kalman smoother derived in section 4.1 and some simple thresholding operations.

Finally, in section 4.3, we consider the general problem (4.1), where $\mathbf{x}, \boldsymbol{\epsilon}, \mathbf{d}, \mathbf{c}$ have to be estimated and $\mathcal{J}(\mathbf{x}, \boldsymbol{\epsilon}, \mathbf{d}, \mathbf{c})$ may contain non-differentiable terms. In this case, (4.1) is non-convex (because the term $\mathcal{P}(\mathbf{x}_{t+1}, \mathbf{d}_{t+1})$ appearing in the constraints is bi-polynomial) and potentially non-differentiable. In order to address this problem, we resort to an optimization procedure introduced by Attouch *et al.* [2, 3], and particularized in [41] to multi-frame SR. The procedure is iterative and exploits the building blocks derived in sections 4.1 and 4.2. The complexity per iteration is linear in the problem dimensions. Moreover, from the arguments exposed in [41], it can be shown that the proposed procedure is convergent to a critical point of the problem.

4.1. The Differentiable Case. In this section, we assume that all the functions appearing in (4.1) are differentiable[4] and show that an efficient resolution of (4.1) via gradient descent algorithms exists. Our approach is based on fast gradient evaluation techniques as exposed in [6].

In order to present our methodology, we first reformulate (4.1) as an (equivalent) unconstrained problem. Notice that, because of the constraints in problem (4.1), any $\mathbf{x}_t$ can be expressed as a deterministic function of $\mathbf{c}$ and $\{\boldsymbol{\epsilon}_{t'}, \mathbf{d}_{t'}\}_{t'=t+1}^{T}$. In other words, there exists a function $\mathcal{Q}(\boldsymbol{\epsilon}, \mathbf{d}, \mathbf{c}) : \mathbb{R}^{Tn} \times \mathbb{R}^{2Tn} \times \mathbb{R}^{q} \to \mathbb{R}^{(T+1)n}$ such that, given $\boldsymbol{\epsilon}, \mathbf{d}$ and $\mathbf{c}$, $\mathbf{x} = \mathcal{Q}(\boldsymbol{\epsilon}, \mathbf{d}, \mathbf{c})$ is the unique vector satisfying the constraints in (4.1). As a consequence, (4.1) can also be equivalently expressed as

$$\underset{(\boldsymbol{\epsilon}, \mathbf{d}, \mathbf{c})}{\arg\min}\, \mathcal{J}(\boldsymbol{\epsilon}, \mathbf{d}, \mathbf{c}), \tag{4.2}$$

where

$$\begin{aligned} \mathcal{J}(\boldsymbol{\epsilon}, \mathbf{d}, \mathbf{c}) &\triangleq \mathcal{J}(\mathbf{x} = \mathcal{Q}(\boldsymbol{\epsilon}, \mathbf{d}, \mathbf{c}), \boldsymbol{\epsilon}, \mathbf{d}, \mathbf{c}), \\ &= \sum_{t=0}^{T} \|\mathcal{H}(\mathcal{Q}_t(\boldsymbol{\epsilon}, \mathbf{d}, \mathbf{c})) - \mathbf{y}_t\|_2^2 + \alpha_1 \sum_{t=1}^{T} \|\boldsymbol{\epsilon}_t\|_p^p \\ &\quad + \alpha_2 \sum_{t=1}^{T} \mathcal{R}(\mathbf{G}^* \mathbf{d}_t) + \alpha_3 \|\mathbf{c}\|_p^p, \end{aligned} \tag{4.3}$$

and $\mathcal{Q}_t(\boldsymbol{\epsilon}, \mathbf{d}, \mathbf{c})$ is the restriction of $\mathcal{Q}(\boldsymbol{\epsilon}, \mathbf{d}, \mathbf{c})$ to $\mathbf{x}_t$.

(4.2) is a (smooth) *unconstrained* minimization problem and can thus be solved by any procedure belonging to the family of gradient descent algorithms. At this point, let us make two remarks: *i)* $\mathcal{J}(\boldsymbol{\epsilon}, \mathbf{d}, \mathbf{c})$ has usually an intricate structure and its gradient does therefore not have any simple analytical expression; *ii)* the computation of the gradient of $\mathcal{J}(\boldsymbol{\epsilon}, \mathbf{d}, \mathbf{c})$ via finite differences is out of reach for the considered

[4] This imposes in particular that $1 < p < \infty$.

problem because it would require to evaluate the cost function twice as many times as the (huge!) number of variables.

As a consequence, the main bottleneck for solving (4.2) lies in the tractable evaluation of the gradient of $\mathcal{J}(\boldsymbol{\epsilon}, \mathbf{d}, \mathbf{c})$. We emphasize in appendix A that the particular structure of $\mathcal{J}(\boldsymbol{\epsilon}, \mathbf{d}, \mathbf{c})$ enables the use of a specific methodology with a complexity scaling linearly with the problem dimensions. More specifically, let

$$\begin{cases} \mathcal{G}_0(\mathbf{x}_0) \triangleq \|\mathcal{H}(\mathbf{x}_0) - \mathbf{y}_0\|_2^2, \\ \mathcal{G}_t(\mathbf{x}_t, \boldsymbol{\epsilon}_t, \mathbf{d}_t) \triangleq \|\mathcal{H}(\mathbf{x}_t) - \mathbf{y}_t\|_2^2 + \alpha_1 \|\boldsymbol{\epsilon}_t\|_p^p + \alpha_2 \mathcal{R}(\mathbf{G}^* \mathbf{d}_t), \quad \text{for } 1 \leq t \leq T-1, \\ \mathcal{G}_T(\mathbf{x}_T, \boldsymbol{\epsilon}_T, \mathbf{d}_T, \mathbf{c}) \triangleq \|\mathcal{H}(\mathbf{x}_T) - \mathbf{y}_T\|_2^2 + \alpha_1 \|\boldsymbol{\epsilon}_T\|_p^p + \alpha_2 \mathcal{R}(\mathbf{G}^* \mathbf{d}_T) + \alpha_3 \|\mathbf{c}\|_p^p. \end{cases} \tag{4.4}$$

Using the notation $\mathcal{G}_T(\boldsymbol{\epsilon}_T, \mathbf{d}_T, \mathbf{c}) \triangleq \mathcal{G}_T(\mathbf{x}_T = \mathbf{D}\mathbf{c}, \boldsymbol{\epsilon}_T, \mathbf{d}_T, \mathbf{c})$, the elements of the gradient of $\mathcal{J}(\boldsymbol{\epsilon}, \mathbf{d}, \mathbf{c})$ at $(\boldsymbol{\epsilon}', \mathbf{d}', \mathbf{c}')$ can then be evaluated as follows:

$$\begin{cases} \nabla_{\mathbf{d}_t} \mathcal{J}(\boldsymbol{\epsilon}', \mathbf{d}', \mathbf{c}') = \nabla_{\mathbf{d}_t} \mathcal{P}^*(\mathbf{x}_t', \mathbf{d}_t') \boldsymbol{\zeta}_{t-1} + \nabla_{\mathbf{d}_t} \mathcal{G}_t(\mathbf{x}_t', \boldsymbol{\epsilon}_t', \mathbf{d}_t'), \\ \nabla_{\boldsymbol{\epsilon}_t} \mathcal{J}(\boldsymbol{\epsilon}', \mathbf{d}', \mathbf{c}') = \boldsymbol{\zeta}_{t-1} + \nabla_{\boldsymbol{\epsilon}_t} \mathcal{G}_t(\mathbf{x}_t', \boldsymbol{\epsilon}_t', \mathbf{d}_t'), \\ \nabla_{\mathbf{c}} \mathcal{J}(\boldsymbol{\epsilon}', \mathbf{d}', \mathbf{c}') = \mathbf{D}^* \boldsymbol{\zeta}_T + \nabla_{\mathbf{c}} \mathcal{G}_T(\boldsymbol{\epsilon}_T', \mathbf{d}_T', \mathbf{c}'), \end{cases} \tag{4.5}$$

where the variables $\mathbf{x}_t'$, $\boldsymbol{\epsilon}_t', \mathbf{d}_t'$ and $\mathbf{c}'$ must satisfy the constraints of problem (4.1), that is

$$\begin{cases} \mathbf{x}_T' = \mathbf{D}\mathbf{c}', \\ \mathbf{x}_t' = \mathcal{P}(\mathbf{x}_{t+1}', \mathbf{d}_{t+1}') + \boldsymbol{\epsilon}_{t+1}', \quad t = T-1, ..., 0, \end{cases} \tag{4.6}$$

and the sequence of "adjoint" variables $\{\boldsymbol{\zeta}_t\}_{t=0}^T$ obeys the following recursion:

$$\begin{cases} \boldsymbol{\zeta}_0 = \nabla \mathcal{G}_0(\mathbf{x}_0'), \\ \boldsymbol{\zeta}_t = \nabla_{\mathbf{x}_t} \mathcal{P}^*(\mathbf{x}_t', \mathbf{d}_t') \boldsymbol{\zeta}_{t-1} + \nabla_{\mathbf{x}_t} \mathcal{G}_t(\mathbf{x}_t', \boldsymbol{\epsilon}_t', \mathbf{d}_t'), \quad t = 1, ..., T-1, \\ \boldsymbol{\zeta}_T = \nabla_{\mathbf{x}_T} \mathcal{P}^*(\mathbf{x}_T', \mathbf{d}_T') \boldsymbol{\zeta}_{T-1} + \nabla_{\mathbf{x}_T} \mathcal{G}_T(\mathbf{x}_T', \boldsymbol{\epsilon}_T', \mathbf{d}_T', \mathbf{c}'). \end{cases} \tag{4.7}$$

Expressions in (4.5) together with recursions (4.6) and (4.7) provide an efficient way to evaluate the gradient of $\mathcal{J}(\boldsymbol{\epsilon}, \mathbf{d}, \mathbf{c})$. The overall methodology can be understood as a 3-step procedure: *i)* given some values of $\boldsymbol{\epsilon}', \mathbf{d}'$ and $\mathbf{c}'$, evaluate $\{\mathbf{x}_t'\}_{t=0}^T$ with recursion (4.6); *ii)* use the value of $\{\mathbf{x}_t'\}_{t=0}^T$ to evaluate the adjoint variables $\{\boldsymbol{\zeta}_t\}_{t=0}^T$ from (4.7); *iii)* compute the gradient of $\mathcal{J}(\boldsymbol{\epsilon}, \mathbf{d}, \mathbf{c})$ by using (4.5). Note that the gradients appearing in the right-hand side of (4.5) and (4.7) typically have simple analytical expressions and are thus straightforward to evaluate.

It is easy to see that the complexity induced by this methodology scales (at worst) as $\mathcal{O}(n^2 T + nq)$ since it only involves matrix-vector multiplications, with matrices of dimension $n \times n$ or $n \times q$. In practice, this complexity can usually be reduced to $\mathcal{O}(nT + q)$, or simply to $\mathcal{O}(nT)$ in the case of a non-redundant dictionary. This linearity in the problem dimensions occurs if the matrices involved in (3.4), (4.5) and (4.7) are typically very sparse and/or they involve fast transforms of linear complexity[5]. In the (typical) example (3.3) considered in this paper, we clearly obtain this linear complexity since $|\mathcal{V}(\chi(s) + \mathbf{d}_t(s))| \ll n$. In the rest of the paper, we focus on model

[5]This is the case for any non-redundant wavelet basis, which will induce an overall complexity of $\mathcal{O}(nT)$. Fast transforms for sparse redundant dictionaries such as curvelets frames also exist but imply a slight complexity overload since the matrix-vector multiplication scales in this case as $\mathcal{O}(n \log n)$, yielding an overall complexity of $\mathcal{O}(n(T + \log n))$.

(3.3) and choose a dictionary so that the complexity related to (4.5)-(4.7) is linear in the problem dimensions.

For now, let us compare this complexity to the one achieved by a Kalman smoother applied to a particular instance of (4.1). More specifically, letting $p = 2$ and assuming $\mathbf{d}$ to be known, (4.1) corresponds to the MAP estimation problem associated to the following probabilistic (backward) state-evolution model:

$$\left\{ \begin{array}{ll} \mathbf{x}_T & \sim \mathcal{N}(\mathbf{0}_n, \alpha_3^{-1}\mathbf{D}\mathbf{D}^*) \\ \mathbf{x}_t & \sim \mathcal{N}(\mathcal{P}(\mathbf{x}_{t+1}, \mathbf{d}_{t+1}), \alpha_1^{-1}\mathbf{I}_n) \\ \mathbf{y}_t & \sim \mathcal{N}(\mathcal{H}(\mathbf{x}_t), \mathbf{I}_m), \end{array} \right. \tag{4.8}$$

where $\mathbf{v} \sim \mathcal{N}(\mathbf{m}, \Gamma)$ indicates that $\mathbf{v}$ is distributed according to a multivariate normal distribution with mean $\mathbf{m}$ and covariance Γ.

Since $\mathcal{J}(\boldsymbol{\epsilon}, \mathbf{d}, \mathbf{c})$ is differentiable, we can use our methodology to compute its gradient with respect to $\boldsymbol{\epsilon}$ and $\mathbf{c}$ (using the two last rows of (4.5)) and implement a descent algorithm. Note that $\mathcal{J}(\boldsymbol{\epsilon}, \mathbf{d}, \mathbf{c})$ being strictly convex in $(\boldsymbol{\epsilon}, \mathbf{c})$ when $p = 2$, this type of algorithm is ensured to converge to the global minimum of the problem. On the other hand, for such a model, it is well-known that the Kalman smoother can compute exactly the solution of (4.1) in a finite number of steps, namely one forward and one backward recursions, see *e.g.*, [33, Chapter 20]. The Kalman smoother involves the update of a length-n mean vector and an $n \times n$ covariance matrix at each step of the two recursions; moreover, the evaluation of these quantities requires the inversion of a $n \times n$ matrix. Hence, the Kalman smoother exhibits a computational complexity scaling as[6] $\mathcal{O}(n^3 T)$. This is in contrast with the complexity of the proposed method which scales as $\mathcal{O}(nT)$ per iteration. Hence, if the descent algorithm has converged (close) to the minimum after a reasonable number of iterations, the proposed methodology drastically reduces the complexity necessary to obtain the desired solution.

4.2. The Convex Non-differentiable Case. We saw in the previous section that setting $p = 2$ in (4.1) is, from a Bayesian point of view, equivalent to imposing a Gaussian prior on the unknowns. In practice, such a prior is often badly-suited to characterize the "physical" reality. For example, the noise $\boldsymbol{\epsilon}_t$ affecting the image transition model (3.2) is often due to apparition or occlusion phenomena. The latter are characterized by a set of large spatially-localized errors, and thus poorly modeled by a Gaussian distribution. On the other hand, it has recently been shown that penalizing an unknown vector by an ℓ_1 norm promotes its sparsity, *i.e.,* encourages a few nonzero elements of arbitrary amplitudes. In the same way, several recent contributions have emphasized that sparse decomposition in a proper redundant dictionary is a good synthesis model for natural images, see [39, 21]. This tends to suggest that the sparsity of $\mathbf{c}$ should be enforced in the sought solution.

In this section, we will therefore assume that both $\boldsymbol{\epsilon}$ and $\mathbf{c}$ are penalized by an ℓ_1 norm, that is $p = 1$. We note that the ℓ_1 norm is convex but non-differentiable; this imposes an additional difficulty to the resolution of (4.1). For now, we will suppose that $\mathbf{d}$ is known; the general problem where $\mathbf{c}$, $\boldsymbol{\epsilon}$ and $\mathbf{d}$ have to be estimated is postponed to the next section. Particularizing (4.1) to these working hypotheses, we obtain the following convex but non-differentiable problem:

[6] We note that the complexity can be reduced to $\mathcal{O}(T(m^3 + mn))$ by using some computational tricks as the well-known Woodbury matrix identity, see *e.g.*, [33, Lemma 4.1]. However, the latter still remain too costly for typical problem sizes.

$$\underset{(\mathbf{x},\boldsymbol{\epsilon},\mathbf{c})}{\arg\min}\, \mathcal{J}(\mathbf{x},\boldsymbol{\epsilon},\mathbf{c}) \quad \text{s.t.} \left\{ \begin{array}{l} \mathbf{x}_t = \mathcal{P}(\mathbf{x}_{t+1},\mathbf{d}_{t+1}) + \boldsymbol{\epsilon}_{t+1}, \quad 0 \leq t \leq T-1 \\ \mathbf{x}_T = \mathbf{D}\mathbf{c}, \end{array} \right. \tag{4.9}$$

where

$$\mathcal{J}(\mathbf{x},\boldsymbol{\epsilon},\mathbf{c}) \triangleq \sum_{t=0}^{T} \|\mathcal{H}(\mathbf{x}_t) - \mathbf{y}_t\|_2^2 + \alpha_1 \sum_{t=1}^{T} \|\boldsymbol{\epsilon}_t\|_1 + \alpha_3 \|\mathbf{c}\|_1.$$

As previously, the main bottleneck for the resolution of this problem lies in its high dimensionality. This, in turn, imposes to resort to low-complexity optimization procedures. We show hereafter, that a complexity scaling linearly with the problem dimensions is possible by means of the "Alternating Direction Method of Multipliers" (ADMM). ADMM has recently emerged in the optimization community to address large-scale optimization problems. Among the particular assets of this type of method, let us mention: *i)* its robustness (the convergence to a global minimum is ensured under very mild conditions); *ii)* its rapid convergence to an acceptable accuracy (typically a few tens of iterations is sufficient). We refer the reader to appendix B for a short description of the ADMM framework.

In order to derive the ADMM recursions, we first need to reformulate (4.9) in the standard form (B.1) in appendix B. Letting

$$\Omega \triangleq \left\{ (\mathbf{x},\boldsymbol{\epsilon},\mathbf{c}) \left| \begin{array}{l} \mathbf{x}_t = \mathcal{P}(\mathbf{x}_{t+1},\mathbf{d}_{t+1}) + \boldsymbol{\epsilon}_{t+1}, \quad 0 \leq t \leq T-1 \\ \mathbf{x}_T = \mathbf{D}\mathbf{c} \end{array} \right. \right\}, \tag{4.10}$$

(4.9) can be reexpressed as:

$$\begin{array}{c} \underset{\substack{(\mathbf{x},\boldsymbol{\epsilon},\mathbf{c})\in\Omega \\ (\tilde{\boldsymbol{\epsilon}},\tilde{\mathbf{c}})\in\mathbb{R}^{nT+q}}}{\arg\min} \quad \displaystyle\sum_{t=0}^{T} \|\mathcal{H}(\mathbf{x}_t) - \mathbf{y}_t\|_2^2 + \alpha_1 \sum_{t=1}^{T} \|\tilde{\boldsymbol{\epsilon}}_t\|_1 + \alpha_3 \|\tilde{\mathbf{c}}\|_1, \\ \text{s.t.} \left\{ \begin{array}{l} \boldsymbol{\epsilon} = \tilde{\boldsymbol{\epsilon}}, \\ \mathbf{c} = \tilde{\mathbf{c}}. \end{array} \right. \end{array}$$

Here, we have added two new variables to the problem, $\tilde{\boldsymbol{\epsilon}}$ and $\tilde{\mathbf{c}}$, which are counterbalanced by the inclusion of two new constraints. Using the formalism exposed in appendix B with $\boldsymbol{z}_1 = (\mathbf{x},\boldsymbol{\epsilon},\mathbf{c})$, $\boldsymbol{z}_2 = (\tilde{\boldsymbol{\epsilon}},\tilde{\mathbf{c}})$, $\Xi_1 = \Omega$ and $\Xi_2 = \mathbb{R}^{nT+q}$, we obtain the following ADMM recursions:

$$(\mathbf{x}^{(k+1)},\boldsymbol{\epsilon}^{(k+1)},\mathbf{c}^{(k+1)}) = \underset{\mathbf{x},\boldsymbol{\epsilon},\mathbf{c}}{\arg\min}\, \mathcal{L}^{(k)}(\mathbf{x},\boldsymbol{\epsilon},\mathbf{c}) \tag{4.11}$$

$$\text{s.t.} \left\{ \begin{array}{l} \mathbf{x}_t = \mathcal{P}(\mathbf{x}_{t+1},\mathbf{d}_{t+1}) + \boldsymbol{\epsilon}_{t+1}, \quad 0 \leq t \leq T-1 \\ \mathbf{x}_T = \mathbf{D}\mathbf{c}, \end{array} \right.$$

$$\left\{ \begin{array}{rcl} \tilde{\boldsymbol{\epsilon}}_t^{(k+1)} & = & \arg\min_{\tilde{\boldsymbol{\epsilon}}_t} \ \|\tilde{\boldsymbol{\epsilon}}_t\|_1 + \frac{\rho}{2\alpha_1}\|\boldsymbol{\epsilon}_t^{(k+1)} - \tilde{\boldsymbol{\epsilon}}_t + \mathbf{u}_{\boldsymbol{\epsilon}_t}^{(k)}\|_2^2, \\ \tilde{\mathbf{c}}^{(k+1)} & = & \arg\min_{\tilde{\mathbf{c}}} \quad \|\tilde{\mathbf{c}}\|_1 + \frac{\rho}{2\alpha_3}\|\mathbf{c}^{(k+1)} - \tilde{\mathbf{c}} + \mathbf{u}_{\mathbf{c}}^{(k)}\|_2^2, \end{array} \right. \tag{4.12}$$

$$\left\{ \begin{array}{rcl} \mathbf{u}_{\boldsymbol{\epsilon}_t}^{(k+1)} & = & \mathbf{u}_{\boldsymbol{\epsilon}_t}^{(k)} + \boldsymbol{\epsilon}_t^{(k+1)} - \tilde{\boldsymbol{\epsilon}}_t^{(k+1)}, \\ \mathbf{u}_{\mathbf{c}}^{(k+1)} & = & \mathbf{u}_{\mathbf{c}}^{(k)} + \mathbf{c}^{(k+1)} - \tilde{\mathbf{c}}^{(k+1)}, \end{array} \right. \tag{4.13}$$

where $\rho > 0$ and we have introduced the function:

$$\mathcal{L}^{(k)}(\mathbf{x}, \boldsymbol{\epsilon}, \mathbf{c}) = \sum_{t=0}^{T} \|\mathcal{H}(\mathbf{x}_t) - \mathbf{y}_t\|_2^2 + \frac{\rho}{2}\|\boldsymbol{\epsilon} - \tilde{\boldsymbol{\epsilon}}^{(k)} + \mathbf{u}_{\boldsymbol{\epsilon}}^{(k)}\|_2^2 + \frac{\rho}{2}\|\mathbf{c} - \tilde{\mathbf{c}}^{(k)} + \mathbf{u}_{\mathbf{c}}^{(k)}\|_2^2. \tag{4.14}$$

Equations (4.11), (4.12) and (4.13) correspond respectively to expressions (B.2), (B.3) and (B.4) in appendix B.

Let us make the following remarks about the different steps of the ADMM recursion. First, problem (4.11) has the same structural form as the problem addressed in section 4.1; in particular, all the terms of the cost function appearing in (4.11) are differentiable while the set of constraints imposed on $\mathbf{x}, \boldsymbol{\epsilon}, \mathbf{d}$ and $\mathbf{c}$ is strictly the same. We can thus apply the methodology described in section 4.1 to solve this problem via a gradient descent algorithm, with a complexity per iteration scaling as $\mathcal{O}(nT)$. Interestingly, let us mention that, under very mild conditions, the convergence of ADMM is still ensured if the minimizations in (4.11)-(4.12) are not performed exactly, see *e.g.*, [18, Theorem 8]. This suggests that the number of gradient steps carried out to search for the minimum of (4.11) can be rather limited without affecting the convergence of the overall ADMM process.

Second, the optimization problems specified in (4.12) have a very simple analytical solution. In fact the right-hand sides of (4.12) correspond to the definition of the proximal operator of the ℓ_1 norm. The latter has been extensively studied in the literature (see *e.g.*, [36, section 6.5.2]) and possesses a simple analytical solution based on soft-thresholding operators. In particular, we have

$$\left\{ \begin{array}{ll} \tilde{\boldsymbol{\epsilon}}_t^{(k+1)}(i) & = \mathrm{soft}_{\frac{\alpha_1}{\rho}}\left(\boldsymbol{\epsilon}_t^{(k+1)}(i) + \mathbf{u}_{\boldsymbol{\epsilon}_t}^{(k)}(i)\right) \\ \tilde{\mathbf{c}}^{(k+1)}(i) & = \mathrm{soft}_{\frac{\alpha_3}{\rho}}\left(\mathbf{c}^{(k+1)}(i) + \mathbf{u}_{\mathbf{c}}^{(k)}(i)\right) \end{array} \right. \qquad \forall i, \tag{4.15}$$

where

$$\mathrm{soft}_{\lambda}(a) = \left\{ \begin{array}{ll} a - \lambda & \text{if } a \geq \lambda, \\ a + \lambda & \text{if } a \leq -\lambda, \\ 0 & \text{otherwise.} \end{array} \right. \tag{4.16}$$

We note that, the solution of (4.12) is typically sparse since the soft-thresholding operator (4.16) enforces the small coefficients to be equal to zero. Moreover, we see from (4.15) that the complexity of this ADMM step clearly scales as $\mathcal{O}(nT + q)$.

As a conclusion, since the last step (4.13) of the procedure only involves vector additions, the particularization of ADMM to our problem leads to an algorithm exhibiting a complexity per iteration scaling linearly in the problem dimensions.

4.3. The General Case. Let us now concentrate our attention on the general problem (4.1) where $p = 1$ and all the variables $\mathbf{x}$, $\mathbf{c}$, $\boldsymbol{\epsilon}$, $\mathbf{d}$ have to be estimated. The cost function then contains both non-differentiable and non-convex terms. In such a case, ensuring the convergence to a global minimum is usually out of reach for any deterministic optimization procedure. In the sequel, we will consider an optimization method proposed in [2, 3] and particularized to multi-frame super-resolution problems in [41]. This procedure addresses optimization problems involving a cost function satisfying the so-called "Kurdyca-Lojasiewicz" property and is guaranteed

to convergence to a critical point of the latter under mild conditions. We refer the reader to [2, 3] for more details on "Kurdyca-Lojasiewicz" functions. Here, we just mention that functions made up of the composition of piecewise polynomial functions obey the "Kurdyca-Lojasiewicz" property. Scrutinizing the structure of (4.1) and taking (3.3) into account, it is easy to see that our cost function is piecewise polynomial; the optimization framework developed in [2, 3] therefore applies.

Our methodology obeys a 2-step recursion which follows the same lines as the procedure presented in [41]. To express these recursions, we focus on the unconstrained formulation (4.2) of our general optimization problem (4.1). The first step of the procedure solves the following problem:

$$(\boldsymbol{\epsilon}^{(k+1)}, \mathbf{c}^{(k+1)}) = \underset{(\boldsymbol{\epsilon}, \mathbf{c})}{\arg\min}\, \mathcal{J}(\boldsymbol{\epsilon}, \mathbf{d}^{(k)}, \mathbf{c}) + \gamma\, \mathcal{C}(\boldsymbol{\epsilon} - \boldsymbol{\epsilon}^{(k)}, \mathbf{c} - \mathbf{c}^{(k)}), \tag{4.17}$$

where $\gamma > 0$, $\mathcal{J}$ is the cost function in (4.3) and $\mathcal{C} : \mathbb{R}^{nT} \times \mathbb{R}^q \to \mathbb{R}_+$ is a nonnegative proper lower-semicontinuous convex function such that $\mathcal{C}(\mathbf{0}_{nT}, \mathbf{0}_q) = 0$. It thus consists in minimizing the (penalized) cost function $\mathcal{J}(\boldsymbol{\epsilon}, \mathbf{d}, \mathbf{c})$ over the subset of variables $(\boldsymbol{\epsilon}, \mathbf{c})$; the penalizing term $\mathcal{C}$ plays the role of a "cost-to-move" function which prevents the new iterate $(\boldsymbol{\epsilon}^{(k+1)}, \mathbf{c}^{(k+1)})$ from differing too much from the previous one. In the sequel, we will focus our attention on the following penalizing term[7]:

$$\mathcal{C}(\boldsymbol{\epsilon} - \boldsymbol{\epsilon}^{(k)}, \mathbf{c} - \mathbf{c}^{(k)}) = \sum_{t=1}^{T} \|\mathbf{H}^*(\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_t^{(k)})\|_1 + \|\mathbf{c} - \mathbf{c}^{(k)}\|_1, \tag{4.18}$$

where $\mathbf{H} \in \mathbb{R}^{n \times n}$ is a wavelet basis. The operational meaning of this "cost-to-move" function is as follows: the ℓ_1 norm enforces its argument to be sparse; hence, the second term in (4.18) ensures that the number of nonzero coefficients in $\mathbf{c}^{(k+1)}$ does not differ too much from the one in $\mathbf{c}^{(k)}$, while the first term plays the same role for the wavelet coefficients of $\boldsymbol{\epsilon}_t^{(k+1)} - \boldsymbol{\epsilon}_t^{(k)}$. Using this type of "cost-to-move" is not mandatory for the convergence of the proposed procedure. However, it has been shown empirically in [41] that it is well-suited to avoid some undesirable local minima of the cost function.

In the second step of the recursion, we update the velocity field $\mathbf{d}$ as:

$$\mathbf{d}^{(k+1)} = \underset{\mathbf{d}}{\arg\min}\, \mathcal{B}(\mathbf{d}, \mathbf{d}^{(k)}) + \alpha_2 \sum_{t=1}^{T} \mathcal{R}(\mathbf{G}^* \mathbf{d}_t), \tag{4.19}$$

where $\mathcal{B}(\mathbf{d}, \mathbf{d}^{(k)})$ is a quadratic approximation[8] of $\mathcal{B}(\mathbf{d}) \triangleq \sum_{t=0}^{T} \|\mathcal{H}(\mathcal{Q}_t(\boldsymbol{\epsilon}^{(k+1)}, \mathbf{d}, \mathbf{c}^{(k+1)})) - \mathbf{y}_t\|_2^2$, that is

$$\mathcal{B}(\mathbf{d}, \mathbf{d}^{(k)}) \triangleq \mathcal{B}(\mathbf{d}^{(k)}) + \nabla_{\mathbf{d}} \mathcal{B}^*(\mathbf{d}^{(k)})(\mathbf{d} - \mathbf{d}^{(k)}) + \frac{\alpha^{(k)}}{2} \|\mathbf{d} - \mathbf{d}^{(k)}\|_2^2, \quad \alpha^{(k)} > 0. \tag{4.20}$$

The choice of $\alpha^{(k)}$ is of course not arbitrary and should be made so that the convergence of the procedure is ensured. We elaborate on this point further in this section. For now, let us first discuss the practical implementation and complexity of recursions

[7] In theory, the ℓ_1-norm should be substituted by a smooth approximation to prove convergence towards a critical point of the cost function, as done in [41]. In practice, we note that this substitution does not impact convergence.

[8] Note that $\mathcal{B}(\mathbf{d})$ is similar to the first term of the cost function in (4.3).

(4.17)-(4.19). It should be noticed that the building blocks presented in sections 4.1 and 4.2 can be exploited to solve efficiently these steps. Indeed, problem (4.17) has the same structural form as the one considered in (4.9): the cost function consists in a quadratic term plus a set of convex but non-differentiable terms. We can thus use the ADMM procedure described in section 4.2 to address it. In the same way, we see from definition (4.20) that the cost function (4.19) is made up of a quadratic term plus some non-differentiable function $\alpha_2 \sum_{t=1}^{T} \mathcal{R}(\mathbf{G}^* \mathbf{d}_t)$. Hence, the ADMM procedure described in section 4.2 can also be applied here to solve (4.19). In comparison to our exposition in section 4.2, only the proximal operators of the non-differentiable terms will change when ADMM is applied to (4.17) and (4.19). In particular, the computation of any gradient of the differentiable part of the cost functions can be efficiently evaluated via the procedure described in section 4.1. We particularize the expression of the proximal operators appearing in the ADMM implementation of (4.17)-(4.19) in appendix C.2. As previously, it turns out that the implementation of the latter only requires a linear complexity. The complexity of each iteration of (4.17), (4.19) is thus once again linear.

To conclude this section, let us discuss the convergence of the proposed procedure. In [41, Theorem 1], the authors proved that if $\mathcal{J}(\boldsymbol{\epsilon}, \mathbf{d}, \mathbf{c})$ satisfies the "Kurdyca-Lojasiewicz" property and the $\alpha^{(k)}$'s are properly selected, the sequence defined in (4.17)-(4.19) is either unbounded or converges to a critical point of $\mathcal{J}(\boldsymbol{\epsilon}, \mathbf{d}, \mathbf{c})$. A procedure to properly select factors $\alpha^{(k)}$ is exposed in [41, section 2.3] and is easy to implement in practice. Particularized to the setup considered in this paper, this procedure reads as follows: select $\alpha^{(k)} = 2^i \xi$ with $\xi > 0$ and with i the smallest positive integer such that

$$\begin{aligned} \mathcal{B}(\mathbf{d}^{(k+1)}) - \mathcal{B}(\mathbf{d}^{(k)}) \leq & \frac{(2^i - 1)\xi}{2} \|\mathbf{d}^{(k+1)} - \mathbf{d}^{(k)}\|^2 + \nabla_{\mathbf{d}} \mathcal{B}^*(\mathbf{d}^{(k)})(\mathbf{d}^{(k+1)} - \mathbf{d}^{(k)}) \\ & + \alpha_2 \sum_{t=1}^{T} \left(\mathcal{R}(\mathbf{G}^* \mathbf{d}_t^{(k)}) - \mathcal{R}(\mathbf{G}^* \mathbf{d}_t^{(k+1)}) \right). \end{aligned} \tag{4.21}$$

As mentioned at the beginning of the section, the cost function $\mathcal{J}(\boldsymbol{\epsilon}, \mathbf{d}, \mathbf{c})$ is piecewise polynomial and therefore satisfies the "Kurdyca-Lojasiewicz" property. Hence, the sequence defined by (4.17), (4.19) with factor selection (4.21) is either unbounded or converges to a critical point of $\mathcal{J}(\boldsymbol{\epsilon}, \mathbf{d}, \mathbf{c})$. Finally, let us note that the boundedness of $\{(\boldsymbol{\epsilon}^{(k)}, \mathbf{d}^{(k)}, \mathbf{c}^{(k)})\}_k$ is usually observed in practice or is easy to enforce by adding box constraints to the optimization problem.

5. Experiments. In this section, we provide an experimental validation of the proposed SR procedures. We focus on the problem of recovering a sequence of HR natural images from blurry and LR observations. In section 5.1, we provide a precise definition of the model parameters used to run our algorithms. In section 5.2, we describe four particularizations of the procedures described in section 4, which will be assessed in our simulations. In sections 5.3 and 5.4, we respectively describe the database and the figures of merit used in our experiments. Finally, a discussion of the performance of the proposed SR methodologies is provided in section 5.5.

5.1. Specification of the Model Parameters. In this section, we discuss the choice of the parameters appearing in our model in section 3. In particular, we specify the definitions of $\mathcal{H}$, $\mathcal{P}$, $\mathbf{D}$, $\mathbf{G}$ and $\mathcal{R}$.

The observation model $\mathcal{H}$ is defined as the composition of a low-pass filtering and down-sampling operation. The low-pass filter is assumed to model the blurring effect induced by the camera transfer function. In our simulations, we use an approximation of a Gaussian kernel of standard deviation equal to 1.12, as proposed in [9]. Down-sampling factors equal to 2 and 4 are considered.

The operator $\mathcal{P}$ is supposed to model a "displaced frame difference" (DFD): $\mathcal{P}$ is thus defined as in (3.3) with the interpolation functions $\{\psi_i\}_{i=1}^n$ equal to bi-dimensional cubic cardinal splines [47]. This representation offers a reasonable accuracy with a complexity scaling linearly with the image dimension, see appendix C.1 for further details.

The dictionary $\mathbf{D}$ is chosen so that natural images have a sparse representation as a combination of a few of its columns. Several choices of such dictionaries have been proposed in the literature, see, *e.g.*, [32]. Hereafter, we consider a dictionary made up of discrete real-valued curvelets [12]; curvelets are known to yield sparse representations of piece-wise smooth functions. The choice of a curvelet dictionary is also motivated by the existence of fast algorithms for the computation of the product between $\mathbf{D}$ and some vector, see [11]: this transform is based on a fast Fourier transform and its complexity[9] scales as $\mathcal{O}(n \log n)$.

To complete our discussion, let us elaborate on the choice of $\mathbf{G}$ and $\mathcal{R}$, characterizing the regularization imposed on the displacement field $\mathbf{d}_t$. In our simulations, we wish to enforce either a global or a piecewise regularity of the motion. We proceed as follows. The spatial derivatives of the motion are approximated by a "finite difference" scheme: each finite difference corresponds to a particular element of the matrix-vector product $\mathbf{G}^*\mathbf{d}_t$ (matrix $\mathbf{G}$ thus contains "± 1" elements located at proper positions). The regularity of the motion field is then enforced by constraining the function $\mathcal{R}(\mathbf{G}^*\mathbf{d}_t)$ to be small. In our experimentations, we choose $\mathcal{R}$ to be defined as in (3.5) with a weighting vector $\mathbf{w}$ as in [50]. Further details are provided in appendix C.1.

5.2. Four Problem Instances. In section 4, we have introduced several optimization procedures dealing with different instances of our general optimization problem (4.1). In particular, we distinguished between the scenarios where $\mathbf{d}$ is known or not, and $p = 1$ or $p = 2$. Each choice leads to a specific algorithm that we summarize below:

- $p = 2$ **and known displacements**: We focus on the unconstrained optimization problem (4.2), which is smooth and convex (as mentioned earlier, the solution of this problem is the same as the one of a Kalman smoother). The gradient of the cost function with respect to $\mathbf{c}$ and $\boldsymbol{\epsilon}$ can be evaluated thanks to recursions (4.6)-(4.7) and by using the last two expressions in (4.5) (Some details of the implementation of (4.5)-(4.7) are provided in appendix C.1, for the particular specification of the model parameters given in section 5.1). The gradient can then be used in a quasi-Newton descent method: in our simulations, we will use a limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) procedure with a line-search routine based on the strong Wolf conditions [35]. We will refer hereafter to this SR procedure as "Algorithm 1".

[9] As mentioned earlier, a linear complexity can be preserved by using, for example, a wavelet basis instead of a curvelet frame.

- $p = 2$ **and unknown displacements**: The problem is similar to the previous one but includes $\mathbf{d}$ as an additional unknown. The target optimization problem (4.2) is then smooth but no longer convex. We use an L-BFGS optimization procedure with Wolf conditions to solve it. The gradient of the cost function is evaluated via (4.5)-(4.7), see appendix C.1 for implementation details. We note that, since the problem is not convex, we are no longer ensured to converge to a global minimum. We refer to this procedure as "Algorithm 2" in the sequel.
- $p = 1$ **and known displacements**: We target problem (4.9), which is non-smooth but convex in $\boldsymbol{\epsilon}$ and $\mathbf{c}$. This problem can be solved by using the ADMM steps (4.11)-(4.13) exposed in section 4.2. We choose $\rho = 1$. We refer to this method as "Algorithm 3" hereafter.
- $p = 1$ **and unknown displacements**: We look for a solution of (4.1) with $p = 1$. The problem is non-smooth and non-convex. We rely on the procedure described in section 4.3 to search for a critical point of the cost function. The matrix $\mathbf{H}$ appearing in the cost-to-move function in (4.18) is chosen to be a Haar wavelet basis[10]. Each step of the recursion (4.17)-(4.19) is solved via an ADMM, as described in appendix C.2. This procedure is referred to as "Algorithm 4" in our experiments.

Convergence of the different ADMM procedures used in Algorithm 3 and 4 was assumed to be reached after 20 iterations. We arbitrarily chose a maximum of 20 iterations for the 2-step recursion used by Algorithm 4.

We notice that, although we have presented the SR procedures for the case of mono-channel image sequences in section 4, their extension to the multi-channel setting (*e.g.*, the usual setting of 3-channel color images) is straightforward and will be considered in our simulations.

5.3. Benchmark dataset. To assess the algorithms, we use a sample of the *MPI Sintel* data set [10]. This recent data set, which is derived from the open source 3D animated short film, was originally constituted for the evaluation of optical flow. The synthesized image sequences are realistic and particularly challenging: on the one hand, displacement fields are characterized by large amplitudes, discontinuities, blur or defocus effects; on the other hand, the image sequence presents many occlusions, specular reflections or atmospheric effects. The first and last images of the "*Bandage 2*" training data set sequence are displayed in Fig. 1. In our simulations, we focus on a region of interest of 436×512 pixels as illustrated in Fig. 2. This figure also represents the motion sequence $\mathbf{d}$ and the associated interpolation errors $\boldsymbol{\epsilon}$. The color images are composed of three spectral bands, each one is coded in 8 bits. We exploit this data set by varying the sequence length from $T = 1$ to 4 and by deteriorating the resolution of observations by applying function $\mathcal{H}$ with a down-sampling factor ranging from 2 to 4.

5.4. Evaluation Procedure. The experimentations deal with the evaluation of the four SR algorithms presented in section 5.2. We also include a comparison

[10] We note that evaluation of products $\mathbf{H}$ or $\mathbf{H}^*$ only requires a linear complexity since they can be implemented by fast wavelet transforms [32, Chapter 7].

with three well-known spatial interpolation techniques, namely, nearest point (block interpolation), bi-cubic spline [47], and Lanczos [46] interpolations.

The performance of the algorithms are assessed in terms of reconstruction of the super-resolved image and estimation of the motion field. We describe the figures of merit used in our assessments hereafter. Let $\{\hat{\mathbf{x}}_t\}_{t=0}^T$ (resp. $\{\hat{\mathbf{d}}_t\}_{t=1}^T$) denote the estimated image sequence (resp. displacements) and $\{\mathbf{x}_t^{true}\}_{t=0}^T$ (resp. $\{\mathbf{d}_t^{true}\}_{t=1}^T$) the corresponding ground truth. Standard criteria [34] to measure the image sequence reconstruction accuracy are the time-averaged Peak-to-Signal Ratio (PSNR):

$$\text{PSNR} = \frac{20}{T+1}\sum_{t=0}^{T}\log_{10}\frac{n\|\mathbf{x}_t^{true}\|_\infty}{\|\mathbf{x}_t^{true}-\hat{\mathbf{x}}_t\|_2},$$

and the time-averaged Correlation Coefficient (CC):

$$\text{CC} = \frac{1}{T+1}\sum_{t=0}^{T}\frac{(\mathbf{x}_t^{true}-\mu_{\mathbf{x}_t^{true}})^*(\hat{\mathbf{x}}_t-\mu_{\hat{\mathbf{x}}_t})}{\|\mathbf{x}_t^{true}-\mu_{\mathbf{x}_t^{true}}\|_2\|\hat{\mathbf{x}}_t-\mu_{\hat{\mathbf{x}}_t}\|_2},$$

where we have denoted the arithmetic mean of vector $\hat{\mathbf{x}}_t$ and $\mathbf{x}_t^{true}$ by $\mu_{\hat{\mathbf{x}}_t}$ and $\mu_{\mathbf{x}_t^{true}}$. We evaluate the accuracy of the estimated motion field with the time-averaged Root Mean Squared Error (RMSE):

$$\text{RMSE} = \frac{1}{T\sqrt{2n}}\sum_{t=1}^{T}\|\mathbf{d}_t^{true}-\hat{\mathbf{d}}_t\|_2,$$

and the time-averaged Mean Barron Angular Error (MBAE) in degrees [5]:

$$\text{MBAE} = \frac{1}{nT}\sum_{t=1}^{T}\sum_{s=1}^{n}\arccos\left(\frac{1+\mathbf{d}_t^{true}(s)\hat{\mathbf{d}}_t(s)+\mathbf{d}_t^{true}(s+n)\hat{\mathbf{d}}_t(s+n)}{\sqrt{(1+\hat{\mathbf{d}}_t(s)^2+\hat{\mathbf{d}}_t(s+n)^2)\,(1+\mathbf{d}_t^{true}(s)^2+\mathbf{d}_t^{true}(s+n)^2)}}\right),$$

where we have adopted the convention that the two n-dimensional components of motion have been arranged one after the other in vectors $\hat{\mathbf{d}}_t$ and $\mathbf{d}_t^{true}$.

In order to compare the results for different window sizes, the four criteria mentioned above (PSNR, CC, RMSE, MBAE) are evaluated for all possible temporal windows of size T in a sequence of 4 images. Averaged criteria are then calculated for each window size.

5.5. Results and Discussion. For the four algorithms, $\mathbf{x}_t$'s are initialized by Lanczos interpolation. Motion fields, when unknown (that is for Algorithms 2 and 4), are initialized by a low-pass filtered version of the ground-truth motion with a very large standard deviation (16 pixels). The parameters of the four algorithms are experimentally tuned in order to satisfy a reasonable tradeoff between visual inspection criteria and error measurements. They are given in table 5.1.

Table 5.2 presents the accuracy of the different algorithms in terms of PSNR and CC for down-sampling factors equal to 2 and 4. Let us analyze these results together with a visual inspection of their SR estimates displayed in Fig. 3, 4 and 6.

First, table 5.2 shows that SR from a single image yields an accuracy way below video SR with a 4-image sequence. Among single-image SR techniques, bi-cubic

	α_1, α_3		α_2		$\gamma, \alpha^{(0)}$	
	SR $\times 4$	SR $\times 2$	SR $\times 4$	SR $\times 2$	SR $\times 4$	SR $\times 2$
Algo. 1	1e-10,1e-5	1e-10,1e-5	·	·	·	·
Algo. 2	1e-10,1e-5	1e-10,1e-5	9e3	1.0	·	·
Algo. 3	1.0, 1.0	1.0, 1.0	·	·	·	·
Algo. 4	1.0, 1.0	1e-10, 1e-3	9e3	1e3	1e2,1e3	1.0,2.0

TABLE 5.1

Parameter setting for the four algorithms and for down-sampling factors equal to 2 and 4 (resp. denoted by SR $\times 2$ and SR $\times 4$ in the table). Parameters α_1, α_2 and α_3 define the cost function given by (4.1) *while parameters γ and $\alpha^{(0)}$ specify the 2 steps* (4.17) *and* (4.19)*. A "·" in the table means that the parameter is irrelevant for the corresponding algorithm.*

		PSNR		CC	
		SR $\times 2$	SR $\times 4$	SR $\times 2$	SR $\times 4$
1 image	nearest point	27.0987	22.7357	0.97804	0.93925
	bi-cubic splines	27.0639	23.3211	0.97775	**0.94162**
	Lanczos	27.0929	**23.3803**	**0.97782**	0.94159
	Algo. 3	**28.0803**	23.3392	0.97762	0.94154
sequence of 4 images	Algo. 1 (known $\mathbf{d}$)	**34.7594**	28.3225	**0.99310**	0.96834
	Algo. 3 (known $\mathbf{d}$)	33.7496	**30.2785**	0.99234	**0.97500**
	Algo. 2	**34.5525**	28.6554	0.98451	**0.95688**
	Algo. 4	33.7097	**28.7960**	**0.98952**	0.95396

TABLE 5.2

Comparison of algorithms accuracy in terms of PSNR and CC for down-sampling factors equal to 2 and 4 (resp. denoted by SR $\times 2$ and SR $\times 4$).

splines, Lanczos interpolation or a sparse prior (Algorithm 3 with a single image) seem to be almost equivalent, although a sparse prior yields a significant increase of the PSNR for a down-sampling factor equal to 2 (see also Fig. 3).

In the case of known displacements with a down-sampling factor equal to 2, Algorithm 1 achieves the best performance. On the other hand, although a noticeable decrease of PSNR can be observed for Algorithm 3, Fig. 4 and 6 (in an larger view) tend to show that this degradation of the performance is not visually prejudicial: both Algorithms 1 and 3 yield very similar results. On the contrary, when the problem becomes severely undetermined (down-sampling factor equal to 4), Algorithm 3 clearly outperforms Algorithm 1 both in terms of PSNR and CC, suggesting the relevance of the sparse prior. Fig. 4 reveals that this quantitative improvement mainly relates to the removal of artifacts, however with the corollary drawback of loosing some image details.

The relevance of sparse priors is further attested when considering the case of unknown displacements. Fig. 4 supports this conclusion, although we observe in table 5.2 similar performances for Algorithm 4 and Algorithm 2 in terms of PSNR and CC. Indeed, a close inspection of Fig. 4 reveals finer details and fewer reconstruction errors for Algorithm 4. This suggests that TV- and ℓ_1-regularizations are preferable to quadratic penalizations when displacements are unknown.

The optimal length T in terms of PSNR and CC is evaluated in table 5.3 for sequences of 2 up to 4 images. This table suggests that, for a down-sampling factor

	PSNR		CC		RMSE		MBAE	
	SR $\times 2$	SR $\times 4$	SR $\times 2$	SR $\times 4$	SR $\times 2$	SR $\times 4$	SR $\times 2$	SR $\times 4$
2 images	33.416	25.516	**0.9904**	0.9484	1.172	1.162	16.18	21.41
3 images	**33.824**	27.334	0.9897	0.9521	1.183	1.184	16.07	21.61
4 images	33.710	**28.796**	0.9895	**0.9540**	**1.102**	**1.155**	**15.76**	**20.76**

TABLE 5.3

Estimation accuracy with Algorithm 4 of HR image (resp. motion) sequence in terms of PSNR and CC (resp. RMSE and MBAE), as a function of the length of the image sequence. $SR \times 2$ and $SR \times 4$ respectively denote the setups with down-sampling factors equal to 2 and 4.

equal to 4, the estimation accuracy of Algorithm 4 increases with the sequence length while, for a down-sampling factor equal to 2, small sequences of 2 or 3 images yield the best results. In Fig. 5, we can notice that the larger the sequence length, the sharper the contours of the reconstructed images.

In summary, these experiments tend to show that TV- and ℓ_1-regularizations constitute interesting priors when displacements are unknown. On the other hand, standard ℓ_2-regularized approaches turn out to be satisfactory when the displacements are known, yielding to relatively fair reconstructions. This tendency observed in table 5.2 is also confirmed by the numerous details visible in Fig. 3, 4 and 6. The simulations also suggest that a gain in accuracy can be obtained by using longer sequences, although small sequences of a few images are sufficient for low down-sampling factors.

Let us finally underline the importance of the motion estimation in Algorithms 2 and 4. Indeed, as shown in Fig. 7, errors in motion discontinuity estimation clearly induce accumulation of errors in the reconstructed HR image sequence. For example, the little dragon's nose presents a noisy border with the background which corresponds to an imprecise estimation of motion discontinuity. However, motion fields remain fairly accurate although the estimation is carried out by using low-resolution observations. Table 5.3 shows the gain in motion estimation accuracy with Algorithm 4 when T increases, *i.e.*, as a function of the length of the image sequence. The repercussion of the gain obtained for $T = 4$ on the reconstructed HR images is clear for video SR with down-sampling factor equal to 4. It is less obvious for a lower SR factor.

6. Conclusion. We have presented a general methodology to solve video SR problems, *i.e.*, to reconstruct a HR image sequence from LR observations. The HR sequence is entirely described by a parametric non-linear dynamics. This dynamics, which connects the different images of the sequence, is parametrized by a final condition, a sequence of non-global displacement fields and a sequence of additive noises. To decrease the ill-possedness of the video SR problem, we assume different forms of prior on the unknown parameters of the dynamical system, with a special focus on sparse models. The joint estimation of the final condition, the displacement and the noise sequences takes the form of a difficult constrained minimization problem, which is, in the general case, high-dimensional, non-differentiable and non-convex. We provide elementary building blocks to tackle each of these specificities, and, by gathering them, we design convergent optimization algorithms enjoying a complexity linear in the problem dimensions. As a first contribution, we derive an optimal control framework to efficiently minimize a cost function while exploiting the nested structure of

the dynamical constraints. Optimization relies on a sequence of backward-forward recursions. This approach enables to avoid approximations used for Kalman smoothing in video SR state-of-the-art methods. An ADMM strategy able to cope in the particular convex case with the non-differentiability of ℓ_1-norms or TV-regularizers is then proposed. This optimization procedure is efficient as it alternates explicit proximal mappings and backward-forward recursions. The last building block ensures the convergence towards a critical point of the cost function in the general non-convex case. It is a two-step recursion which exploits a quadratic approximation of the non-convex terms. Four algorithms derived from this general methodology are numerically evaluated. This assessment provides a proof of concept for the efficiency of the proposed methodologies. Besides, the experiments show the relevance of sparse priors when the problem becomes severely undetermined and highlight the importance of motion estimation.

Appendix A. Proof of (4.5)**-**(4.7)**.** The proof of this specific *backward* optimal control solution follows the sketch of the demonstration for the more standard *forward* problem presented in [6]. We will focus on the following optimization problem:

$$\underset{(\boldsymbol{\epsilon},\mathbf{d},\mathbf{c})}{\arg\min}\, \mathcal{T}(\mathbf{x} = \mathcal{Q}(\boldsymbol{\epsilon},\mathbf{d},\mathbf{c}), \boldsymbol{\epsilon},\mathbf{d},\mathbf{c}), \tag{A.1}$$

where $\mathcal{T}$ denotes some objective function to be defined below. We recall that, given $(\boldsymbol{\epsilon},\mathbf{d},\ \mathbf{c})$, the function $\mathcal{Q}(\boldsymbol{\epsilon},\mathbf{d},\mathbf{c})$ determines a unique vector $\mathbf{x} = \mathcal{Q}(\boldsymbol{\epsilon},\mathbf{d},\mathbf{c})$ satisfying the constraints in (4.1), see section 4.1. In this appendix, we will use the following short-hand notation for the constraints in (4.1):

$$\begin{cases} \mathbf{x}_t = \mathcal{F}_t(\mathbf{x}_{t+1}, \boldsymbol{\epsilon}_{t+1}, \mathbf{d}_{t+1}), \quad 0 \leq t \leq T-1, \\ \mathbf{x}_T = \mathbf{D}\mathbf{c}, \end{cases} \tag{A.2}$$

with $\mathcal{F}_t(\mathbf{x}_{t+1}, \boldsymbol{\epsilon}_{t+1}, \mathbf{d}_{t+1}) \triangleq \mathcal{P}(\mathbf{x}_{t+1}, \mathbf{d}_{t+1}) + \boldsymbol{\epsilon}_{t+1}$. We will also alleviate the notation for the constraint $\mathbf{x} = \mathcal{Q}(\boldsymbol{\epsilon},\mathbf{d},\mathbf{c})$ by denoting this vector simply by $\mathbf{x}$. Therefore, $\mathbf{x}$ should be understood as a function of $\boldsymbol{\epsilon},\mathbf{d},\mathbf{c}$ and no longer as an independent variable.

The proof of (4.5)-(4.7) is made of two different parts, in which we study different instances of optimization problem (A.1). In a first step, we will consider an objective function only depending on the initial state[11]:

$$\mathcal{T}(\mathbf{x},\boldsymbol{\epsilon},\mathbf{d},\mathbf{c}) \triangleq \mathcal{G}_0(\mathbf{x}_0). \tag{A.3}$$

Then, in a second step, we will come back to the more general problem (4.2), *i.e.*, an optimization problem where the objective function $\mathcal{T}$ will match the cost function, $\mathcal{J}$, given in (4.3):

$$\mathcal{T}(\mathbf{x},\boldsymbol{\epsilon},\mathbf{d},\mathbf{c}) \triangleq \mathcal{G}_0(\mathbf{x}_0) + \sum_{t=1}^{T-1} \mathcal{G}_t(\mathbf{x}_t,\boldsymbol{\epsilon}_t,\mathbf{d}_t) + \mathcal{G}_T(\mathbf{x}_T,\boldsymbol{\epsilon}_T,\mathbf{d}_T,\mathbf{c}) \triangleq \mathcal{J}(\mathbf{x},\boldsymbol{\epsilon},\mathbf{d},\mathbf{c}). \tag{A.4}$$

First Part of the Proof. We begin by considering problem (A.1) with the objective function (A.3). By the chain rule of derivation applied to (A.2) at some

[11] As mentioned previously, $\mathbf{x}_0$ must be understood as a function of $\boldsymbol{\epsilon},\mathbf{d},\mathbf{c}$.

point in the set

$$\left\{ (\mathbf{x}', \boldsymbol{\epsilon}', \mathbf{d}', \mathbf{c}') \;\middle|\; \begin{array}{c} \mathbf{x}'_t = \mathcal{F}_t(\mathbf{x}'_{t+1}, \boldsymbol{\epsilon}'_{t+1}, \mathbf{d}'_{t+1}),\ 0 \leq t \leq T-1 \\ \mathbf{x}'_T = \mathbf{D}\mathbf{c}' \end{array} \right\}, \tag{A.5}$$

we can decompose the gradients into the products:

$$\left\{ \begin{array}{l} \nabla_{\boldsymbol{\epsilon}_t} \mathcal{T}(\mathbf{x}', \boldsymbol{\epsilon}', \mathbf{d}', \mathbf{c}') = \nabla_{\boldsymbol{\epsilon}_t} \mathcal{F}^*_{t-1}(\mathbf{x}'_t, \boldsymbol{\epsilon}'_t, \mathbf{d}'_t)\, \nabla_{\mathbf{x}_{t-1}} \mathcal{F}^*_{t-2} \cdots \nabla_{\mathbf{x}_2} \mathcal{F}^*_1\, \nabla_{\mathbf{x}_1} \mathcal{F}^*_0\, \nabla_{\mathbf{x}_0} \mathcal{G}_0(\mathbf{x}'_0), \\ \nabla_{\mathbf{d}_t} \mathcal{T}(\mathbf{x}', \boldsymbol{\epsilon}', \mathbf{d}', \mathbf{c}') = \nabla_{\mathbf{d}_t} \mathcal{F}^*_{t-1}(\mathbf{x}'_t, \boldsymbol{\epsilon}'_t, \mathbf{d}'_t)\, \nabla_{\mathbf{x}_{t-1}} \mathcal{F}^*_{t-2} \cdots \nabla_{\mathbf{x}_2} \mathcal{F}^*_1\, \nabla_{\mathbf{x}_1} \mathcal{F}^*_0\, \nabla_{\mathbf{x}_0} \mathcal{G}_0(\mathbf{x}'_0), \\ \nabla_{\mathbf{c}} \mathcal{T}(\mathbf{x}', \boldsymbol{\epsilon}', \mathbf{d}', \mathbf{c}') = \mathbf{D}^* \nabla_{\mathbf{x}_T} \mathcal{F}^*_{T-1} \cdots \nabla_{\mathbf{x}_2} \mathcal{F}^*_1\, \nabla_{\mathbf{x}_1} \mathcal{F}^*_0\, \nabla_{\mathbf{x}_0} \mathcal{G}_0(\mathbf{x}'_0), \end{array} \right. \tag{A.6}$$

where we recall that $\nabla_{\mathbf{x}_t} \mathcal{F}_{t-1}$ denotes the Jacobian matrice of $\mathcal{F}_{t-1}$ with respect to function $\mathbf{x}_t$ evaluated at $(\mathbf{x}'_t, \boldsymbol{\epsilon}'_t, \mathbf{d}'_t)$ and $\nabla_{\mathbf{x}_t} \mathcal{F}^*_{t-1}$ its transpose. We can rewrite gradients in (A.6) in order to exhibit their recursive structures. By defining the forward recursion

$$\left\{ \begin{array}{l} \boldsymbol{\zeta}_0 = \nabla_{\mathbf{x}_0} \mathcal{G}_0(\mathbf{x}'_0), \\ \boldsymbol{\zeta}_t = \nabla_{\mathbf{x}_t} \mathcal{F}^*_{t-1} \boldsymbol{\zeta}_{t-1}, \quad 1 \leq t \leq T, \end{array} \right. \tag{A.7}$$

we obtain the following rewriting:

$$\left\{ \begin{array}{l} \nabla_{\boldsymbol{\epsilon}_t} \mathcal{T}(\mathbf{x}', \boldsymbol{\epsilon}', \mathbf{d}', \mathbf{c}') = \nabla_{\boldsymbol{\epsilon}_t} \mathcal{F}^*_{t-1}(\mathbf{x}'_t, \boldsymbol{\epsilon}'_t, \mathbf{d}'_t) \boldsymbol{\zeta}_{t-1}, \quad 1 \leq t \leq T, \\ \nabla_{\mathbf{d}_t} \mathcal{T}(\mathbf{x}', \boldsymbol{\epsilon}', \mathbf{d}', \mathbf{c}') = \nabla_{\mathbf{d}_t} \mathcal{F}^*_{t-1}(\mathbf{x}'_t, \boldsymbol{\epsilon}'_t, \mathbf{d}'_t) \boldsymbol{\zeta}_{t-1}, \quad 1 \leq t \leq T, \\ \nabla_{\mathbf{c}} \mathcal{T}(\mathbf{x}', \boldsymbol{\epsilon}', \mathbf{d}', \mathbf{c}') = \mathbf{D}^* \boldsymbol{\zeta}_T. \end{array} \right. \tag{A.8}$$

Second Part of the Proof. We now consider problem (A.1) with objective function (A.4). By making a change of variables, we want to obtain a rewriting of function (A.4) with a structure analogous to (A.3), so that the gradients are given by a recursion of the form (A.7)-(A.8). In other words, by making some change of variables we intend to rewrite the sum of functions in (A.4) as a unique function depending solely on an "initial state". In order to do so, let us define variables $\boldsymbol{\kappa}_i$'s recursively as follows:

$$\left\{ \begin{array}{l} \boldsymbol{\kappa}_T = 0, \\ \boldsymbol{\kappa}_{T-1}(\mathbf{x}_T, \boldsymbol{\epsilon}_T, \mathbf{d}_T, \mathbf{c}) = \boldsymbol{\kappa}_T + \mathcal{G}_T(\mathbf{x}_T, \boldsymbol{\epsilon}_T, \mathbf{d}_T, \mathbf{c}), \\ \boldsymbol{\kappa}_{t-1}(\mathbf{x}_t, \boldsymbol{\epsilon}_t, \mathbf{d}_t, \mathbf{c}) = \boldsymbol{\kappa}_t + \mathcal{G}_t(\mathbf{x}_t, \boldsymbol{\epsilon}_t, \mathbf{d}_t), \quad T-1 \geq t \geq 1. \end{array} \right.$$

We then obtain that

$$\boldsymbol{\kappa}_0(\mathbf{x}, \boldsymbol{\epsilon}, \mathbf{d}, \mathbf{c}) = \sum_{t=1}^{T-1} \mathcal{G}_t(\mathbf{x}_t, \boldsymbol{\epsilon}_t, \mathbf{d}_t) + \mathcal{G}_T(\mathbf{x}_T, \boldsymbol{\epsilon}_T, \mathbf{d}_T, \mathbf{c}),$$

and the objective function $\mathcal{T}$ given in (A.4) can be rewritten as

$$\mathcal{T}(\mathbf{x}, \boldsymbol{\epsilon}, \mathbf{d}, \mathbf{c}) = \boldsymbol{\kappa}_0(\mathbf{x}, \boldsymbol{\epsilon}, \mathbf{d}, \mathbf{c}) + \mathcal{G}_0(\mathbf{x}_0). \tag{A.9}$$

Considering the following change of variables $\tilde{\mathbf{x}}_t \triangleq \begin{pmatrix} \mathbf{x}_t \\ \boldsymbol{\kappa}_t \end{pmatrix}$, we then have that the right-hand side of (A.9) can be rewritten as a function of $\tilde{\mathbf{x}}_0$ only. In the sequel, we will use the following specific notation to emphasize this fact:

$$\tilde{\mathcal{G}}_0(\tilde{\mathbf{x}}_0) = \boldsymbol{\kappa}_0 + \mathcal{G}_0(\mathbf{x}_0). \tag{A.10}$$

Moreover, it is easy to see that functions $\tilde{\mathbf{x}}_t$'s satisfy the following backward recursion:

$$\begin{cases} \tilde{\mathbf{x}}_t = \tilde{\mathcal{F}}_t(\tilde{\mathbf{x}}_{t+1}, \boldsymbol{\epsilon}_{t+1}, \mathbf{d}_{t+1}), \quad T-1 \geq t \geq 0, \\ \tilde{\mathbf{x}}_T = \tilde{\mathcal{F}}_T(\boldsymbol{\epsilon}_T, \mathbf{d}_T, \mathbf{c}), \end{cases} \tag{A.11}$$

where

$$\tilde{\mathcal{F}}_t(\tilde{\mathbf{x}}_{t+1}, \boldsymbol{\epsilon}_{t+1}, \mathbf{d}_{t+1}) = \begin{pmatrix} \mathcal{F}_t(\mathbf{x}_{t+1}, \boldsymbol{\epsilon}_{t+1}, \mathbf{d}_{t+1}) \\ \boldsymbol{\kappa}_{t+1} + \mathcal{G}_{t+1}(\mathbf{x}_{t+1}, \boldsymbol{\epsilon}_{t+1}, \mathbf{d}_{t+1}) \end{pmatrix},$$
$$\tilde{\mathcal{F}}_T(\boldsymbol{\epsilon}_T, \mathbf{d}_T, \mathbf{c}) = \begin{pmatrix} \mathbf{D}\mathbf{c} \\ \mathcal{G}_T(\mathbf{D}\mathbf{c}, \boldsymbol{\epsilon}_T, \mathbf{d}_T, \mathbf{c}) \end{pmatrix}.$$

We remark that the cost function (A.10), recursion (A.11) and the set

$$\left\{ (\tilde{\mathbf{x}}', \boldsymbol{\epsilon}', \mathbf{d}', \mathbf{c}') \;\middle|\; \begin{array}{c} \tilde{\mathbf{x}}'_t = \tilde{\mathcal{F}}_t(\tilde{\mathbf{x}}'_{t+1}, \boldsymbol{\epsilon}'_{t+1}, \mathbf{d}'_{t+1}), \;\; 0 \leq t \leq T-1 \\ \tilde{\mathbf{x}}'_T = \tilde{\mathcal{F}}_T(\boldsymbol{\epsilon}'_T, \mathbf{d}'_T, \mathbf{c}') \end{array} \right\}, \tag{A.12}$$

have respectively the same structure as (A.3), (A.2) and (A.5). We can then apply the result obtained previously and get the gradients of $\mathcal{T}$ using the same reasoning as the one made to derive (A.7)-(A.8). More specifically, let $(\tilde{\mathbf{x}}', \boldsymbol{\epsilon}', \mathbf{d}', \mathbf{c}')$ be some point in (A.12), and let $\tilde{\boldsymbol{\zeta}}_t$ be an "adjoint" variable verifying:

$$\begin{cases} \tilde{\boldsymbol{\zeta}}_0 = \nabla_{\tilde{\mathbf{x}}_0} \tilde{\mathcal{G}}_0(\tilde{\mathbf{x}}'_0), \\ \tilde{\boldsymbol{\zeta}}_t = \nabla_{\tilde{\mathbf{x}}_t} \tilde{\mathcal{F}}^*_{t-1} \tilde{\boldsymbol{\zeta}}_{t-1}, \quad 1 \leq t \leq T, \end{cases} \tag{A.13}$$

where the Jacobian matrix of $\tilde{\mathcal{F}}_{t-1}$ evaluated at some point $(\tilde{\mathbf{x}}'_t, \boldsymbol{\epsilon}'_t, \mathbf{d}'_t)$ is denoted $\nabla_{\tilde{\mathbf{x}}_t} \tilde{\mathcal{F}}_{t-1}$. Using (A.8), we obtain the following expressions:

$$\begin{cases} \nabla_{\mathbf{d}_t} \mathcal{T}(\tilde{\mathbf{x}}', \boldsymbol{\epsilon}', \mathbf{d}', \mathbf{c}') = \nabla_{\mathbf{d}_t} \tilde{\mathcal{F}}^*_{t-1}(\tilde{\mathbf{x}}'_t, \boldsymbol{\epsilon}'_t, \mathbf{d}'_t) \tilde{\boldsymbol{\zeta}}_{t-1}, \quad 1 \leq t \leq T, \\ \nabla_{\boldsymbol{\epsilon}_t} \mathcal{T}(\tilde{\mathbf{x}}', \boldsymbol{\epsilon}', \mathbf{d}', \mathbf{c}') = \nabla_{\boldsymbol{\epsilon}_t} \tilde{\mathcal{F}}^*_{t-1}(\tilde{\mathbf{x}}'_t, \boldsymbol{\epsilon}'_t, \mathbf{d}'_t) \tilde{\boldsymbol{\zeta}}_{t-1}, \quad 1 \leq t \leq T, \\ \nabla_{\mathbf{c}} \mathcal{T}(\tilde{\mathbf{x}}', \boldsymbol{\epsilon}', \mathbf{d}', \mathbf{c}') = \nabla_{\mathbf{c}} \tilde{\mathcal{F}}^*_T(\boldsymbol{\epsilon}'_T, \mathbf{d}'_T, \mathbf{c}') \tilde{\boldsymbol{\zeta}}_T. \end{cases}$$

To finalize the proof, we re-express recursion (A.13) by developing it with respect to the two different components of the adjoint variable $\tilde{\boldsymbol{\zeta}}_t \triangleq \begin{pmatrix} \boldsymbol{\zeta}_t \\ \omega_t \end{pmatrix}$, where the $\boldsymbol{\zeta}_t$'s have the dimension of $\mathbf{x}_t$'s and ω_t's are scalars. Particularizing the first equation in (A.13) by taking (A.10) into account, we obtain

$$\begin{pmatrix} \boldsymbol{\zeta}_0 \\ \omega_0 \end{pmatrix} = \begin{pmatrix} \nabla_{\mathbf{x}_0} \tilde{\mathcal{G}}_0(\tilde{\mathbf{x}}'_0) \\ \nabla_{\boldsymbol{\kappa}_0} \tilde{\mathcal{G}}_0(\tilde{\mathbf{x}}'_0) \end{pmatrix} = \begin{pmatrix} \nabla \mathcal{G}_0(\mathbf{x}'_0) \\ 1 \end{pmatrix}. \tag{A.14}$$

Moreover, using the definition of $\tilde{\mathcal{F}}_t$, the second equation in (A.13) leads to

$$\begin{pmatrix} \boldsymbol{\zeta}_t \\ \omega_t \end{pmatrix} = \begin{pmatrix} \nabla_{\mathbf{x}_t} \mathcal{F}^*_{t-1} & \nabla_{\mathbf{x}_t} \mathcal{G}_t(\mathbf{x}'_t, \boldsymbol{\epsilon}'_t, \mathbf{d}'_t) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{\zeta}_{t-1} \\ \omega_{t-1} \end{pmatrix}, \quad 1 \leq t \leq T-1, \tag{A.15}$$

$$\begin{pmatrix} \boldsymbol{\zeta}_T \\ \omega_T \end{pmatrix} = \begin{pmatrix} \nabla_{\mathbf{x}_T} \mathcal{F}^*_{T-1} & \nabla_{\mathbf{x}_T} \mathcal{G}_T(\mathbf{x}'_T, \boldsymbol{\epsilon}'_T, \mathbf{d}'_T, \mathbf{c}') \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{\zeta}_{T-1} \\ \omega_{T-1} \end{pmatrix}. \tag{A.16}$$

Equations (A.14)-(A.16) imply that $\omega_t = 1\ \forall t$; moreover, the recursion in $\boldsymbol{\zeta}_t$ is equivalent to (4.7).

Appendix B. The Alternating Direction Method of Multipliers. The alternating direction method of multipliers (ADMM) focusses on the following type of optimization problems:

$$\min_{\boldsymbol{z}_1\in\Xi_1,\boldsymbol{z}_2\in\Xi_1} \mathcal{G}_1(\boldsymbol{z}_1)+\mathcal{G}_2(\boldsymbol{z}_2), \quad \text{s.t. } \mathbf{A}\boldsymbol{z}_1+\mathbf{B}\boldsymbol{z}_2=\mathbf{0}_r, \tag{B.1}$$

where $\mathbf{A}\in\mathbb{R}^{r\times n_1}$, $\mathbf{B}\in\mathbb{R}^{r\times n_2}$, $\mathcal{G}_1:\mathbb{R}^{n_1}\to\mathbb{R}$, $\mathcal{G}_2:\mathbb{R}^{n_2}\to\mathbb{R}$ are closed, proper and convex functions, and Ξ_1, Ξ_2 are non-empty convex sets. We note that the conditions on $\mathcal{G}_1$ and $\mathcal{G}_2$ are pretty mild; in particular, $\mathcal{G}_1$ and $\mathcal{G}_2$ are not required to be differentiable and can take on infinite values.

ADMM is an iterative procedure inspired by the well-known method of multipliers [6]. It searches for a minimizer of (B.1) by sequentially minimizing the corresponding augmented Lagrangian with respect to each primal variables $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$, before updating a dual variable $\mathbf{u}\in\mathbb{R}^r$. Formally, the ADMM recursions take the form:

$$\boldsymbol{z}_1^{(k+1)} = \arg\min_{\boldsymbol{z}_1\in\Xi_1} \mathcal{G}_1(\boldsymbol{z}_1)+\frac{\rho}{2}\|\mathbf{A}\boldsymbol{z}_1+\mathbf{B}\boldsymbol{z}_2^{(k)}+\mathbf{u}^{(k)}\|_2^2, \tag{B.2}$$

$$\boldsymbol{z}_2^{(k+1)} = \arg\min_{\boldsymbol{z}_2\in\Xi_2} \mathcal{G}_2(\boldsymbol{z}_2)+\frac{\rho}{2}\|\mathbf{A}\boldsymbol{z}_1^{(k+1)}+\mathbf{B}\boldsymbol{z}_2+\mathbf{u}^{(k)}\|_2^2, \tag{B.3}$$

$$\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)}+\mathbf{A}\boldsymbol{z}_1^{(k+1)}+\mathbf{B}\boldsymbol{z}_2^{(k+1)}, \tag{B.4}$$

for some $\rho>0$.

ADMM has recently sparked a surge of interest in the signal-processing community for several reasons. First, the conditions on $\mathcal{G}_1$ and $\mathcal{G}_2$ in (B.1) (*i.e.*, closed, proper and convex) are mild and (B.1) therefore encompasses a large number of optimization problems as particular cases. Second, the ADMM recursion (B.2)-(B.4) converges to a solution of (B.1) under very general conditions, see [8, section 3.2]. Third, although ADMM is known to be slow to converge to a solution with high accuracy, it has been shown empirically that ADMM converges to modest accuracy in a few tens of iterations.

Appendix C. Algorithm's Details.

C.1. Computation of (4.5)-(4.7). In this appendix, we complement the exposition done in section 4.1 on the fast evaluation of gradient of cost function $\mathcal{J}(\boldsymbol{\epsilon},\mathbf{d},\mathbf{c})$ given in (4.3), particularized to the model parameters specified in section 5.1. First of all, we expose the particularization of recursions (4.5)-(4.7) to this setting. It is straightforward to see that it results in the following procedure:

i) Compute sequence $\{\mathbf{x}'_t\}_{t=0}^T$ by the backward recursion:
$$\begin{cases}\mathbf{x}'_T=\mathbf{D}\mathbf{c}',\\ \mathbf{x}'_t=\mathcal{P}(\mathbf{x}'_{t+1},\mathbf{d}'_{t+1})+\boldsymbol{\epsilon}'_{t+1}.\end{cases}$$

ii) Compute sequence $\{\boldsymbol{\zeta}_t\}_{t=0}^T$ by the forward recursion:
$$\begin{cases}\boldsymbol{\zeta}_0=2\nabla_{\mathbf{x}_0}\mathcal{H}^*(\mathcal{H}(\mathbf{x}'_0)-\mathbf{y}_0),\\ \boldsymbol{\zeta}_{t+1}=\nabla_{\mathbf{x}_{t+1}}\mathcal{P}^*(\mathbf{x}'_{t+1},\mathbf{d}'_{t+1})\boldsymbol{\zeta}_t,+2\nabla_{\mathbf{x}_{t+1}}\mathcal{H}^*(\mathcal{H}(\mathbf{x}'_{t+1})-\mathbf{y}_{t+1}).\end{cases}$$

iii) Compute the gradients:

$$\begin{cases} \nabla_{\boldsymbol{\epsilon}_t}\mathcal{J}(\boldsymbol{\epsilon}', \mathbf{d}', \mathbf{c}') = \boldsymbol{\zeta}_{t-1} + 2\alpha_1\boldsymbol{\epsilon}'_t, \\ \nabla_{\mathbf{d}_t}\mathcal{J}(\boldsymbol{\epsilon}', \mathbf{d}', \mathbf{c}') = \mathbf{E}_t\boldsymbol{\zeta}_{t-1} + 2\alpha_2\mathbf{W}_t\mathbf{G}\mathbf{G}^*\mathbf{d}'_t, \\ \nabla_{\mathbf{c}}\mathcal{J}(\boldsymbol{\epsilon}', \mathbf{d}', \mathbf{c}') = \mathbf{D}^*\boldsymbol{\zeta}_T + 2\alpha_3\mathbf{c}' \end{cases}$$

where $\mathbf{W}_t \in \mathbb{R}^{2n\times 2n}$ and $\mathbf{E}_t \in \mathbb{R}^{2n\times n}$ are respectively diagonal and block-diagonal matrices which will be defined in the following. We detail hereafter the elements of the procedure which have not been fully described yet.

We begin by making some comments on the evaluation of the warping function $\mathcal{P}(\mathbf{x}'_t, \mathbf{d}'_t)$ and its Jacobian $\nabla_{\mathbf{x}_t}\mathcal{P}(\mathbf{x}'_t, \mathbf{d}'_t)$, which constitute the core of the recursion. We propose to use the family of bi-dimensional cubic cardinal splines $\{\psi_i\}_{i=1}^n$ for the representation (3.3). In practice, we compute an equivalent representation based on the family of bi-dimensional cubic B-splines functions $\{\phi_i\}_{i=1}^n$. Indeed, this representation presents some computational advantages because of the existence of fast B-splines transforms. The relation between cardinal cubic splines and cubic B-splines functions is given in [47]. This reference also provides details on the fast cubic B-splines transform by recursive filtering. Let matrix $\mathbf{C}^* = [\mathbf{c}_1, ..., \mathbf{c}_n]^* \in \mathbb{R}^{n\times n}$ denote the direct B-spline transform of a discrete bi-dimensional signal, *i.e.*, the transform computing from a discrete signal $\mathbf{x}_t$ its representation with spline coefficients $\mathbf{C}^*\mathbf{x}_t$. Rewritten (3.3) with cubic B-spline functions, we get:

$$\mathcal{P}_s(\mathbf{x}_t, \mathbf{d}_t) = \sum_{i\in\vartheta(\chi(s)+\mathbf{d}_t(s))} \mathbf{c}_i^*\mathbf{x}_t\, \phi_i(\chi(s) + \mathbf{d}_t(s)), \tag{C.1}$$

where $\vartheta(\chi(s)+\mathbf{d}_t(s))$ denotes a subset of vector indices corresponding to the neighborhood of the spatial position $\chi(s)$ (which differs from the subset $\mathcal{V}_s$ previously defined in (3.3)). To simplify notations, we denote by $\mathcal{I} : \mathbb{R}^n \times \mathbb{R}^{2n} \to \mathbb{R}^n$ the function taking as a first argument spline coefficients $\mathbf{C}^*\mathbf{x}_t$ and as a second argument a motion field $\mathbf{d}_t$, and whose s-th component is given by (C.1). Using this notation, (C.1) can be rewritten in the vectorial form

$$\mathcal{P}(\mathbf{x}_t, \mathbf{d}_t) = \mathcal{I}(\mathbf{C}^*\mathbf{x}_t, \mathbf{d}_t).$$

We denote by $\nabla\mathcal{I}(\mathbf{C}^*\mathbf{x}_t, \mathbf{d}_t)$ the Jacobian of function $\mathcal{I}$ at point $(\mathbf{C}^*\mathbf{x}_t, \mathbf{d}_t)$ with respect to its first argument, *i.e.*, spline coefficients. Since function $\mathcal{I}$ is linear with respect to spline coefficients, the Jacobian is only dependent on the value of its second argument, *i.e.*, $\mathbf{d}_t$. Therefore, we will adopt the notation $\nabla\mathcal{I}(\mathbf{d}_t)$ in the sequel.

The complexity of evaluating both spline coefficients $\mathbf{C}^*\mathbf{x}_t$ and the interpolated function $\mathcal{I}$, scales linearly with the image dimension, *i.e.*, $\mathcal{O}(n)$, thanks to the representation separability and to recursive linear filtering [47]. Multiplication with the Jacobian transpose

$$\nabla_{\mathbf{x}_t}\mathcal{P}^*(\mathbf{x}_{t+1}, \mathbf{d}_{t+1}) = \mathbf{C}\nabla\mathcal{I}^*(\mathbf{d}_{t+1}),$$

implies also a linear complexity: first, matrix $\mathbf{C}$ is symmetric[12] so that it is identical to the direct B-spline transformation $\mathbf{C}^*$, computed by recursive linear filtering; second, the multiplication of the Jacobian transpose of function $\mathcal{I}$ with vector $\boldsymbol{\zeta}_t$ is equal to

$$\nabla\mathcal{I}^*(\mathbf{d}_t)\boldsymbol{\zeta}_t(s) = \sum_{i|s\in\vartheta(\chi(i)+\mathbf{d}_t(i))} \boldsymbol{\zeta}_t(i)\phi_s(\chi(i) + \mathbf{d}_t(i)).$$

[12] Matrix $\mathbf{C}$ is symmetric in the case of periodic boundary conditions [47].

Concerning the Jacobian transpose $\nabla_{\mathbf{x}_t}\mathcal{H}^*$, it is easy to see that this matrix is an up-sampling operation, inserting zeros, followed by the same low-pass filtering as in $\mathcal{H}$.

We continue by detailing matrices appearing in the last step of the procedure. First, we note that matrix $\mathbf{D}^*$ is simply the direct Real-valued Fast Curvelet Transform. This transform is, as well as its transpose $\mathbf{D}$, based on fast Fourier transforms, whose complexity scales in $\mathcal{O}(n\log n)$ [32]. Next, the two diagonals of the two-block matrix $\mathbf{E}_t$ are the two n-dimensional vectors $\partial_{s_j}\left(\mathcal{I}(\mathbf{C}^*\mathbf{x}_t,\mathbf{d}_t)\right)$ for $j=1,2$, where s_j denotes the j-th spatial coordinate. We approach these partial derivatives by second-order centered finite differences. Then, the diagonal of matrix $\mathbf{W}_t$ is the vector concatening twice the weight vector $\mathbf{w}_t$, *i.e.*, $\mathbf{W}_t(s,s)=\mathbf{W}_t(2s,2s)=\mathbf{w}_t(s)$ for $s=1,...,n$.

To finalize the description of this procedure, it remains to give some details on matrix $\mathbf{G}$. Let the elements of vector $\mathbf{G}^*\mathbf{d}_t$ be first-order forward finite-difference approximations of the spatial gradients of the two motion components, which have been rearranged beforehand on the pixel grid. This gradient approximation becomes exact assuming that components of vector $\mathbf{d}_t$ are coefficients associated to the decomposition of some continuous motion field in a basis of interpolating and separable scaling functions (see a proof in [31]). Straightforward calculus then show that elements of vector $\mathbf{G}\mathbf{G}^*\mathbf{d}_t$ are second-order finite difference approximations of the Laplacian of the two motion components, which have been rearranged beforehand on the pixel grid.

C.2. ADMM Resolution of Minimization Problems (4.17) **and** (4.19)**.** In this appendix, we present an ADMM implementation of the two minimization problems (4.17) and (4.19) appearing in the procedure described in section 4.3 (which also corresponds to Algorithm 4 introduced later on in section 5). In the following, iterations of the 2-step recursion presented in section 4.3 will be indexed by the exponent $^{(\ell)}$, in order to differentiate them from the iterations related to ADMM, which will be indexed by the exponent $^{(k)}$.

We begin by the analysis of minimization problem (4.17). This problem can be equivalently reexpressed as:

$$\underset{(\mathbf{x},\boldsymbol{\epsilon},\mathbf{c})\in\Omega,(\tilde{\boldsymbol{\epsilon}},\tilde{\mathbf{c}})}{\arg\min}\sum_{t=0}^{T}\|\mathcal{H}(\mathbf{x}_t)-\mathbf{y}_t\|_2^2+\sum_{t=1}^{T}\left(\alpha_1\|\tilde{\boldsymbol{\epsilon}}_t\|_1+\gamma\|\tilde{\delta}_{\boldsymbol{\epsilon}_t}\|_1\right)+\alpha_3\|\tilde{\mathbf{c}}\|_1+\gamma\,\|\tilde{\delta}_{\mathbf{c}}\|_1$$
$$\text{s.t.}\quad\begin{cases}\boldsymbol{\epsilon}_t=\tilde{\boldsymbol{\epsilon}}_t, & \forall t,\\ \mathbf{H}^*(\boldsymbol{\epsilon}_t-\boldsymbol{\epsilon}_t^{(\ell)})=\tilde{\delta}_{\boldsymbol{\epsilon}_t}, & \forall t,\\ \mathbf{c}=\tilde{\mathbf{c}},\\ \mathbf{c}-\mathbf{c}^{(\ell)}=\tilde{\delta}_{\mathbf{c}},\end{cases}$$

where

$$\Omega\triangleq\left\{(\mathbf{x},\boldsymbol{\epsilon},\mathbf{c})\;\middle|\;\begin{array}{c}\mathbf{x}_t=\mathcal{P}(\mathbf{x}_{t+1},\mathbf{d}_{t+1})+\boldsymbol{\epsilon}_{t+1},\;\;0\le t\le T-1\\ \mathbf{x}_T=\mathbf{D}\mathbf{c}\end{array}\right\}.$$

Here, we have added four new variables to the problem, $\tilde{\boldsymbol{\epsilon}}=(\tilde{\boldsymbol{\epsilon}}_1,...,\tilde{\boldsymbol{\epsilon}}_T)$, $\tilde{\mathbf{c}}$, $\tilde{\delta}_{\boldsymbol{\epsilon}}=(\tilde{\delta}_{\boldsymbol{\epsilon}_1},...,\tilde{\delta}_{\boldsymbol{\epsilon}_T})$ and $\tilde{\delta}_{\mathbf{c}}$, which are counterbalanced by the inclusion of four new constraints. We use the formalism exposed in appendix B with $\boldsymbol{z}_1=(\mathbf{x},\boldsymbol{\epsilon},\mathbf{c})$,

$\boldsymbol{z}_2 = (\tilde{\boldsymbol{\epsilon}}, \tilde{\mathbf{c}}, \tilde{\delta}_{\boldsymbol{\epsilon}}, \tilde{\delta}_{\mathbf{c}})$, $\Xi_1 = \Omega$ and $\Xi_2 = \mathbb{R}^{nT} \times \mathbb{R}^q \times \mathbb{R}^{nT} \times \mathbb{R}^q$ and obtain the following ADMM recursions:

$$(\mathbf{x}^{(k+1)}, \boldsymbol{\epsilon}^{(k+1)}, \mathbf{c}^{(k+1)}) = \underset{(\mathbf{x},\boldsymbol{\epsilon},\mathbf{c})\in\Omega}{\arg\min}\, \mathcal{L}^{(k)}(\mathbf{x}, \boldsymbol{\epsilon}, \mathbf{c}) + \frac{\rho}{2}\sum_{t=1}^{T} \|\mathbf{H}^*(\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_t^{(\ell)}) - \tilde{\delta}_{\boldsymbol{\epsilon}_t}^{(k)} + \mathbf{u}_{\delta_{\boldsymbol{\epsilon}_t}}^{(k)}\|_2^2 + \frac{\rho}{2}\|\mathbf{c} - \mathbf{c}^{(\ell)} - \tilde{\delta}_{\mathbf{c}}^{(k)} + \mathbf{u}_{\delta_{\mathbf{c}}}^{(k)}\|_2^2, \tag{C.2}$$

$$\begin{cases} \tilde{\boldsymbol{\epsilon}}_t^{(k+1)} = & \arg\min_{\tilde{\boldsymbol{\epsilon}}_t} \ \|\tilde{\boldsymbol{\epsilon}}_t\|_1 + \frac{\rho}{2\alpha_1}\|\boldsymbol{\epsilon}_t^{(k+1)} - \tilde{\boldsymbol{\epsilon}}_t + \mathbf{u}_{\boldsymbol{\epsilon}_t}^{(k)}\|_2^2, \\ \tilde{\mathbf{c}}^{(k+1)} = & \arg\min_{\tilde{\mathbf{c}}} \ \|\tilde{\mathbf{c}}\|_1 + \frac{\rho}{2\alpha_3}\|\mathbf{c}^{(k+1)} - \tilde{\mathbf{c}} + \mathbf{u}_{\mathbf{c}}^{(k)}\|_2^2, \\ \tilde{\delta}_{\boldsymbol{\epsilon}_t}^{(k+1)} = & \arg\min_{\tilde{\delta}_{\boldsymbol{\epsilon}_t}} \ \|\tilde{\delta}_{\boldsymbol{\epsilon}_t}\|_1 + \frac{\rho}{2\gamma}\|\mathbf{H}^*(\boldsymbol{\epsilon}_t^{(k+1)} - \boldsymbol{\epsilon}_t^{(\ell)}) - \tilde{\delta}_{\boldsymbol{\epsilon}_t} + \mathbf{u}_{\delta_{\boldsymbol{\epsilon}_t}}^{(k)}\|_2^2, \\ \tilde{\delta}_{\mathbf{c}}^{(k+1)} = & \arg\min_{\tilde{\delta}_{\mathbf{c}}} \ \|\tilde{\delta}_{\mathbf{c}}\|_1 + \frac{\rho}{2\gamma}\|\mathbf{c}^{(k+1)} - \mathbf{c}^{(\ell)} - \tilde{\delta}_{\mathbf{c}} + \mathbf{u}_{\delta_{\mathbf{c}}}^{(k)}\|_2^2, \end{cases} \tag{C.3}$$

$$\begin{cases} \mathbf{u}_{\boldsymbol{\epsilon}_t}^{(k+1)} = & \mathbf{u}_{\boldsymbol{\epsilon}_t}^{(k)} + \boldsymbol{\epsilon}_t^{(k+1)} - \tilde{\boldsymbol{\epsilon}}_t^{(k+1)}, \\ \mathbf{u}_{\mathbf{c}}^{(k+1)} = & \mathbf{u}_{\mathbf{c}}^{(k)} + \mathbf{c}^{(k+1)} - \tilde{\mathbf{c}}^{(k+1)}, \\ \mathbf{u}_{\delta_{\boldsymbol{\epsilon}_t}}^{(k+1)} = & \mathbf{u}_{\delta_{\boldsymbol{\epsilon}_t}}^{(k)} + \mathbf{H}^*(\boldsymbol{\epsilon}_t^{(k+1)} - \boldsymbol{\epsilon}_t^{(\ell)}) - \tilde{\delta}_{\boldsymbol{\epsilon}_t}^{(k+1)}, \\ \mathbf{u}_{\delta_{\mathbf{c}}}^{(k+1)} = & \mathbf{u}_{\delta_{\mathbf{c}}}^{(k)} + \mathbf{c}^{(k+1)} - \mathbf{c}^{(\ell)} - \tilde{\delta}_{\mathbf{c}}^{(k+1)}, \end{cases} \tag{C.4}$$

where $\mathcal{L}^{(k)}$ is defined in (4.14). Equations (C.2), (C.3) and (C.4) correspond respectively to expressions (B.2), (B.3) and (B.4) in appendix B. We comment on the two first steps of the ADMM algorithm, the last one being trivial. First, we can apply the methodology described in section 4.1 to solve the constrained differentiable problem (C.2) via a gradient descent algorithm. We remark that the complexity associated to the evaluation of the gradient scales as $\mathcal{O}(nT+q)$. Second, the optimization problems specified in (C.3) all have simple analytical solutions based on soft-thresholding operators (4.16). We immediately remark that the two first updates in (C.3) are identical to the ADMM steps (4.12) used to treat the convex case in section 4.2. Moreover, the solutions to the last two problems in (C.3) are given by

$$\begin{cases} \tilde{\delta}_{\boldsymbol{\epsilon}_t}^{(k+1)}(i) & = \mathrm{soft}_{\frac{\gamma}{\rho}}\left(\mathbf{h}_i^*(\boldsymbol{\epsilon}_t^{(k+1)} - \boldsymbol{\epsilon}_t^{(\ell)}) + \mathbf{u}_{\delta_{\boldsymbol{\epsilon}_t}}^{(k)}(i)\right) \\ \tilde{\delta}_{\mathbf{c}}^{(k+1)}(i) & = \mathrm{soft}_{\frac{\gamma}{\rho}}\left(\mathbf{c}^{(k+1)}(i) - \mathbf{c}^{(\ell)}(i) + \mathbf{u}_{\delta_{\mathbf{c}}}^{(k)}(i)\right) \end{cases} \qquad \forall i, \tag{C.5}$$

where $\mathbf{h}_i$ is the ith column of $\mathbf{H}$ and "soft" denotes the soft-threshloding operator defined in (4.16).

We continue with the analysis of minimization problem (4.19). We first remark that we can apply the methodology described in section 4.1 to compute the gradient $\nabla_{\mathbf{d}_t}\mathcal{B}(\mathbf{d}^{(\ell)})$ required to build the quadratic approximation (4.20). Once this quadratic approximation has been obtained, the task now is to solve minimization problem (4.19). We can notice that this problem does unfortunately not possess an explicit solution. To circumvent this issue, we use an ADMM strategy, as detailed below. Problem (4.19) is reexpressed as:

$$\begin{aligned} &\underset{\mathbf{d},(\tilde{\mathbf{d}}_1,\ldots,\tilde{\mathbf{d}}_T)}{\arg\min} \quad \mathcal{B}(\mathbf{d}, \mathbf{d}^{(\ell)}) + \alpha_2 \sum_{t=1}^{T} \mathcal{R}(\tilde{\mathbf{d}}_t) \\ &\text{s.t.} \quad \mathbf{G}^*\mathbf{d}_t = \tilde{\mathbf{d}}_t, \qquad \forall t. \end{aligned} \tag{C.6}$$

Here, we have added the new variables $\tilde{\mathbf{d}}_t$'s to the problem which are counterbalanced by the inclusion of new constraints. We use the formalism exposed in appendix B

with $\boldsymbol{z}_1 = (\mathbf{d}_1, ..., \mathbf{d}_T)$, $\boldsymbol{z}_2 = (\tilde{\mathbf{d}}_1, ..., \tilde{\mathbf{d}}_T)$, $\Xi_1 = \mathbb{R}^{2nT}$ and $\Xi_2 = \mathbb{R}^{hT}$ and obtain the following ADMM recursions:

$$\mathbf{d}^{(k+1)} = \arg\min_{\mathbf{d}} \mathcal{B}(\mathbf{d}, \mathbf{d}^{(\ell)}) + \frac{\rho}{2} \sum_{t=1}^{T} \|\mathbf{G}^* \mathbf{d}_t - \tilde{\mathbf{d}}_t^{(k)} + \mathbf{u}_{\mathbf{d}_t}^{(k)}\|_2^2, \tag{C.7}$$

$$\tilde{\mathbf{d}}_t^{(k+1)} = \arg\min_{\tilde{\mathbf{d}}_t} \quad \mathcal{R}(\tilde{\mathbf{d}}_t) + \frac{\rho}{2\alpha_2} \|\mathbf{G}^* \mathbf{d}_t^{(k+1)} - \tilde{\mathbf{d}}_t + \mathbf{u}_{\mathbf{d}_t}^{(k)}\|_2^2, \tag{C.8}$$

$$\mathbf{u}_{\mathbf{d}_t}^{(k+1)} = \mathbf{u}_{\mathbf{d}_t}^{(k)} + \mathbf{G}^* \mathbf{d}_t^{(k+1)} - \tilde{\mathbf{d}}_t^{(k+1)}. \tag{C.9}$$

Equations (C.7), (C.8) and (C.9) correspond respectively to expressions (B.2), (B.3) and (B.4) in appendix B.

We comment now on the resolution of (C.7) and (C.8). First, the unconstrained differentiable problem (C.7) can be easily solved via a gradient descent algorithm. The gradient of the cost function in (C.7) with respect to $\mathbf{d}_t$ can be expressed as

$$\nabla_{\mathbf{d}_t} \mathcal{B}(\mathbf{d}^{(\ell)}) + \alpha^{(\ell)}(\mathbf{d}_t - \mathbf{d}_t^{(\ell)}) + \rho \mathbf{G}(\mathbf{G}^* \mathbf{d}_t - \tilde{\mathbf{d}}_t^{(k)} + \mathbf{u}_{\mathbf{d}_t}^{(k)}). \tag{C.10}$$

As mentioned previously, $\nabla_{\mathbf{d}_t} \mathcal{B}(\mathbf{d}^{(\ell)})$ is simple to evaluate via the recursions described in section 4.1; moreover, the multiplications by $\mathbf{G}$ and $\mathbf{G}^*$ appearing in the last term of (C.10) can be done efficiently for the particular choice of $\mathbf{G}$ considered in this paper (see section C.1 for details on this topic).

Second, the solution of problem (C.8) is closed-form (see *e.g.*, [36, section 6.5.2]). It is given for any $j \in \mathcal{S}_i$ with $1 \leq i \leq n$ by

$$\tilde{\mathbf{d}}_t^{(k+1)}(j) = \begin{cases} 0 \quad \text{if } \tau_i \leq \alpha_2 \mathbf{w}(i)/\rho, \\ \dfrac{(\tau_i - \alpha_2 \mathbf{w}(i)/\rho)}{\tau_i} (\mathbf{g}_j^* \mathbf{d}_t^{(k+1)} + \mathbf{u}_{\mathbf{d}_t}^{(k)}(j)) \quad \text{otherwise,} \end{cases}$$

where the scalar τ_i is given by

$$\tau_i = \sqrt{\sum_{j \in \mathcal{S}_i} \left(\mathbf{g}_j^* \mathbf{d}_t^{(k+1)} + \mathbf{u}_{\mathbf{d}_t}^{(k)}(j) \right)^2}.$$

REFERENCES

[1] Akgun, T., Altunbasak, Y., Mersereau, R.M.: Super-resolution reconstruction of hyperspectral images. Image Processing, IEEE Transactions on **14**(11), 1860–1875 (2005)

[2] Attouch, H., Bolte, J., Redont, P., Soubeyran, A.: Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the Kurdyka-Lojasiewicz inequality. Math. Oper. Res. **35**(2), 438–457 (2010)

[3] Attouch, H., Bolte, J., Svaiter, B.: Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward-backward splitting, and regularized Gauss-Seidel methods. Mathematical Programming **137**(1-2), 91–129 (2013)

[4] Baker, S., Kanade, T.: Super-resolution optical flow. Tech. rep., Carnegy Mellon University (1999)

[5] Baker, S., Scharstein, D., Lewis, J.P., Roth, S., Black, M., Szeliski, R.: A database and evaluation methodology for optical flow. International Journal of Computer Vision **92**(1), 1–31 (2011)

[6] Bertsekas, D.P.: Nonlinear Programming, 2nd edn. Athena Scientific (1999)

[7] Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends in Machine Learning **3**(1), 1–122 (2011)

[8] Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Found. Trends Mach. Learn. **3**(1), 1–122 (2011)
[9] Burt, P.J.: Fast filter transform for image processing. Computer Graphics and Image Processing **16**(1), 20 – 51 (1981)
[10] Butler, D.J., Wulff, J., Stanley, G.B., Black, M.J.: A naturalistic open source movie for optical flow evaluation. In: A. Fitzgibbon et al. (Eds.) (ed.) European Conf. on Computer Vision, Part IV, LNCS 7577, pp. 611–625. Springer-Verlag (2012)
[11] Candès, E., Demanet, L., Donoho, D., Ying, L.: Fast discrete curvelet transforms. SIAM Multiscale Model Simul **5**(3), 861–899 (2006)
[12] Candès, E.J., Donoho, D.L.: New tight frames of curvelets and optimal representations of objects with piecewise $\mathcal{C}_2$ singularities. Comm. on Pure and Appl. Math. pp. 219–266 (2002)
[13] Capel, D., Zisserman, A.: Super-resolution from multiple views using learnt image models. In: Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, vol. 2, pp. II–627–II–634 vol.2. IEEE (2001)
[14] Costa, G.H., Bermudez, J.C.M.: On the design of the LMS algorithm for robustness to outliers in super-resolution video reconstruction. In: Image Processing, 2006 IEEE International Conference on, pp. 1737–1740. IEEE (2006)
[15] Costa, G.H., Bermudez, J.C.M.: Statistical analysis of the LMS algorithm applied to super-resolution image reconstruction. Signal Processing, IEEE Transactions on **55**(5), 2084–2095 (2007)
[16] Costa, G.H., Bermudez, J.C.M.: Registration errors: Are they always bad for super-resolution? Signal Processing, IEEE Transactions on **57**(10), 3815–3826 (2009)
[17] De Santis, P., Gori, F.: On an iterative method for super-resolution. Optica Acta: International Journal of Optics **22**(8), 691–695 (1975)
[18] Eckstein, J., Bertsekas, D.P.: On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. Math. Program. **55**(3), 293–318 (1992)
[19] Elad, M., Feuer, A.: Super-resolution reconstruction of image sequences. Pattern Analysis and Machine Intelligence, IEEE Transactions on **21**(9), 817–834 (1999)
[20] Elad, M., Feuer, A.: Superresolution restoration of an image sequence: adaptive filtering approach. Image Processing, IEEE Transactions on **8**(3), 387–395 (1999)
[21] Elad, M., Starck, J.L., Querre, P., Donoho, D.L.: Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA). Applied and Computational Harmonic Analysis **19**(3), 340–358 (2005)
[22] Elad, P., Feuer, A.: Super-resolution restoration of continuous image sequence using the LMS algorithm. In: Electrical and Electronics Engineers in Israel, 1995., Eighteenth Convention of, pp. 2.2.5/1–2.2.5/5. IEEE (1995)
[23] Farsiu, S., Elad, M., Milanfar, P.: Video-to-video dynamic super-resolution for grayscale and color sequences. EURASIP Journal on Advances in Signal Processing **2006**(1), 061,859+ (2006)
[24] Farsiu, S., Robinson, M.D., Elad, M., Milanfar, P.: Fast and robust multiframe super resolution. Image Processing, IEEE Transactions on **13**(10), 1327–1344 (2004)
[25] Foucart, S., Rauhut, H.: A mathematical introduction to compressive sensing. Applied and Numerical Harmonic Analysis. Birkhaüser (2013)
[26] Fransens, R., Strecha, C., Van Gool, L.: Optical flow based super-resolution: A probabilistic approach. Computer Vision and Image Understanding **106**(1), 106–115 (2007)
[27] Gerchberg, R.W.: Super-resolution through error energy reduction. Optica Acta: International Journal of Optics **21**(9), 709–720 (1974)
[28] Hardie, R.: A fast image super-resolution algorithm using an adaptive Wiener filter. Image Processing, IEEE Transactions on **16**(12), 2953–2964 (2007)
[29] Hardie, R.C., Barnard, K.J., Bognar, J.G., Armstrong, E.E., Watson, E.A.: High-resolution image reconstruction from a sequence of rotated and translated frames and its application to an infrared imaging system. Optical Engineering **37**(1), 247–260 (1998)
[30] Keller, S.H., Lauze, F., Nielsen, M.: Video super-resolution using simultaneous motion and intensity calculations. Image Processing, IEEE Transactions on **20**(7), 1870–1884 (2011)
[31] Lemarié-Rieusset, P.: Analyses multirésolutions non orthogonales, commutation entre projecteurs et dérivation et ondelettes vecteurs à divergence nulle. Revista Matematica Iberoamericana **8**, 221–237 (1992)
[32] Mallat, S.: A Wavelet Tour of Signal Processing: The Sparse Way. Academic Press (2008)
[33] Mendel, J.M.: Lessons in Estimation Theory for Signal Processing Communications and Control. Prentice Hall Signal Processing Series, Englewood Cliffs, NJ (1995)

[34] Nasrollahi, K., Moeslund, T.: Super-resolution: a comprehensive survey **25**(6), 1423–1468 (2014)
[35] Nocedal, J., Wright, S.J.: Numerical Optimization. Springer Series in Operations Research. Springer-Verlag, New York (1999)
[36] Parikh, N., Boyd, S.: Proximal algorithms. Foundations and Trends in Optimization **1**(3), 127–239 (2014)
[37] Patti, A.J., Sezan, M.I., Tekalp, A.M.: Superresolution video reconstruction with arbitrary sampling lattices and nonzero aperture time. Image Processing, IEEE Transactions on **6**(8), 1064–1076 (1997)
[38] Peleg, S., Keren, D., Schweitzer, L.: Improving image resolution using subpixel motion. Pattern Recogn. Lett. **5**(3), 223–226 (1987)
[39] Peyré, G.: Sparse modeling of textures. Journal of Mathematical Imaging and Vision **34**(1), 17–31 (2009)
[40] Purkait, P., Chanda, B.: Super resolution image reconstruction through bregman iteration using morphologic regularization. Image Processing, IEEE Transactions on **21**(9), 4029–4039 (2012)
[41] Puy, G., Vandergheynst, P.: Robust image reconstruction from multiview measurements. SIAM J. Imaging Sci. **7**(1), 128–156 (2014)
[42] Schatzberg, A., Devaney, A.J.: Super-resolution in diffraction tomography. Inverse Problems **8**(1), 149+ (1992)
[43] Schultz, R.R., Stevenson, R.L.: Extraction of high-resolution frames from video sequences. Image Processing, IEEE Transactions on **5**(6), 996–1011 (1996)
[44] Tom, B.C., Katsaggelos, A.K., Galatsanos, N.P.: Reconstruction of a high resolution image from registration and restoration of low resolution images. In: Image Processing, 1994. Proceedings. ICIP-94., IEEE International Conference, vol. 3, pp. 553–557 vol.3. IEEE (1994)
[45] Tsai, R., Huang, T.: Multiframe image restoration and registration. Advances in Computer Vision and Image Processing **1**, 317–339 (1984)
[46] Turkowski, K.: Graphics gems. chap. Filters for Common Resampling Tasks, pp. 147–165. Academic Press Professional, Inc., San Diego, CA, USA (1990)
[47] Unser, M., Aldroubi, A., Eden, M.: Fast B-Spline transforms for continuous image representation and interpolation. IEEE Transactions on Pattern Analysis and Machine Intelligence **13**(3), 277–285 (1991)
[48] Wang, Z., Qi, F.: Super-resolution video restoration with model uncertainties. In: Image Processing. 2002. Proceedings. 2002 International Conference on, vol. 2, pp. II–853–II–856 vol.2. IEEE (2002)
[49] Weickert, J., Schnörr, C.: A theoretical framework for convex regularizers in pde-based computation of image motion. Int. J. Computer Vision pp. 245–264 (2004)
[50] Xu, L., Jia, J., Matsushita, Y.: Motion detail preserving optical flow estimation. IEEE Transactions on Pattern Analysis and Machine Intelligence **34**(9), 1744–1757 (2012)
[51] Yang, J., Schonfeld, D.: New results on performance analysis of super-resolution image reconstruction. In: Image Processing (ICIP), 2009 16th IEEE International Conference on, pp. 1517–1520. IEEE (2009)
[52] Yang, J., Wright, J., Huang, T.S., Ma, Y.: Image super-resolution via sparse representation. Image Processing, IEEE Transactions on **19**(11), 2861–2873 (2010)
[53] Yuan, Q., Zhang, L., Shen, H.: Multiframe super-resolution employing a spatially weighted total variation model. Circuits and Systems for Video Technology, IEEE Transactions on **22**(3), 379–392 (2012)
[54] Zach, C., Pock, T., Bischof, H.: A duality based approach for realtime TV-ℓ_1 optical flow. In: In Ann. Symp. German Association Patt. Recogn, pp. 214–223 (2007)
[55] Zhao, W., Sawhney, H.: Is super-resolution with optical flow feasible? In: A. Heyden, G. Sparr, M. Nielsen, P. Johansen (eds.) Computer Vision - ECCV 2002, Lecture Notes in Computer Science, vol. 2350, chap. 40, pp. 599–613. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)

FIG. 1. ***Data set****: first and last frame of the "Bandage 2 " sequence.*

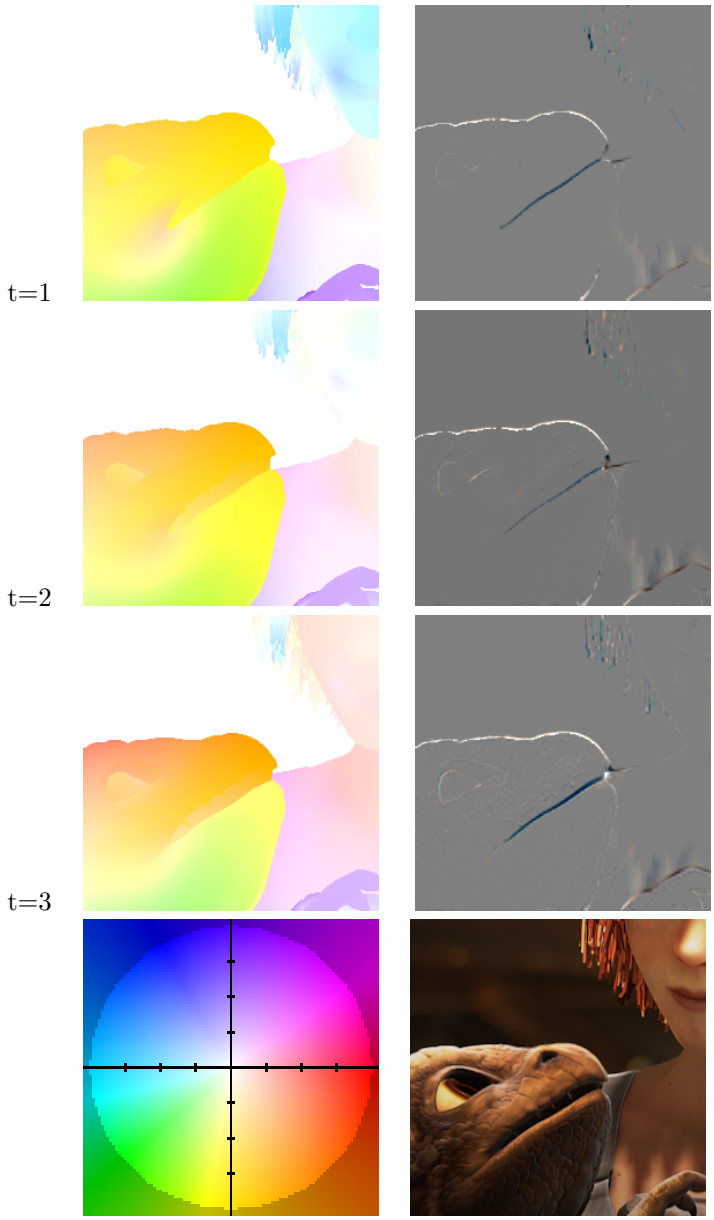

FIG. 2. ***Sample from the "Bandage 2" sequence. 1-st column****: from up to down, motion at t=1, 2, 3 and motion colormap.* ***2-nd column****: from up to down, interpolation error at t=1, 2, 3 and last frame from the sequence of 4 images.*

FIG. 3. ***Comparison of algorithms 2 and 4 with SR from a single image (zoom on the girl's face). 1-st column****: from up to down, subsampled* $\times 2$ *image, Lanczos interpolate, SR* $\times 2$ *with Algorithm 3 using a single image.* ***2-nd column****: from up to down, SR using a sequence of 4 images with Algorithm 2 and Algorithm 4 and ground truth.*

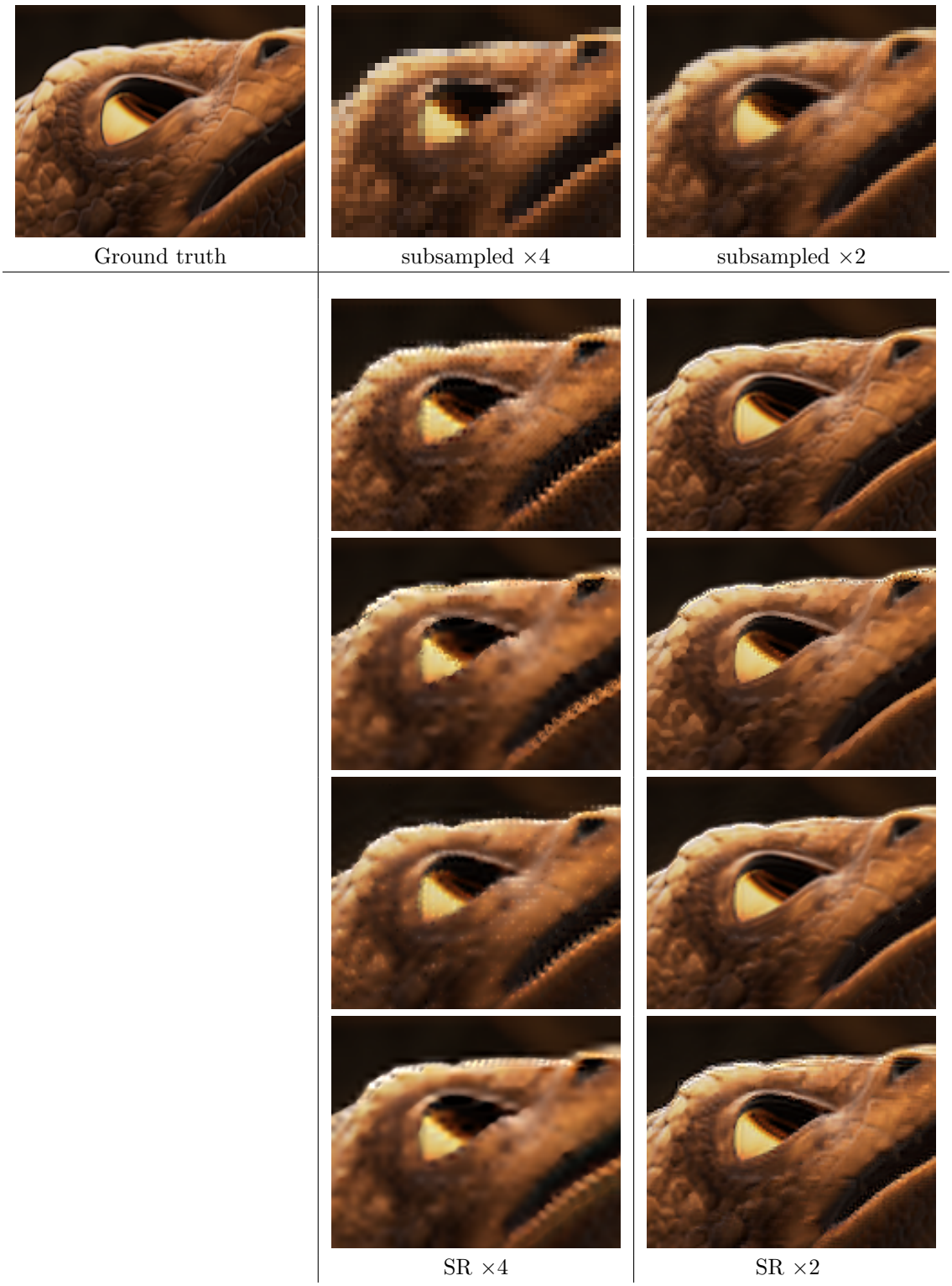

FIG. 4. ***Visual comparison of the 4 algorithms (zoom on the little dragon).*** *1-st row: Ground truth (left) and low resolution image down-sampling factors equal to 4 (middle) and 2 (right). From the 2-nd to the 5-th row: reconstructions obtained with Algorithms 1, 2, 3 and 4 for down-sampling factors equal to* 4 *(middle) and* 2 *(right).*

FIG. 5. ***Reconstruction of the HR images for increasing sequence length (zoom on the little dragon).*** *From left to right, SR* $\times 4$ *reconstruction using Algorithm 4 with* $T = 2$, 3 *and* 4.

a b

c d

FIG. 6. ***Visual comparison of Algorithms 1 (a), 2 (b), 3 (c), and 4 (d) on a large region of interest and with a down-sampling factor equal to 2.***

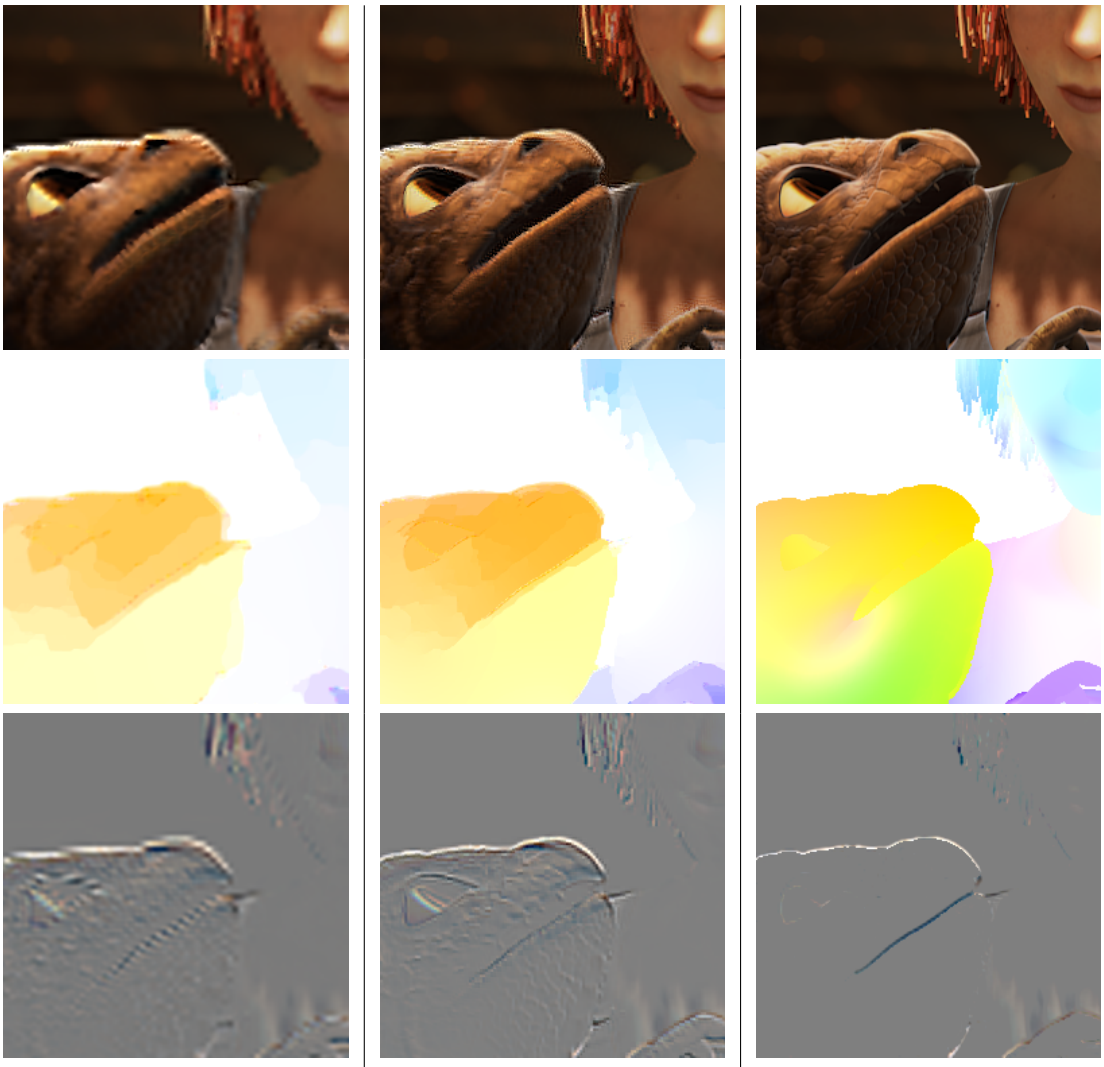

FIG. 7. ***Reconstruction of the HR image, motion and interpolation error with Algorithm 4. 1-st row:*** *From left to right, reconstructed image* $\hat{\mathbf{x}}_0$ *with down-sampling factor* 4 *(left), down-sampling factor* 2 *(middle) and original HR image (right) corresponding to the first frame of the sequence (*$\mathbf{x}_0$*).* ***2-nd row:*** *Associated reconstructed motion fields* $\hat{\mathbf{d}}_1$ *and ground truth* $\mathbf{d}_1$*.* ***3-rd row:*** *Associated reconstructed interpolation errors* $\hat{\boldsymbol{\epsilon}}_1$ *and ground truth* $\boldsymbol{\epsilon}_1$*.*