2016 ICSEE International Conference on the Science of Electrical Engineering

Reducing Artifacts of Intra-Frame Video Coding via Sequential Denoising

Yehuda Dar, Alfred M. Bruckstein, Michael Elad
Department of Computer Science
Technion – Israel Institute of Technology

Raja Giryes
School of Electrical Engineering
Tel-Aviv University

***Abstract*—In this work we propose a new postprocessing method for video sequences compressed using intra-frame coding techniques. The suggested method extends our previously published approach for handling compressed still-images. We rely on the Plug-and-Play Prior framework, which shows that a general inverse problem can be cast as a sequence of Gaussian denoising steps. We formulate the video recovery task as such an inverse problem, with a regularization that leverages on existing state-of-the-art *video denoising* algorithms. Our method's strength emerges from two origins: (i) the flexibility of using the best available video denoising algorithm; and (ii) the fact that, while intra-coding is treated, an inter-frame force is introduced via the denoising stage. As such, our scheme can be interpreted as belonging to the distributed video coding paradigm with an extended decompression procedure coupled with a relatively simple compression. A prominent part in our approach is a linearization of the nonlinear compression-decompression operation, while leveraging the intra-coding structure to obtain a block-diagonal matrix form. We demonstrate significant quality improvements for video sequences compressed using Motion-JPEG2000.**

I. Introduction

Lossy compression is a widely used approach for representing a signal under bit-budget constraints while allowing some errors in the reconstruction. Typically, artifacts are introduced as part of the inaccurate recovery of the signal. The artifact type stems from the compression architecture, e.g., block-based image/video coding results in blockiness that becomes more visible as the bit-rate reduces. Consequently, many artifact-reduction techniques were proposed over the years, usually considering specific signal and/or artifact types (e.g., image deblocking techniques [1], [2]).

In our previous work [3], we proposed a novel postprocessing technique for compression artifact reduction by a regularized restoration of the original (precompressed) signal. Specifically, we formulated the compression postprocessing procedure as a regularized inverse-problem for estimating the original signal given its decompressed form. We also approximated the nondifferentiable and nonlinear compression-decompression process by a linear operator, so as to obtain a tractable inverse problem formulation. This interesting approach of locally linearizing the nondifferentiable compression procedures was thoroughly analyzed in [3].

Whereas many studies focused on corrections of specific artifacts, our approach attempts to generally restore the signal and thus implicitly repairs multiple artifacts. The major strength of our method comes from the regularization used. It relies on the ”Plug-and-Play Priors” framework [4], where the alternating direction method of multipliers (ADMM) [5] is utilized to efficiently solve regularized inverse problems by decoupling the inversion and the regularization parts of the optimization problem. The Plug and Play approach relies on the equivalence between the regularization step and an additive Gaussian denoising optimization. This framework is flexible and proposes the replacement of the regularization step by a general-purpose Gaussian image denoiser. As mentioned above, in [3] we have developed, using this strategy, a post-processing algorithm for compressed images.

In this work we extend our work in [3] and propose a postprocessing technique for video signals compressed using intra-frame coding methods, where each frame is coded independently of the rest of the sequence. Our proposed method is iterative – in each step it solves an optimization problem that involves the compression-decompression operator, and applies a Gaussian video denoiser. As video frames are separately coded, the treatment of the compression-decompression operator elegantly reduces to individually considering single frames, and thus, the corresponding optimization can be efficiently solved. Contrastingly, the spatio-temporal structure of the signal is utilized by incorporating an efficient video Gaussian-denoiser (such as V-BM4D [6] or BM4D [7]), which is applied on frame groups.

The distributed coding approach suggests to separately encode dependent data elements, and to compensate this suboptimality via a complicated decoder that utilizes the inter-component relations by jointly reconstructing them. Accordingly, the low-complexity intra-frame video encoder together with the extended decoder, defined as the intra-frame decoder followed by the proposed spatio-temporal postprocessing, constitute an elementary distributed video-coding system [8].

*The research leading to these results has received funding from the European Research Council under European Unions Seventh Framework Program, ERC Grant agreement no. 320649.

*E-mail addresses:
{ydar,freddy,elad}@cs.technion.ac.il, raja@tauex.tau.ac.il.

II. The Proposed Postprocessing Method

A. The Video Signal and Intra-Frame Coding

Let us consider a video signal consisting of T frames, each has a spatial resolution of W pixel width and H pixel height. Accordingly, the column-stack form of the signal is denoted here as $\mathbf{x} \in \mathbb{R}^N$, where $N = T \cdot W \cdot H$ is the total number of samples in the signal. The signal $\mathbf{x}$ is, in fact, a vertical concatenation of the column-stack form of its T frames, i.e.,

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}^{(1)} \\ \vdots \\ \mathbf{x}^{(T)} \end{bmatrix} \tag{1}$$

where $\mathbf{x}^{(i)} \in \mathbb{R}^{N_f}$ is the column-stack form of the i^{th} frame ($i = 1, ..., T$), and $N_f = W \cdot H$ is the number of pixels in a single frame.

The video signal $\mathbf{x}$ undergoes a compression-decompression procedure, $C : \mathbb{R}^N \rightarrow \mathbb{R}^N$, resulting in the reconstructed signal $\mathbf{y} = C(\mathbf{x})$. More specifically, we examine here an intra-frame coding procedure, where each frame is separately compressed-decompressed via the same procedure, $C_f : \mathbb{R}^{N_f} \rightarrow \mathbb{R}^{N_f}$. Accordingly, the reconstructed video satisfies

$$\mathbf{y} = C(\mathbf{x}) = \begin{bmatrix} C_f\left(\mathbf{x}^{(1)}\right) \\ \vdots \\ C_f\left(\mathbf{x}^{(T)}\right) \end{bmatrix}. \tag{2}$$

B. Problem Formulation using ADMM

For lossy compression methods an error is introduced of magnitude that depends on the bit-budget, the specific-signal characteristics, and the compression algorithm. We aim at restoring the precompressed signal $\mathbf{x}$ from the reconstruction $\mathbf{y}$ using the following regularized inverse-problem:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \; \|\mathbf{y} - C(\mathbf{x})\|_2^2 + \beta s(\mathbf{x}), \tag{3}$$

where $s(\cdot)$ is a regularizer, which can be associated with a given Gaussian denoiser, weighted by the parameter β.

Similar to [4] and [9], we develop an iterative algorithm for the solution of (3). We start by applying variable splitting that yields the following equivalent form of (3):

$$\min_{\mathbf{x},\mathbf{v}} \; \|\mathbf{y} - C(\mathbf{x})\|_2^2 + \beta s(\mathbf{v}) \quad \text{s.t.} \quad \mathbf{x} = \mathbf{v}, \tag{4}$$

where $\mathbf{v} \in \mathbb{R}^N$ is an additional vector due to the split. The constrained problem (4) is addressed by forming an augmented Lagrangian and its corresponding iterative solution (of its scaled version) via the method of multipliers [5, ch. 2], where the i^{th} iteration consists of

$$\begin{aligned} (\hat{\mathbf{x}}_i, \hat{\mathbf{v}}_i) = \arg\min_{\mathbf{x},\mathbf{v}} \|\mathbf{y} - C(\mathbf{x})\|_2^2 + \beta s(\mathbf{v}) \\ + \frac{\lambda}{2} \|\mathbf{x} - \mathbf{v} + \mathbf{u}_i\|_2^2 \\ \mathbf{u}_{i+1} = \mathbf{u}_i + (\hat{\mathbf{x}}_i - \hat{\mathbf{v}}_i), \end{aligned} \tag{5}$$

where $\mathbf{u}_i \in \mathbb{R}^n$ is the scaled dual-variable and λ is an auxiliary parameter, both introduced in the Lagrangian.

Please note the following notation remark for a general vector $\mathbf{u}$. Whereas $\mathbf{u}_i$ stands for vector $\mathbf{u}$ in the i^{th} iteration, u_j represents the j^{th} component (a scalar) of the vector $\mathbf{u}$. In addition, $\mathbf{u}^{(j)}$ denotes the j^{th} frame of a video signal $\mathbf{u}$ and, accordingly, $\mathbf{u}_i^{(j)}$ denotes the j^{th} frame of the video signal $\mathbf{u}_i$.

Approximating the joint optimization of $\mathbf{x}$ and $\mathbf{v}$ in (5), using one iteration of alternating minimization, results in the iterative solution in the ADMM form, where the i^{th} iteration consists of

$$\hat{\mathbf{x}}_i = \arg\min_{\mathbf{x}} \|\mathbf{y} - C(\mathbf{x})\|_2^2 + \frac{\lambda}{2} \|\mathbf{x} - \tilde{\mathbf{x}}_i\|_2^2 \tag{6}$$

$$\hat{\mathbf{v}}_i = \arg\min_{\mathbf{v}} \frac{\lambda}{2} \|\mathbf{v} - \tilde{\mathbf{v}}_i\|_2^2 + \beta s(\mathbf{v}) \tag{7}$$

$$\mathbf{u}_{i+1} = \mathbf{u}_i + (\hat{\mathbf{x}}_i - \hat{\mathbf{v}}_i). \tag{8}$$

Here $\tilde{\mathbf{x}}_i = \hat{\mathbf{v}}_{i-1} - \mathbf{u}_i$ and $\tilde{\mathbf{v}}_i = \hat{\mathbf{x}}_i + \mathbf{u}_i$.

The regularization step (7) is of the form of a Gaussian denoising optimization-problem (of a noise level determined by β/λ) and therefore can be viewed as applying a denoising algorithm to the signal $\tilde{\mathbf{v}}_i$. Indeed, the Plug-and-Play Priors framework [4] suggests exactly this strategy, replacing (7) with an independent denoiser; even one that does not explicitly have in its formulation a minimization problem of the form of (7), i.e., it replaces (7) with a denoising operation, $\hat{\mathbf{v}}_i = Denoise_{\beta/\lambda}(\tilde{\mathbf{v}}_i)$. The deployment of a favorable denoiser introduces valuable practical benefits to the design of the proposed postprocessing procedure, and yields a powerful generic method. Moreover, by choosing an effective video denoiser that utilizes the inter-frame relations, our postprocessing becomes a spatio-temporal procedure attempting to repair the inefficiency of the intra-frame encoder.

C. Linear Approximation of the Intra-Frame Compression-Decompression Procedure

Due to the high nonlinearity of $C(\mathbf{x})$, we further simplify the forward-model step (6) using a first-order Taylor approximation of the compression-decompression function around $\hat{\mathbf{x}}_{i-1}$, i.e.,

$$C_{lin}(\mathbf{x}) = C(\hat{\mathbf{x}}_{i-1}) + \left.\frac{dC(\mathbf{z})}{d\mathbf{z}}\right|_{\mathbf{z}=\hat{\mathbf{x}}_{i-1}} \cdot (\mathbf{x} - \hat{\mathbf{x}}_{i-1}) \tag{9}$$

where $\left.\frac{dC(\mathbf{z})}{d\mathbf{z}}\right|_{\mathbf{z}=\hat{\mathbf{x}}_{i-1}}$ is the $N \times N$ Jacobian matrix of the compression-decompression at the point $\hat{\mathbf{x}}_{i-1}$.

Since the approximation of the Jacobian, $\frac{dC(\mathbf{z})}{d\mathbf{z}}$, deeply influences the restoration result and the computational cost, this is a delicate task. First, C is a nonlinear and even nondifferentiable function as the compression relies on quantization and/or thresholding. Second, we provide here a generic technique, and therefore do not explicitly consider the compression-decompression formulation except from utilizing its intra-frame coding structure.

For calculating the entries of the Jacobian, we rely on the standard definition of the derivative, assuming that C is locally linear. We justify this approach via a mathematical analysis provided in [3]. As we might be approximating the derivative in the neighborhood of a nondifferential point, we take several step-sizes in the calculation of the derivative and average over all of them. This leads to the following approximation to the k^{th} column of the Jacobian:

$$\frac{dC(\mathbf{z})}{dz_k} = \frac{1}{|S_\delta|} \sum_{\delta \in S_\delta} \frac{C(\mathbf{z} + \delta \cdot \mathbf{e}_k) - C(\mathbf{z} - \delta \cdot \mathbf{e}_k)}{2\delta}, \quad (10)$$

where $\mathbf{e}_k$ is the k^{th} standard direction vector, and S_δ is a set of step lengths for approximating the derivative using the standard definition (the set size is denoted as $|S_\delta|$).

The straightforward computation of the Jacobian according to (10) is costly, especially for video signals, as it heavily depends on the total number of samples, N. Particularly, the Jacobian matrix has N columns, each should be computed using (10) via $2 \cdot |S_\delta|$ compression-decompression applications. In addition, the Jacobian matrix of C is of $N \times N$ size and, thus, may also impose practical difficulties in solving (6).

Fortunately, the intra-frame coding procedure has the form in (2), where the frames are individually encoded-decoded using the procedure C_f. Consequently, the Jacobian matrix has the block-diagonal form of

$$\left.\frac{dC(\mathbf{z})}{d\mathbf{z}}\right|_{\mathbf{z}=\hat{\mathbf{x}}_{i-1}} = \quad (11)$$
$$\begin{bmatrix} \left.\frac{dC_f(\mathbf{w})}{d\mathbf{w}}\right|_{\mathbf{w}=\hat{\mathbf{x}}_{i-1}^{(1)}} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \left.\frac{dC_f(\mathbf{w})}{d\mathbf{w}}\right|_{\mathbf{w}=\hat{\mathbf{x}}_{i-1}^{(2)}} & \mathbf{0} & \vdots \\ \vdots & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \left.\frac{dC_f(\mathbf{w})}{d\mathbf{w}}\right|_{\mathbf{w}=\hat{\mathbf{x}}_{i-1}^{(T)}} \end{bmatrix}$$

where $\left.\frac{dC_f(\mathbf{w})}{d\mathbf{w}}\right|_{\mathbf{w}=\hat{\mathbf{x}}_{i-1}^{(j)}}$ is the $N_f \times N_f$ Jacobian matrix of the frame compression-decompression procedure, C_f, around the N_f-dimensional point, $\hat{\mathbf{x}}_{i-1}^{(j)}$, that corresponds to the j^{th} frame of $\hat{\mathbf{x}}_{i-1}$. The block-diagonal form in (11) lets us to concurrently compute T Jacobian columns, each belonging to a different video frame. Namely, instead of using (10) for $\mathbf{z} = \hat{\mathbf{x}}_{i-1}$, we employ the following relation:

$$\begin{bmatrix} \left.\frac{dC_f(\mathbf{w})}{dw_k}\right|_{\mathbf{w}=\mathbf{z}^{(1)}} \\ \vdots \\ \left.\frac{dC_f(\mathbf{w})}{dw_k}\right|_{\mathbf{w}=\mathbf{z}^{(T)}} \end{bmatrix} = \frac{1}{|S_\delta|} \times \quad (12)$$
$$\sum_{\delta \in S_\delta} \frac{C\left(\mathbf{z} + \delta \sum_{j=0}^{T-1} \mathbf{e}_{(jN_f+k)}\right) - C\left(\mathbf{z} - \delta \sum_{j=0}^{T-1} \mathbf{e}_{(jN_f+k)}\right)}{2\delta},$$

where $\mathbf{e}_{(jN_f+k)}$ is the $(jN_f+k)^{th}$ standard direction vector, and $\left.\frac{dC_f(\mathbf{w})}{dw_k}\right|_{\mathbf{w}=\mathbf{z}^{(j)}}$ is the k^{th} column ($k = 1, ..., N_f$) of the Jacobian matrix corresponding to encoding-decoding the j^{th} frame of the video signal $\mathbf{z}$. Accordingly, the complete Jacobian matrix can be formed, for $\mathbf{z} = \hat{\mathbf{x}}_{i-1}$, following the structure in (11).

Due to the high nonlinearity of C, the linear approximation (9) is reasonable in a small neighborhood around the approximating point $\hat{\mathbf{x}}_{i-1}$. Accordingly, we further constrain the distance of the solution from the linear-approximation point by modifying (6) to

$$\hat{\mathbf{x}}_i = \arg\min_{\mathbf{x}} \|\mathbf{y} - C_{lin}(\mathbf{x})\|_2^2 \quad (13)$$
$$+ \frac{\lambda}{2}\|\mathbf{x} - \tilde{\mathbf{x}}_i\|_2^2 + \mu \|\mathbf{x} - \hat{\mathbf{x}}_{i-1}\|_2^2 .$$

The intra-frame coding structure in (2) and the corresponding block-diagonal form of its Jacobian matrix (11) allow us to decompose the linear form given in (9) for the entire video signal, and to write

$$C_{lin}(\mathbf{x}) = \begin{bmatrix} C_{f,lin}\left(\mathbf{x}^{(1)}\right) \\ \vdots \\ C_{f,lin}\left(\mathbf{x}^{(T)}\right) \end{bmatrix} \quad (14)$$

where $C_{f,lin}$ is the linear approximation of the frame compression-decompression procedure, formulated for the j^{th} frame ($j = 1, ..., T$) of the video $\mathbf{x}$ as

$$C_{f,lin}\left(\mathbf{x}^{(j)}\right) = \quad (15)$$
$$C_f\left(\hat{\mathbf{x}}_{i-1}^{(j)}\right) + \left.\frac{dC_f(\mathbf{w})}{d\mathbf{w}}\right|_{\mathbf{w}=\hat{\mathbf{x}}_{i-1}^{(j)}} \cdot \left(\mathbf{x}^{(j)} - \hat{\mathbf{x}}_{i-1}^{(j)}\right).$$

Accordingly, the optimization problem in (13), defined for the entire video signal, reduces to a set of distinct optimizations, each considering a single frame. More specifically, the problem of the j^{th} frame ($j = 1, ..., T$) is formulated as

$$\hat{\mathbf{x}}_i^{(j)} = \arg\min_{\mathbf{x}^{(j)}} \left\|\mathbf{y} - C_{f,lin}\left(\mathbf{x}^{(j)}\right)\right\|_2^2 \quad (16)$$
$$+ \frac{\lambda}{2}\left\|\mathbf{x}^{(j)} - \tilde{\mathbf{x}}_i^{(j)}\right\|_2^2 + \mu \left\|\mathbf{x}^{(j)} - \hat{\mathbf{x}}_{i-1}^{(j)}\right\|_2^2 .$$

The results of the frame-level optimizations are concatenated to form the entire spatio-temporal solution $\hat{\mathbf{x}}_i$.

The proposed method for postprocessing intra-frame coded video signals is summarized in Algorithm 1.

D. Further Reduction of the Computational Complexity

The computational complexity of our method is mainly determined by the complexity levels of the utilized denoiser and the Jacobian estimation procedure. The latter further depends on the implementation of the compression-decompression method, as it is repeatedly applied according to (12). In addition, Equation (12) exhibits also the effect of the size of the set S_δ utilized for jointly approximating T columns of the Jacobian of the video compression-decompression procedure, C. Since the number of columns in the Jacobian of the frame compression-decompression procedure, C_f, is the number of frame samples (denoted as N_f), computation of the Jacobian in (11) may still be costly, as it requires N_f calculations of (12).

Algorithm 1 The Proposed Postprocessing Method

```
1: $\hat{\mathbf{x}}_0 = \mathbf{y}$ , $\hat{\mathbf{v}}_0 = \mathbf{y}$
2: $i = 1$, $\mathbf{u}_1 = \mathbf{0}$
3: repeat
4:     Approximate $C_{f,lin}(\cdot)$ around $\hat{\mathbf{x}}_{i-1}^{(j)}$ ($j = 1, ..., T$) using (15) and (12)
5:     $\tilde{\mathbf{x}}_i = \hat{\mathbf{v}}_{i-1} - \mathbf{u}_i$
6:     Form $\hat{\mathbf{x}}_i$ by solving for $j = 1, ..., T$:
       $\hat{\mathbf{x}}_i^{(j)} = \arg\min_{\mathbf{x}^{(j)}} \left\| \mathbf{y} - C_{f,lin}\left(\mathbf{x}^{(j)}\right) \right\|_2^2 + \frac{\lambda}{2} \left\| \mathbf{x}^{(j)} - \tilde{\mathbf{x}}_i^{(j)} \right\|_2^2 + \mu \left\| \mathbf{x}^{(j)} - \hat{\mathbf{x}}_{i-1}^{(j)} \right\|_2^2$
7:     $\tilde{\mathbf{v}}_i = \hat{\mathbf{x}}_i + \mathbf{u}_i$
8:     $\hat{\mathbf{v}}_i = \text{Denoise}_{\beta/\lambda}\left(\tilde{\mathbf{v}}_i\right)$
9:     $\mathbf{u}_{i+1} = \mathbf{u}_i + \left(\hat{\mathbf{x}}_i - \hat{\mathbf{v}}_i\right)$
10:    $i \leftarrow i + 1$
11: until stopping criterion is satisfied
```

Fortunately, the computational requirements for estimating the Jacobian matrix can be further relaxed for many compression methods that operate independently on non-overlapping blocks within each frame. In this case, the Jacobian of the frame compression-decompression procedure, $\frac{dC_f(\mathbf{w})}{d\mathbf{w}}$, becomes a block-diagonal matrix and, therefore, its columns can be arranged in independent subsets for concurrent computation. This reduces the number of compression-decompression applications to the order of the block size. Moreover, the block-diagonal structure of the Jacobian allows to decompose the computation of (16) to handle each spatial-block separately. The details of this computational simplification are inherited from the discussion given above for linearization of separately coded frames.

In addition, the block-diagonal structure of $\frac{dC_f(\mathbf{w})}{d\mathbf{w}}$ can be assumed even for compression methods that do not conform with it (e.g., Motion-JPEG2000), and thus somewhat compromising the postprocessing result, in order to offer a reasonable run-time. The (possibly assumed) spatial-block size of the compression procedure is denoted here as $B_H \times B_W$, and yields a Jacobian with blocks of size $B_H B_W \times B_H B_W$ along its main diagonal.

III. Experimental Results

In this section we demonstrate the performance of the proposed postprocessing method by presenting results obtained in conjunction with the well-known Motion-JPEG2000 standard [10]. Motion-JPEG2000 is an intra-frame coding method that naturally extends the JPEG2000 still-image compression standard [11] to video signals. More specifically, each frame in the sequence is independently encoded using a wavelet-based transform coding procedure, applied on large spatial tiles (of at least 128×128 pixels).

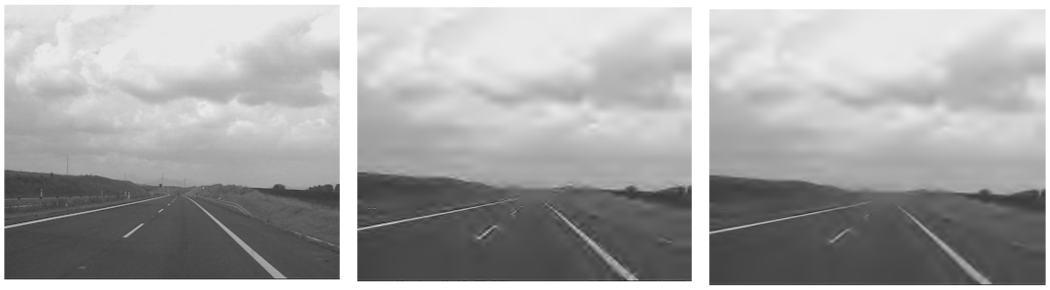
We use the BM4D method [7] as the denoiser. While the BM4D was designed to denoise volumetric data, it was also established in [7] as suitable for video denoising (especially for low-motion sequences). Since the proposed postprocessing technique uses a well established denoiser as a subroutine, we compare our method with a single application of this denoiser as a postprocessing procedure. This competing approach is further strengthened by endorsing the denoiser with an oracle capability by searching for the best parameter in terms of maximal average frame-PSNR (AFPSNR) result. More specifically, this oracle denoiser optimizes its output AFPSNR based on the knowledge of the precompressed video, a capability that cannot be applied in a real postprocessing task.

The code was implemented in Matlab. The following stopping criterion was applied. In (5) we introduced the scaled dual-variable of the i^{th} iteration, $\mathbf{u}_i \in \mathbb{R}^N$. We here denote $\Delta\mathbf{u}_i = \frac{1}{N}\|\mathbf{u}_i - \mathbf{u}_{i-1}\|_1$ and set the algorithm termination conditions to be at one of the following: $\Delta\mathbf{u}_i < 0.05$, $\Delta\mathbf{u}_i > \Delta\mathbf{u}_{i-1}$ or maximal number of five iterations attained. The remaining parameters are set as follows. The derivatives are approximated according to a spatial block-size $B_H \times B_W$ of 8×8 pixels and $S_\delta = \left\{27 \cdot k \times (bpp)^{-1}\right\}_{k=1}^{5}$, where bpp is the bit-rate of the group of frames. The components in the optimization problems are weighted by $\lambda = 0.15$, $\beta = b_0 \times 5 \cdot 10^{-4} \times (bpp)^{-1}$ and $\mu = 0.3 \times 2^{0.1875 \cdot bpp}$, where b_0 is a parameter depending on the temporal characteristics of the processed video. We set $b_0 = 10$ for low-motion videos (e.g., Akiyo, News and Hall Monitor), and $b_0 = 1$ for sequences of higher motion-levels (such as Highway and Ice). Note that the parameter settings can be further improved, e.g., the parameter b_0 can be automatically determined as a function of the average squared difference of frames in the postprocessed sequence.

We evaluated our method for several video sequences at CIF resolution (frame size of 352×288 pixels), where a group of 16 frames forms the spatio-temporal signal to consider. The quality of the reconstructed-from-compression and the postprocessed videos is measured in terms of average frame-PSNR (AFPSNR) of the sequence. The reconstruction AFPSNR of our method reached up to 0.9 dB improvement of the Motion-JPEG2000 output (Table I). In addition, our results compete with the oracle denoiser (see Table I). These results are encouraging since the oracle denoiser needs the precompressed video and, therefore, is not suitable for the common compression applications. Furthermore, the results in Table I establish our technique as suitable for a wide range of bit-rates. The restoration results visually demonstrated the artifact reduction using our method, specifically, handling of the ringing artifact (see Figures 1-2).

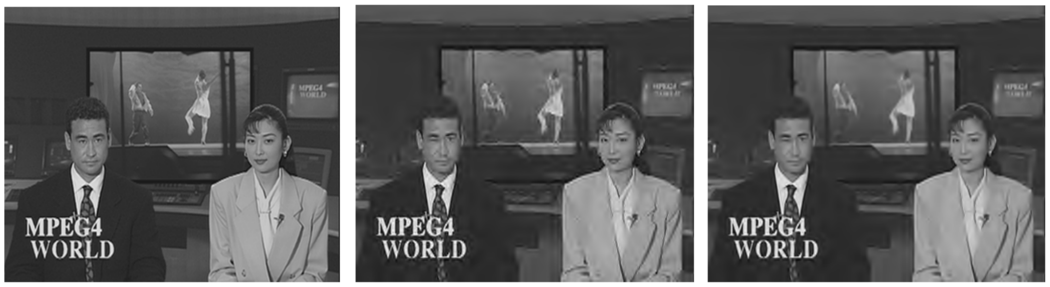
IV. Conclusion

In this paper we proposed a postprocessing method for reducing artifacts in intra-frame coded videos. The task was formulated as a regularized inverse problem, that was subsequently transformed into an iterative form by relying on the ADMM and the Plug-and-Play frameworks. The resulting generic algorithm separately treats the inversion and the regularization, where the latter is implemented by sequentially applying an existing state-of-the-art video Gaussian denoiser.



(a) Precompressed Frame (b) Motion-JPEG2000 (33.48dB) (c) Postprocessing Result (34.16dB)

Fig. 1. Reconstruction of the 5^{th} frame of Highway (CIF) from Motion-JPEG2000 compression at 0.08bpp.



(a) Precompressed Frame (b) Motion-JPEG2000 (32.19dB) (c) Postprocessing Result (32.85dB)

Fig. 2. Reconstruction of News (CIF) from Motion-JPEG2000 compression at 0.27bpp.

TABLE I
RESULT COMPARISON FOR MOTION-JPEG2000

Video	Bit Rate	Motion-JPEG2000	Oracle Denoiser	Proposed Method
		AFPSNR	AFPSNR	AFPSNR
Akiyo	0.08	30.76	**31.08**	**31.07**
	0.09	31.76	**32.17**	**32.14**
	0.12	33.08	**33.58**	**33.53**
	0.16	35.08	**35.80**	35.71
	0.27	38.61	39.19	**39.28**
News	0.08	25.41	**25.65**	**25.68**
	0.09	26.31	**26.53**	**26.56**
	0.12	27.35	**27.68**	**27.68**
	0.16	28.97	**29.58**	**29.54**
	0.27	32.08	**32.76**	**32.77**
Hall Monitor	0.08	25.89	**26.17**	**26.17**
	0.09	26.80	**27.13**	**27.15**
	0.12	27.97	**28.45**	**28.44**
	0.16	29.78	**30.57**	30.51
	0.27	33.29	**34.34**	34.26
Highway	0.08	33.36	**33.98**	**34.00**
	0.09	34.42	34.83	**35.01**
	0.12	35.64	35.75	**36.25**
	0.16	37.36	36.73	**37.86**
	0.27	39.98	37.69	**40.13**
Ice	0.08	28.25	**28.88**	28.67
	0.09	29.35	**30.09**	29.91
	0.12	30.67	**31.47**	31.39
	0.16	32.47	32.88	**33.16**
	0.27	35.49	34.77	**36.32**

The best results (up to a difference of 0.05dB in the average frame-PSNR) are marked in bold text.

For practicality, we simplified the inversion step by representing the nonlinear compression-decompression procedure using a linear approximation. Moreover, by utilizing the structure of the intra-frame coding procedure, we further eased the computational cost of the linearization and the corresponding optimization problem. We demonstrated our approach for Motion-JPEG2000 compression and presented experimental-results showing impressive gains.

REFERENCES

[1] A. Averbuch, A. Schclar, and D. L. Donoho, “Deblocking of block-transform compressed images using weighted sums of symmetrically aligned pixels,” *IEEE Trans. Image Process.*, vol. 14, no. 2, pp. 200–212, 2005.

[2] C. Jung, L. Jiao, H. Qi, and T. Sun, “Image deblocking via sparse representation,” *Signal Processing: Image Communication*, vol. 27, no. 6, pp. 663–677, 2012.

[3] Y. Dar, A. M. Bruckstein, M. Elad, and R. Giryes, “Postprocessing of compressed images via sequential denoising,” *IEEE Trans. on Image Process.*, vol. 25, no. 7, pp. 3044–3058, July 2016.

[4] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg, “Plug-and-play priors for model based reconstruction,” in *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2013, pp. 945–948.

[5] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[6] M. Maggioni, G. Boracchi, A. Foi, and K. Egiazarian, “Video denoising using separable 4D nonlocal spatiotemporal transforms,” *Proc. SPIE*, vol. 7870, pp. 787 003–787 003–11, 2011.

[7] M. Maggioni, V. Katkovnik, K. Egiazarian, and A. Foi, “Nonlocal transform-domain filter for volumetric data denoising and reconstruction,” *IEEE Trans. on Image Process.*, vol. 22, no. 1, pp. 119–133, 2013.

[8] B. Girod, A. M. Aaron, S. Rane, and D. Rebollo-Monedero, “Distributed video coding,” *Proceedings of the IEEE*, vol. 93, no. 1, pp. 71–83, 2005.

[9] M. V. Afonso, J. M. Bioucas-Dias, and M. A. T. Figueiredo, “Fast image recovery using variable splitting and constrained optimization,” *IEEE Trans. Image Process.*, vol. 19, no. 9, pp. 2345–2356, 2010.

[10] T. Fukuhara, K. Katoh, S. Kimura, K. Hosaka, and A. Leung, “Motion-JPEG2000 standardization and target market,” in *IEEE ICIP*, 2000.

[11] C. Christopoulos, A. Skodras, and T. Ebrahimi, “The JPEG2000 still image coding system: an overview,” *IEEE Trans. Consum. Electron.*, vol. 46, no. 4, pp. 1103–1127, 2000.