Joint Adaptive Sparsity and Low-Rankness on the Fly: An Online Tensor Reconstruction Scheme for Video Denoising

Bihan Wen Yanjun Li Luke Pfister Yoram Bresler
Electrical and Computer Engineering and Coordinated Science Laboratory
University of Illinois at Urbana-Champaign, IL, USA.
{bwen3, yli145, lpfiste2, ybresler}@illinois.edu

Abstract

Recent works on adaptive sparse and low-rank signal modeling have demonstrated their usefulness, especially in image/video processing applications. While a patch-based sparse model imposes local structure, low-rankness of the grouped patches exploits non-local correlation. Applying either approach alone usually limits performance in various low-level vision tasks. In this work, we propose a novel video denoising method, based on an online tensor reconstruction scheme with a joint adaptive sparse and low-rank model, dubbed SALT. An efficient and unsupervised online unitary sparsifying transform learning method is introduced to impose adaptive sparsity on the fly. We develop an efficient 3D spatio-temporal data reconstruction framework based on the proposed online learning method, which exhibits low latency and can potentially handle streaming videos. To our knowledge, this is the first work that combines adaptive sparsity and low-rankness for video denoising, and the first work of solving the proposed problem in an online fashion. We demonstrate video denoising results over commonly used videos from public datasets. Numerical experiments show that the proposed video denoising method outperforms competing methods.

1. Introduction

Denoising is one of the most important problems in video processing. Despite today's vast improvement in camera sensors, videos captured at high speed and in low light conditions are still corrupted by severe noise due to high sensitivity (*i.e.* ISO) among other reasons. The problem of noise in videos is gaining prominence with the ubiquitous use of relatively low-quality cameras in smart phones and other devices. Therefore, recovering high quality videos from noisy footage is of great interest as a low-level vision problem, and also improves robustness in high-level vision tasks [2].

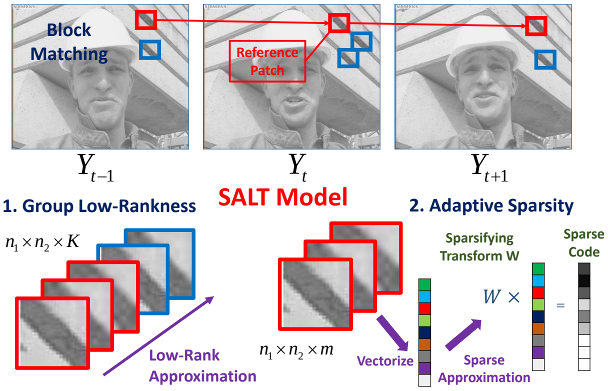


Figure 1. A simple illustration of the SALT model for video

Video denoising presents challenges that are distinct from other multi-frame image data, such as volumetric data (*e.g.* 3D medical image data) or hyperspectral data. Hyperspectral images, in particular, typically exhibit strong correlation in a small spatial window along the spectral dimension [12, 17]. In video, however, objects can move throughout or exit the scene and such long-term correlations may not exist [11]. Further, many video denoising applications are of a streaming nature and a low-latency denoising method is required. In this environment a denoising algorithm can depend only on a small number of frames [28].

Most video denoising methods take advantage of local or non-local structures present in video data. Natural videos have local structures that are sparse or compressible in some transform domain or over certain dictionaries, *e.g.*, discrete cosine transform (DCT) and wavelets. One can exploit this fact and reduce noise by coefficient shrinkage, *e.g.*, sparse approximation or Wiener filtering, of the compressible representation [18, 24]. Beyond these local structures captured by sparsity, videos also contain non-local structures, such as spatial similarity and temporal redundancy. State-of-the-art video and image denoising algorithms group similar structures across the spatial and temporal dimensions (usually within a spatio-temporal neighborhood) and apply a denoising operation jointly to a group. A successful approach of this nature comprises the following steps: 1) group similar patches; 2) jointly denoise a group of patches; 3) aggregate the denoised patches to construct the final estimate [2–7, 11–13, 30]. State-of-the-art algorithms can be compared by considering the methods by which patch grouping and denoising are implemented.

The well-known BM3D image denoising algorithm [5] has been extended to both volumetric data [12] and video data [11]. In both cases, a block matching (BM) algorithm is used to group similar 3D cubes of data forming patch groups and patches are denoised by coefficient shrinkage in a 4D transform domain. The video version, VBM4D, augments the BM algorithm with motion estimation to track objects as they move throughout the scene [11]. Buades *et al.* proposed a similar video denoising algorithm that differs in both the patch grouping and denoising strategy [3]. Patch grouping incorporates the optical flow algorithm for motion estimation, and the grouped patches are denoised by low-rank (LR) matrix approximation. Dong *et al.* proposed a multi-frame image denoising algorithm that uses BM to extract similar 3D patches of data [6]. Rather than transform domain thresholding, they denoise the resulting tensor using a low-rank approximation. A recent approach splits videos into sparse and low-rank “layers” before denoising [9].

While some of the above algorithms leverage sparsity in the denoising stage, they do so in a fixed transform domain. However, it has been shown in many low-level vision tasks, including image and video denoising, that data-adaptive representations usually lead to superior performance over fixed sparse representations [8,27,28]. Synthesis dictionary learning is the most well-known adaptive representation learning scheme [1, 8, 26]. Unfortunately, dictionary learning features typically NP-hard sparse coding steps [14], for which commonly-used greedy approximate algorithms still involve relatively expensive computations [15, 16]. As an alternative, sparsifying transform learning [20] with cheap sparse coding steps has been proposed and shown to be efficient and effective in finding sparse approximations of image data [21, 27, 29]. The recent online variants of the transform learning [22] are especially applicable to streaming large scale, or high-dimensional data, and have demonstrated promising performance for video denoising [28].

In summary, transform domain sparsity and low-rankness in groups of similar patches capture local and non-local structures in video data, respectively. Similar observations are also true for images, and the combination of these two priors has been exploited in single-frame and hyperspectral image denoising algorithms [17]. However, to the best of our knowledge, no video denoising algorithm has to date utilized both data-adaptive sparse and low-rank priors. In this paper, we introduce an online video denoising scheme called Sparse And Low-rank Tensor (**SALT**) reconstruction, which exploits both local and non-local structures. Table 1 summarizes the proposed SALT method, its variations (BM-DCT, BM-TL, and BM-LR), as well as some of the aforementioned competing video denoising methods with their key attributes (see Section 5.2 for more details about the variations and the competing methods).

Methods	**Local Sparse Model**			**BM**	**Non -local Method**
	Fixed	Adapt -ive	Online Update		
fBM3D [5]	✓			✓	✓
sKSVD [23]		✓			
VIDOSAT [28]		✓	✓		
VBM3D / VBM4D [4] / [11]	✓			✓	✓
BM-DCT	✓			✓	
BM-TL		✓	✓	✓	
BM-LR				✓	✓
SALT		✓	✓	✓	✓

Table 1. Comparison of the key attributes between the proposed SALT denoising, its variations, and the competing methods.

Our contributions can be summarized as follows:

- We propose a video denoising algorithm that combines benefits from sparse and low-rank approximations, and produces reconstruction results that are better than either alone. The proposed algorithm also outperforms the competing video denoising methods.
- Our video denoising algorithm processes noisy videos in an online fashion. Given incoming frames, it: 1) groups similar patches using BM, 2) adapts a sparsifying transform, 3) finds sparse approximation of patches, 4) finds low-rank approximation of a group of similar patches, 5) reconstructs clean video frames. The algorithm is efficient and scalable, and hence is applicable to high-definition and high-speed videos.
- We propose an online unitary transform learning algorithm, which is especially applicable to large scale streaming data. This algorithm enables faster reconstruction when applied to denoising and potentially other signal restoration problems.
- In fact, the proposed SALT model can be applied to restoration of videos with “local” corruption (such as noise, defective pixels, blur, and color mosaic) with little change to the algorithm.

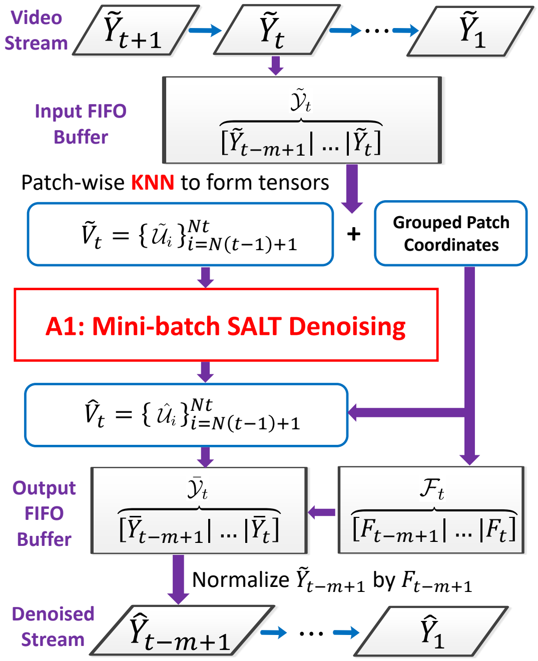

Video Stream
$\tilde{Y}_{t+1}$
$\tilde{Y}_t$
$\tilde{Y}_1$
Input FIFO Buffer
$\tilde{\mathcal{Y}}_t$
$[\tilde{Y}_{t-m+1}| \dots |\tilde{Y}_t]$
Patch-wise KNN to form tensors
$\tilde{V}_t = \{\tilde{\mathcal{U}}_i\}_{i=N(t-1)+1}^{Nt}$
+
Grouped Patch Coordinates
A1: Mini-batch SALT Denoising
$\hat{V}_t = \{\hat{\mathcal{U}}_i\}_{i=N(t-1)+1}^{Nt}$
Output FIFO Buffer
$\bar{\mathcal{Y}}_t$
$[\bar{Y}_{t-m+1}| \dots |\bar{Y}_t]$
$\mathcal{F}_t$
$[F_{t-m+1}| \dots |F_t]$
Normalize $\bar{Y}_{t-m+1}$ by F_{t-m+1}
Denoised Stream
$\hat{Y}_{t-m+1}$
$\hat{Y}_1$

Figure 2. A diagram for SALT based video denoising

2. SALT Video Denoising Framework

We present a video denoising framework based on SALT online reconstruction, in which streaming frames can be denoised online with a constant buffer and fixed latency.

Prior work [28] on video denoising based on transform learning introduced a video stream processing method, called VIDOSAT, which learns a sparsifying transform for 3D spatio-temporal patches of contiguous pixels. As video typically involves various types of motion, patch grouping methods are widely used to generate high-dimensional data with better correlation and redundancy [6, 11]. We therefore extend the streaming scheme of VIDOSAT, so that group matching is applied to generate 3D tensors, which are then sequentially denoised using the mini-batch SALT denoising method, which we discuss in Section 4. The reconstructed tensors are aggregated to output denoised frame estimates.

Figure 2 illustrates the streaming scheme in the proposed SALT based video denoising framework. We assume that the video stream is corrupted by additive i.i.d. Gaussian noise. The noisy frames, denoted by $\tilde{Y}_\tau = Y_\tau + \xi_\tau \in \mathbb{R}^{a\times b}$, arrive sequentially at time $\tau = 1, 2, 3$, *etc.* At time instant $\tau = t$, the newly arrived $\tilde{Y}_t$ is added to a fixed-size first-in-first-out (FIFO) input buffer $\tilde{\mathcal{Y}}_t \in \mathbb{R}^{a\times b\times m}$. The buffer stores m (set to be odd) consecutive frames $\tilde{\mathcal{Y}}_t = \left[\tilde{Y}_{t-m+1} \mid \tilde{Y}_{t-m+2} \mid \ \dots \ \mid \tilde{Y}_t\right]$, and drops the oldest frame $\tilde{Y}_{t-m}$ once the new frame $\tilde{Y}_t$ arrives. We extract all 2D overlapping patches from the middle frame $\tilde{Y}_{t-(m-1)/2}$ of $\tilde{\mathcal{Y}}_t$. Suppose there exist N such patches in total, and we denote the i-th patch by $\tilde{Z}_i \in \mathbb{R}^{n_1\times n_2}$, where i belongs to an index set $S_t = \{N(t-1)+1, \dots, Nt\}$. For each $i \in S_t$, we set an $h_1 \times h_2 \times m$ search window centered at $\tilde{Z}_i$ and use the K-nearest neighbor (KNN) method to find the K most similar patches within this window in terms of their Euclidean distances to $\tilde{Z}_i$. The grouped patches, in ascending order of Euclidean distances, form a tensor $\tilde{\mathcal{U}}_i \in \mathbb{R}^{n_1\times n_2\times K}$ which is assumed to satisfy the SALT model. As $\tilde{Z}_i$ has zero distance to itself, it is always found as the leading patch in $\tilde{\mathcal{U}}_i$. The coordinates of the grouped patches are also recorded, and later used for video reconstruction. The set of extracted tensors from the input buffer $\tilde{\mathcal{Y}}_t$, denoted by $\tilde{V}_t = \{\tilde{\mathcal{U}}_i\}_{i\in S_t}$, forms the input to the mini-batch SALT denoising scheme.

The output of the mini-batch denoising algorithm $\hat{V}_t = \{\hat{\mathcal{U}}_i\}$ are accumulated to the fixed-size output buffer $\bar{\mathcal{Y}}_t = \left[\bar{Y}_{t-m+1} \mid \ \dots \ \mid \bar{Y}_t\right] \in \mathbb{R}^{a\times b\times m}$, *i.e.*, the 2D patches grouped in $\hat{V}_t$ are added to $\bar{\mathcal{Y}}_t$ at their respective locations, and the numbers of occurrences of these 2D patches are accumulated accordingly in the output weighting buffer $\mathcal{F}_t = \left[F_{t-m+1} \mid \ \dots \ \mid F_t\right] \in \mathbb{R}^{a\times b\times m}$. Similar to the FIFO $\tilde{\mathcal{Y}}_t$, once the newly denoised $\{\hat{\mathcal{U}}_i\}$ and the counts of occurrences of its patches are accumulated in the output buffers, the streaming scheme outputs the oldest (leftmost) $\bar{Y}_{t-m+1}$ and F_{t-m+1}, which have finished aggregation and will not be influenced by future output of the mini-batch denoising algorithm. The denoised estimate $\hat{Y}_{t-m+1}$ of the frame Y_{t-m+1} is computed by normalizing $\bar{Y}_{t-m+1}$ by the weights F_{t-m+1}. The remaining frames in $\bar{\mathcal{Y}}_t$ will be updated further based on future outputs of the mini-batch denoising algorithm $\{\hat{V}_\tau\}_{\tau=t+1}^{t+m-1}$. Thus, there is a fixed latency of $(m-1)$ frames between the arrival of noisy $\tilde{Y}_\tau$ and the production of its final denoised estimate $\hat{Y}_\tau$.

3. SALT Formulation

In this section, we first introduce the formulations of online unitary transform learning, and online SALT denoising. Then we propose a mini-batch SALT denoising formulation, which is extended from the online formulation, and is used in the video denoising scheme illustrated in Figure 2.

3.1. Online Sparsifying Transform Learning with Unitary Constraint

We propose to learn a unitary sparsifying transform from streaming data in an online fashion. We wish to adaptively update a unitary transform to approximately sparsify sequentially arrived, or processed data. For time $t = 1, 2, \dots$, we compute the unitary transform $\hat{W}_t \in \mathbb{R}^{n\times n}$ and the sparse code $\hat{\alpha}_t \in \mathbb{R}^n$ for new data $x_t \in \mathbb{R}^n$ by solving the following optimization problem:

$$\left\{\hat{W}_t, \hat{\alpha}_t\right\} = \underset{W, \alpha_t}{\operatorname{argmin}} \frac{1}{t} \sum_{\tau=1}^{t} \left\{ \|W x_\tau - \alpha_\tau\|_2^2 + \rho^2 \|\alpha_\tau\|_0 \right\}$$
$$s.t. \quad W^T W = I_n \qquad \text{(P1)}$$

where $I_n \in \mathbb{R}^{n \times n}$ is the identity matrix, and a unitary constraint $W^T W = I_n$ is imposed. Here $\hat{\alpha}_t$ is the optimal sparse code for x_t, and $\hat{W}_t$ is optimized for all $\{x_\tau\}_{\tau=1}^t$ and $\{\alpha_\tau\}_{\tau=1}^t$ until time t. The ℓ_0 "norm" $\|\alpha_\tau\|_0$ counts the number of nonzeros in α_τ, thus imposing sparsity on x_τ under transform W. Since only the latest α_t is updated at time t, we assume $\alpha_\tau = \hat{\alpha_\tau}$ for $1 \leq \tau \leq t-1$ [22,28].

3.2. Online SALT Denoising

Based on the online unitary transform learning formulation, we propose an online tensor reconstruction scheme, dubbed online SALT, that denoises streaming tensor data $\{\tilde{\mathcal{U}}_\tau\}_{\tau=1}^t$ based on sparse and low-rank approximation. The noisy tensor measurement is $\tilde{\mathcal{U}}_\tau = \mathcal{U}_\tau + \epsilon_\tau$, where $\mathcal{U}_\tau$ is the clean tensor, and ϵ_τ is additive noise.

To facilitate our discussion of sparse and low-rank approximation, we define some reshaping operations on tensors. We use $\operatorname{mat}(\cdot) : \mathbb{R}^{n_1 \times n_2 \times K} \to \mathbb{R}^{n_s \times K}$ to denote the matricization operation that unfolds the first two modes of a third-order tensor, where $n_s = n_1 \times n_2$. We use $\operatorname{vec}(\cdot) : \mathbb{R}^{n_s \times K} \to \mathbb{R}^n$ to denote the vectorization operation on a matrix, where $n = n_s \times K$. The relations between a third-order tensor $\mathcal{U} \in \mathbb{R}^{n_1 \times n_2 \times K}$, its matricization $U = \operatorname{mat}(\mathcal{U})$, and its vectorization $u = \operatorname{vec}(U)$ can be summarized by the following diagram:

$$\mathcal{U} \in \mathbb{R}^{n_1 \times n_2 \times K} \underset{\operatorname{mat}^{-1}}{\overset{\operatorname{mat}}{\rightleftharpoons}} U \in \mathbb{R}^{n_s \times K} \underset{\operatorname{vec}^{-1}}{\overset{\operatorname{vec}}{\rightleftharpoons}} u \in \mathbb{R}^n.$$

The **SALT model** assumes that the vectorization u is approximately sparsifiable by some unitary transform $W \in \mathbb{R}^{n \times n}$, *i.e.*, $Wu = \alpha + e$, where α is a sparse vector, and e is a small (in terms of ℓ_2 norm) modeling error. Additionally, the SALT model enforces the matricization U to be approximately low-rank, *i.e.*, $U = D + E$, where D is a low-rank matrix, and E is a small (in terms of Frobenius norm) residual. Figure 1 illustrates SALT model for video.

Consider streaming tensor data with noise corruption, $\left\{\tilde{\mathcal{U}}_\tau\right\}_{\tau=1}^t$, that we wish to denoise sequentially. The online SALT denoising scheme is solving the following optimization problem sequentially (for $t = 1, 2, 3, ...$):

$$\min_{\{W, \alpha_t, D_t, U_t\}} \gamma_s \frac{1}{t} \sum_{\tau=1}^{t} \left\{ \|W u_\tau - \alpha_\tau\|_2^2 + \rho^2 \|\alpha_\tau\|_0 \right\}$$
$$+ \gamma_l \frac{1}{t} \sum_{\tau=1}^{t} \left\{ \|U_\tau - D_\tau\|_F^2 + \theta^2 \operatorname{rank}(D_\tau) \right\}$$
$$+ \gamma_f \frac{1}{t} \sum_{\tau=1}^{t} \left\| U_\tau - \operatorname{mat}(\tilde{\mathcal{U}}_\tau) \right\|_F^2$$
$$s.t. \quad u_\tau = \operatorname{vec}(U_\tau) \ \forall \tau, \ W^T W = I_n \qquad \text{(P2)}$$

where $\operatorname{rank}(\cdot)$ returns the rank of a matrix. The solution to (P2) at time t is denoted as $\left\{\hat{W}_t, \hat{\alpha}_t, \hat{D}_t, \hat{U}_t\right\}$, which jointly minimizes the sparsity and the LR modeling errors, as well as the data fidelity to $\operatorname{mat}(\tilde{\mathcal{U}}_\tau)$ – the matricized version of the noisy tensor measurement. Here $\hat{\alpha}_t$ is the optimal sparse code for u_t, $\hat{D}_t$ is the low-rank approximation of U_t, and $\hat{U}_t$ is the reconstruction of U_t under the SALT model. We update the sparsifying transform $\hat{W}_t$, and the sparse code $\hat{\alpha}_t$ online to be optimal for $\{u_\tau\}_{\tau=1}^t$, which coincides with the online unitary transform learning problem in Section 3.1.

3.3. Mini-Batch SALT in Video Denoising

We now discuss the mini-batch SALT denoising formulation, which is extended from the online SALT denoising problem (P2) described in Section 3.2, and used in the proposed video denoising framework. The modified mini-batch SALT denoising problem is the following

$$\min_{\{W, \{\alpha_i, D_i, U_i\}_{i \in S_t}\}} \frac{\gamma_f}{tN} \sum_{\tau=1}^{t} \varrho^{t-\tau} \sum_{i \in S_\tau} \left\| U_i - \operatorname{mat}(\tilde{\mathcal{U}}_i) \right\|_F^2$$
$$+ \frac{\gamma_l}{tN} \sum_{\tau=1}^{t} \varrho^{t-\tau} \sum_{i \in S_\tau} \left\{ \|U_i - D_i\|_F^2 + \theta^2 \operatorname{rank}(D_i) \right\}$$
$$+ \frac{\gamma_s}{tN} \sum_{\tau=1}^{t} \varrho^{t-\tau} \sum_{i \in S_\tau} \left\{ \|W u_i^m - \alpha_i\|_2^2 + \rho^2 \|\alpha_i\|_0 \right\}$$
$$s.t. \quad u_i^m = \operatorname{vec}(C_{1:m} U_i^m) \ \forall i, \ W^T W = I_n \qquad \text{(P3)}$$

where $S_\tau = \{N(\tau-1)+1, \ldots, N\tau\}$ indicates the range of tensors $\{\tilde{\mathcal{U}}_i\}_{i \in S_\tau}$ in the current mini-batch V_τ. There are in total N tensors in each mini-batch. Comparing to the online SALT denoising problem (P2), there are three major variations introduced in this extension: (a) mini-batch transform update, (b) temporal forgetting factor, and (c) reduced-size sparse approximation.

(a) **Mini-batch transform update**: Instead of updating the transform after each tensor reconstruction, we only update it once per mini-batch [22, 28]. This is motivated by two reasons: a) each mini-batch V_τ contains relatively stationary training data, which can be sparsified by the same transform W, and b) transform update involves a relatively intensive computation of a full SVD with $O(n^3)$ complexity. Mini-batch updates lower the overall computational cost by reducing the number of transform updates by a factor of N.

(b) **Temporal forgetting factor**: To better adapt the sparsifying transform W to temporally local structures of video data, we introduce a temporal forgetting factor $\varrho^{t-\tau}$ with a constant $0 < \varrho < 1$. The use of the forgetting factor diminishes the influence from early training data [28]. This is especially useful when denoising videos with dynamically changing frames, or scene changes.

(c) **Reduced-size sparse approximation**: In the online SALT reconstruction, we find the sparse approximation of the entire $U_i \in \mathbb{R}^{n_s \times K}$ under the adaptive 3D transform W. As a relatively large K is used in our approach, we need to train a large transform W, which leads to high computational cost and overfitting. To alleviate this issue, we only find sparse approximation of the reduced-size $U_i^m = C_{1:m} U_i$, where the operator $C_{1:m}$ maps U_i to the sub-matrix formed by the first m columns of U_i. The sparsifying transform W is of reduced size $n \times n$, where $n = n_s \times m$.

4. Algorithm

We solve problem (P3) using an efficient block coordinate descent algorithm, which runs one iteration per time instance t. Each iteration involves 4 steps: (i) sparse coding, (ii) mini-batch transform update, (iii) LR approximation, and (iv) SALT reconstruction, which compute or update $\{\alpha_i\}_{i \in S_t}$, W_t, $\{D_i\}_{i \in S_t}$, and $\{U_i\}_{i \in S_t}$, respectively.

At each time instance t, each noisy tensor $\tilde{\mathcal{U}}_i$ from the current input $\tilde{V}_t$ (*i.e.*, $\forall i \in S_t$), is first matricized to $\mathrm{mat}(\tilde{\mathcal{U}}_i)$ as an initial estimate of U_i. Once an iteration completes, we recover each tensor $\hat{\mathcal{U}}_i$ by reshaping the denoised output $\hat{U}_i$ back to tensor $\hat{\mathcal{U}}_i = \mathrm{mat}^{-1}(\hat{U}_i)$, to form the output of the mini-batch algorithm $\hat{V}_t$. The four steps of one iteration at time t are illustrated as follows:

(i) **Sparse Coding**: Given the initial value of each U_i and the updated sparsifying transform $\hat{W}_{t-1}$ from the last iteration, we first vectorize the first m columns of the noisy measurement as $u_i^m = \mathrm{vec}(C_{1:m} U_i)$. We solve the Problem (P3) for the optimal sparse code $\forall i \in S_t$,

$$\hat{\alpha}_i = \underset{\alpha_i}{\operatorname{argmin}} \left\| \hat{W}_{t-1} u_i^m - \alpha_i \right\|_2^2 + \rho^2 \left\| \alpha_i \right\|_0 \qquad (1)$$

which is the standard sparse coding problem under the transform model. The optimal solution $\hat{\alpha}_i$ is obtained as $\hat{\alpha}_i = H_\rho(\hat{W}_{t-1} u_i^m)$ by cheap hard thresholding [21], where the hard thresholding operator $H_\rho(\cdot)$ is defined as

$$(H_\rho(b))_j = \begin{cases} 0 & , \ |b_j| < \rho \\ b_j & , \ |b_j| \geq \rho \end{cases}$$

where $b \in \mathbb{R}^n$ denotes the input vector, scalar $\rho \geq 0$ denotes the threshold value, and the subscript j denotes indices of vector entries. Note that $H_\rho(\cdot)$ can be generalized to take a matrix as the input, and similarly it zeros out all elements with magnitude smaller than ρ in the matrix.

```
Algorithm A1: Mini-batch SALT Denoising

Input: The noisy mini-batch $\{\tilde{V}_\tau\}_{\tau=1}^{t}$ sequence ($\tilde{V}_\tau = \{\tilde{\mathcal{U}}_i\}_{i=N(\tau-1)+1}^{N\tau}$), and the initial transform $W_0$.
Initialize: $\hat{W}_0 = W_0$, $\tilde{\Gamma}_0 = 0$, and $U_i = \mathrm{mat}(\tilde{\mathcal{U}}_i)\ \forall i = 1, 2, ...Nt$.
For $\tau = 1, 2, ..., t$ Repeat
  Index set: $S_\tau = \{N(\tau-1)+1, \dots, N\tau\}$.
  1. Sparse Coding: $\forall i \in S_\tau$
      (a) Vectorize $u_i^m = \mathrm{vec}(\,C_{1:m}(U_i)\,)$.
      (b) Sparsify $\hat{\alpha}_i = H_\rho(\hat{W}_{\tau-1} u_i^m)$.
  2. Mini-batch Transform Update:
      (a) $\Gamma_\tau = \varrho(1-\tau^{-1})\Gamma_{\tau-1} + \tau^{-1}\sum_{i \in S_\tau} u_i^m\, \hat{\alpha}_i^T$.
      (b) Full SVD: $\Phi_\tau \Sigma_\tau \Psi_\tau^T = \mathrm{SVD}(\Gamma_\tau)$.
      (c) Update $\hat{W}_\tau = \Psi_\tau \Phi_\tau^T$.
  3. LR Approximation: $\forall i \in S_\tau$
      (a) Economy-size: $\Lambda_i \Omega_i \Delta_i^T = \mathrm{SVD}(U_i)$.
      (b) LR Approximate $\hat{D}_i = \Lambda_i H_\theta(\Omega_i) \Delta_i^T$.
  4. SALT Reconstruction: $\forall i \in S_\tau$
      (a) Sparse coding: $\hat{\alpha}_i^* = H_\rho(\hat{W}_\tau u_i^m)$.
      (b) Reconstruct first $m$ columns of $\hat{U}_i$ by (4).
      (c) Reconstruct last $K-m$ columns of $\hat{U}_i$ by (5).
      (d) Tensorize $\hat{\mathcal{U}}_i = \mathrm{mat}^{-1}([\hat{U}_{i,1} \mid \hat{U}_{i,2}])$.
End
Output: The reconstructed (denoised) tensor mini-batch $\left\{\hat{V}_\tau\right\}_{\tau=1}^{t}$ sequence, the learned transform $\hat{W}_t$.
```

(ii) **Mini-batch transform update**. Fixing $\{u_i^m\}_{i=1}^{Nt}$ and $\{\hat{\alpha}_i\}_{i=1}^{Nt}$, we solve for the mini-batch unitary transform update sub-problem at time t in (P3) as follows:

$$\hat{W}_t = \underset{W}{\operatorname{argmin}} \ \frac{1}{tN} \sum_{\tau=1}^{t} \varrho^{t-\tau} \sum_{i \in S_\tau} \| W u_i^m - \hat{\alpha}_i \|_2^2 \qquad (2)$$
$$s.t. \quad W^T W = I_n$$

Prior work on batch unitary transform learning introduced closed-form transform update [19]. Similarly, the optimal solution $\hat{W}_t$ to problem (2) has a simple and exact solution. We define $\Gamma_t = \sum_{\tau=1}^{t} \varrho^{t-\tau} \sum_{i \in S_\tau} u_i^m\, \hat{\alpha}_i^T$, and compute its full SVD $\Phi_t \Sigma_t \Psi_t^T = \mathrm{SVD}(\Gamma_t)$. The closed-form solution to problem (2) is $\hat{W}_t = \Psi_t \Phi_t^T$. The matrix Γ_τ is computed sequentially over time as $\Gamma_\tau = \varrho(1-\tau^{-1})\Gamma_{\tau-1} + \tau^{-1}\sum_{i \in S_\tau} u_i^m\, \hat{\alpha}_i^T$.

(iii) **LR Approximation**: We solve (P3) for the LR matrix $\hat{D}_i$ to approximate $U_i \; \forall i \in S_t$ as

$$\hat{D}_i = \underset{D_i}{\operatorname{argmin}} \; \|U_i - D_i\|_F^2 + \theta^2 \operatorname{rank}(D_i). \qquad (3)$$

Suppose the economy-size SVD of U_i is $\Lambda_i \Omega_i \Delta_i^T = \operatorname{SVD}(U_i)$. Then (3) has a closed-form solution: $\hat{D}_i = \Lambda_i H_\theta(\Omega_i) \Delta_i^T$.

(iv) **SALT reconstruction**. We reconstruct each U_i, part of which has a sparse approximation, based on the SALT model. With fixed $\hat{W}_t$, $\hat{\alpha}_i^*$, and $\hat{D}_i$, we solve (P3) for U_i as follows:

$$\begin{aligned}\hat{U}_i = \underset{U_i}{\operatorname{argmin}} \; & \gamma_s \left\| \operatorname{vec}(C_{1:m} \, U_i) - \hat{W}_t^T \hat{\alpha}_i^* \right\|_2^2 \\ & + \gamma_l \left\| U_i - \hat{D}_i \right\|_F^2 + \gamma_f \left\| U_i - \operatorname{mat}(\tilde{\mathcal{U}}_i) \right\|_F^2 \end{aligned}$$

Denote the optimal $\hat{U}_i = \left[\hat{U}_{i,1} \mid \hat{U}_{i,2} \right]$, where $\hat{U}_{i,1} \in \mathbb{R}^{n \times m}$ and $\hat{U}_{i,2} \in \mathbb{R}^{n \times (K-m)}$ are two sub-matrices. The closed-form solutions for the sub-matrices are

$$\hat{U}_{i,1} = \frac{\gamma_s \operatorname{vec}^{-1}(W_t^T \; \hat{\alpha}_i^*) + C_{1\,:\,m}(\gamma_l \hat{D}_i + \gamma_f \operatorname{mat}(\tilde{\mathcal{U}}_i))}{\gamma_s + \gamma_l + \gamma_f} \qquad (4)$$

$$\hat{U}_{i,2} = \frac{C_{m+1\,:\,K}(\gamma_l \, \hat{D}_i + \operatorname{mat}(\tilde{\mathcal{U}}_i))}{\gamma_l + \gamma_f} \qquad (5)$$

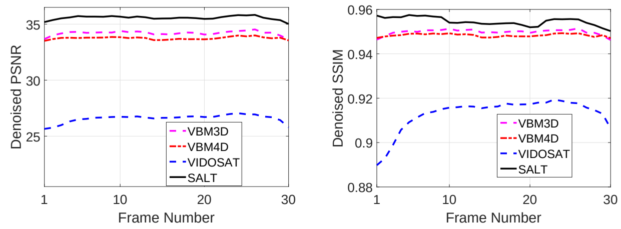
When the iteration completes at time t, each denoised $\hat{U}_i$ is tensorized to be $\hat{\mathcal{U}}_i = \operatorname{mat}^{-1}(\hat{U}_i)$ as output. Algorithm **A1** summarizes the SALT mini-batch denoising algorithm.

Algorithm Complexity. The computational cost for SALT algorithm is $O(Ntmh_1h_2 + Ntn^2K + Ntm^2n^2 + tm^3n^3 + Nt)$, corresponding to block matching (BM), low-rank approximation, sparse coding, transform update, and aggregation steps. It is on par with the state-of-the-art VBM3D, which is $O(Ntmh_1h_2 + Ntn^2K)$. The current implementation of SALT algorithm, including single-thread patch extraction and BM Matlab functions, is not yet optimized for real-time applications. We anticipate optimized code on a GPU to be significantly faster in future works.

5. Experiment

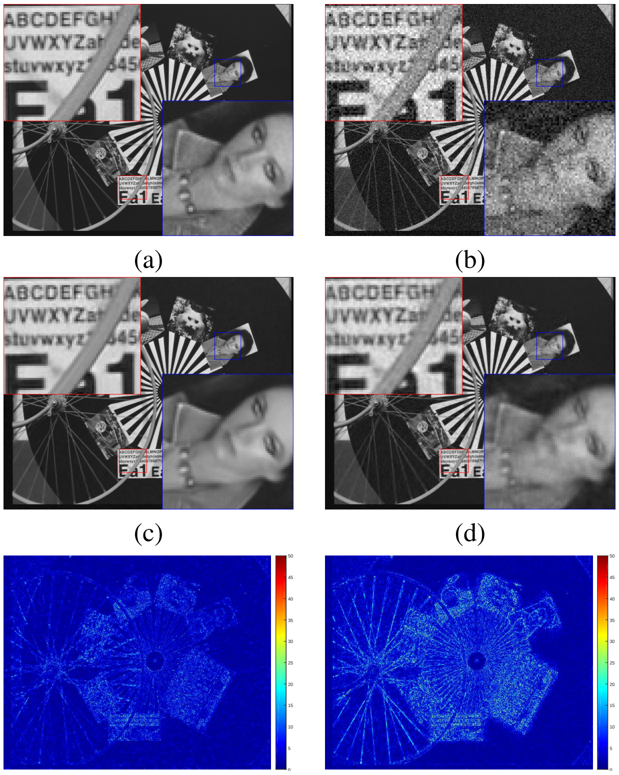
5.1. Implementation and Parameters

Testing data. We present experimental results demonstrating the promise of the proposed SALT video denoising scheme. We evaluate the proposed algorithm over commonly used videos from the Arizona State University (ASU) dataset [25] [1] and Tampere University of Technology (TUT) dataset [4, 11]. The selected testing videos contain 50 to 494 frames, with different spatial resolutions ranging from 176 × 144 to 720 × 576. Each video involves different types of motion, including translation, rotation, scaling, etc. The color videos are first converted to gray-scale. We simulate i.i.d. zero-mean Gaussian noise at 5 different noise levels (*i.e.*, with standard deviation σ = 5, 10, 15, 20, and 50) for each video.



(a) (b)

Figure 3. Frame-by-frame (a) PSNR(dB) and (b) SSIM of the video *Gbicycle* with σ = 20, denoised by the proposed SALT denoising scheme, VIDOLSAT, VBM3D and VBM4D.

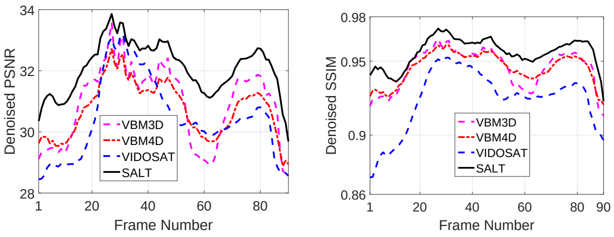


(a) (b) (c) (d) (e) (f)

Figure 4. Denoising result: (a) One frame of the clean video *Gbicycle*, (b) Frame corrupted with noise at σ = 20 (PSNR = 22.12 dB), (c) Denoised frame using the proposed SALT denoising (PSNR = 35.67 dB). (d) Denoised frame using VIDOSAT (PSNR = 31.80 dB). (e) Magnitude of error in (c). (f) Magnitude of error in (d).

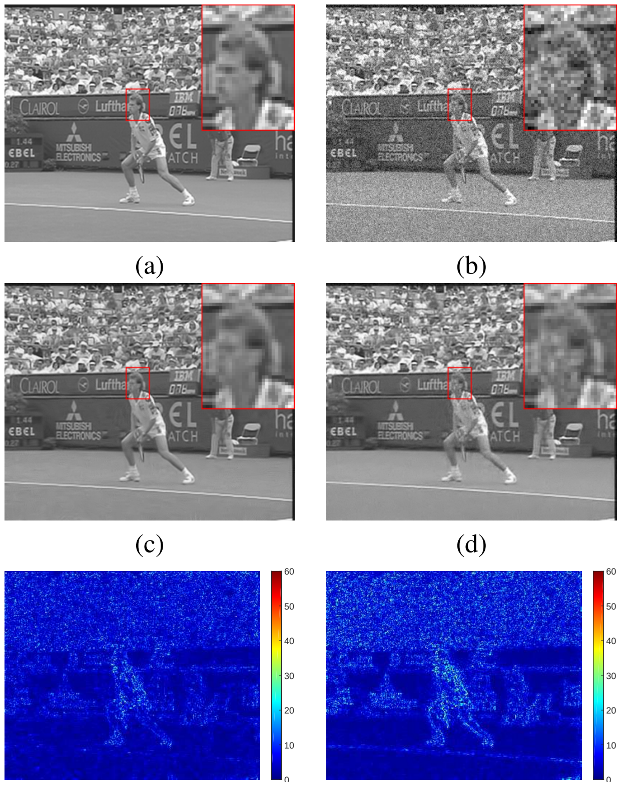
Implementation details. We explain several implementation details and minor modifications. First, at each time instant t, instead of grouping the noisy patches directly

[1] Videos from ASU dataset with less than 1000 frames are selected.



(a) (b)

Figure 5. Frame-by-frame (a) PSNR(dB) and (b) SSIM of the video *Stefan* with $\sigma = 20$, denoised by the proposed SALT denoising scheme, VIDOLSAT, VBM3D and VBM4D.



(a) (b) (c) (d) (e) (f)

Figure 6. Denoising result: (a) One frame of the clean video *Stefan*, (b) Frame corrupted with noise at $\sigma = 20$ (PSNR = 22.11 dB), (c) Denoised frame using the proposed SALT denoising (PSNR = 29.69 dB). (d) Denoised frame using VBM4D (PSNR = 28.56 dB). (e) Magnitude of error in (c). (f) Magnitude of error in (d).

by KNN, we pre-clean the input buffer sequentially using the online algorithm VIDOSAT [28], and then group pre-cleaned patches. Secondly, when the KNN searching window slides through a video, the spatial and temporal corner cases need special treatment. We extend frames by mirroring them at all boundaries and corners (symmetric boundary conditions) to accommodate search windows exceeding frame boundaries [10]. The reconstruction of the extended pixels are not aggregated to the output buffer, for the sake of computational and memory efficiency. We choose the $h_1 \times h_2 \times m$ window surrounding a patch in the first $(m-1)/2$ frames to be the same window centered at the patch in the $(m+1)/2$-th frame with the same spatial location, to ensure that the window does not temporally exceed the first frame, and still has the same size. We also apply similar treatments to the last $(m-1)/2$ frames. Thirdly, when each denoised tensor $\hat{\mathcal{U}}_i$ is aggregated to the output buffer $\bar{\mathcal{Y}}_t$, we weight the first m slices of $\hat{\mathcal{U}}_i$ by an extra factor of $(\gamma_s + \gamma_l + \gamma_f)/(\gamma_l + \gamma_f)$ (assuming the last $K - m$ slices have unit weights). Intuitively, as the first m slices of $\hat{\mathcal{U}}_i$ are reconstructed with both sparse and low-rank approximations, we expect their denoised estimates to be better, and hence assign more weights to them in the aggregation.

Parameters. The proposed SALT video denoising scheme uses an unsupervised approach, though there are several hyperparameters that require tuning. We randomly select a tuning set of 10 videos from ASU dataset, which are excluded in the denoising test in this paper. After tuning, all of the hyperparameters are fixed for evaluation over the other 18 videos from ASU dataset, and 8 videos from TUT dataset.

We work with square patches of size $n_1 = n_2 = 8$. We set the temporal search range $m = 9$, the penalty weights $\rho = 3\sigma$, $\theta = 1.1\sigma(\sqrt{K} + \sqrt{n_s})$, $\gamma_l = 1$, and $\gamma_f = 10^{-4}/\sigma$. We set $\gamma_{s,i} = 60/s_i$ for each $\hat{\mathcal{U}}_i$ (see A1, Step 4(b)), where s_i is the sparsity of $\hat{\alpha}_i^*$ (see A1, Step 4(a)). We use square search windows of size $h_1 = h_2 = h$, where h decreases from 30 to 16 as σ increases from 5 to 50. We set $K = 32$, 48, 64, 80 and 96, for $\sigma = 5$, 10, 15, 20, and 50, respectively. We use the same forgetting factor values as in the VIDOSAT algorithm [28], which are tuned empirically for each σ. We initialize the sparsifying transform with the 3D DCT W_0. Analysis behind the choices of parameters are included in the supplementary materials.

5.2. Video Denoising

Competing methods. We compare the numerical results obtained using our proposed online denoising algorithm (SALT), to various well-known alternatives including frame-wise BM3D denoising (fBM3D) [5], sparse KSVD image sequence denoising (sKSVD) [23], VIDOSAT [28], VBM3D [4], and VBM4D [11]. We use their publicly available codes for implementation. Among these methods, fBM3D makes use of only non-local spatial structures by applying a state-of-the-art image denoising method, while sKSVD and VIDOSAT exploit local spatial-temporal sparsity. VBM3D and VBM4D are considered state-of-the-art methods for video denoising. Additionally, to better understand the benefit of each of the regularizers used in our SALT model, we evaluate the denoising results reconstructed separately using only the adaptive sparse approximation (BM-TL) and the low-rank approximation (BM-LR). To verify the advantage of adaptive transform learning,

Data	**TUT** Dataset (8 videos)					Δ P
σ	5	10	15	20	50	(std.)
fBM3D [5]	38.05	34.06	31.89	30.42	25.88	-2.86 (0.78)
sKSVD [23]	38.87	34.95	32.80	31.33	26.89	-1.95 (1.02)
VIDO-SAT [28]	39.56	35.75	33.54	31.98	27.29	-1.30 (0.92)
VBM3D [4]	39.20	35.75	33.87	32.49	26.51	-1.36 (0.57)
VBM4D [11]	39.37	35.73	33.70	32.24	26.68	-1.38 (0.51)
BM-DCT	38.76	34.80	32.63	31.15	26.82	-2.09 (1.13)
BM-LR	40.54	36.93	34.82	33.32	28.42	-0.11 (0.05)
BM-TL	40.03	36.41	34.31	32.84	27.49	-0.70 (0.32)
SALT	**40.65**	**37.05**	**34.98**	**33.47**	**28.47**	0

Table 2. Comparison of video denoising PSNR values, averaged over **TUT** dataset, for the proposed SALT and competing methods. ΔP denotes the average PSNR difference (with its standard deviation) relative to SALT. For each video and noise level, the best denoising PSNR is marked in bold.

Data	**ASU** Dataset (18 videos)					Δ P
σ	5	10	15	20	50	(std.)
fBM3D [5]	39.44	35.47	33.26	31.73	27.00	-3.90 (0.95)
sKSVD [23]	41.83	38.09	35.96	34.46	29.80	-1.45 (0.77)
VIDO-SAT [28]	42.49	38.63	36.36	34.79	29.78	-0.87 (0.47)
VBM3D [4]	41.66	38.55	36.32	34.70	29.72	-1.09 (0.82)
VBM4D [11]	42.00	38.36	36.18	34.58	28.70	-1.32 (0.52)
BM-DCT	39.70	35.74	33.52	32.01	27.73	-3.54 (0.93)
BM-LR	43.13	39.36	37.05	35.42	30.11	-0.26 (0.20)
BM-TL	42.61	39.01	36.84	35.27	29.97	-0.54 (0.28)
SALT	**43.29**	**39.59**	**37.38**	**35.73**	**30.41**	0

Table 3. Comparison of video denoising PSNR values, averaged over **ASU** dataset, for the proposed SALT and competing methods. For each video and noise level, the best denoising PSNR is marked in bold.

we fix the sparsifying transform in BM-TL as 3D DCT, and denote such a method as BM-DCT. Table 1 summarizes the key attributes of the SALT denoising, as well as other competing methods.

Denoising results. To evaluate the performance of the denoising schemes, we measure the peak signal-to-noise-ratio (PSNR) in decibel (dB), which is computed between the noiseless reference and the denoised video. Table 2 and 3 list the average denoised PSNRs over videos from TUT and ASU (excluding the 10 videos used for tuning) datasets, obtained by our proposed SALT video denoising method, as well as the eight competing methods. The proposed SALT video denoising method provides PSNR improvements (averaged over all 26 testing videos from both datasets) of 1.3 dB, 1.2 dB, 1.0 dB, 1.6 dB, and 3.6 dB, over the VBM4D, VBM3D, VIDOSAT, sKSVD, and fBM3D denoising methods, respectively. The proposed SALT denoising method consistently provides better PSNRs than all of the competing methods for almost all videos and noise levels, demonstrating state-of-the-art performance in denoising natural videos. Furthermore, we observe that the average PSNR improvements of SALT denoising over BM-LR, BM-TL, and BM-DCT are 0.2 dB, 0.6 dB, and 3.1 dB, respectively. The empirical evidence indicates that both low-rank and sparse approximations contribute positively to the final denoising quality. Additionally, adaptively learned transform can provide much better data sparse representation, which translates to improved sparse approximation.

Figures 3 and 5 show the frame-by-frame denoised PSNRs and SSIMs, which are obtained using the SALT denoising algorithm for the video *Gbicycle* (example from TUT dataset) and *Stefan* (example from ASU dataset) respectively at $\sigma = 20$, along with the corresponding PSNR and SSIM values for VIDOSAT, VBM3D, and VBM4D. It is clear that SALT outperforms the three competing methods in terms of both PSNRs and SSIMs for all frames. Figure 4 illustrates the visual comparisons of the denoised results, by showing one frame of the denoised *Gbicycle* at $\sigma = 20$ (the clean and noisy frames are shown in Figures

4(a) and (b)), obtained by SALT (see Figure 4(c)) and VIDOSAT (see Figure 4(d)). The denoised frame by SALT preserves more details while VIDOSAT generates undesired artifacts, *e.g.*, the zoomed-in region in the red and blue boxes. It is also evident that the denoised frame by VIDOSAT exhibits higher reconstruction error than that by SALT, especially around the moving objects (see Figure 4(e) and (f)). Similarly in Figure 6, we observe better denoised result by SALT compared to VBM4D.

6. Conclusion

We propose an efficient and scalable online video denoising method called SALT. Our method groups similar noisy patches into tensors, adaptively learns a sparsifying transform, and cleans the patches jointly by adaptive sparse and low-rank approximations. Denoising experiments show that our method outperforms competing methods consistently, sometimes by a sizable margin.

References

[1] M. Aharon, M. Elad, and A. Bruckstein. K-SVD : An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006.

[2] A. Buades, B. Coll, and J.-M. Morel. Nonlocal image and movie denoising. *International Journal of Computer Vision*, 76(2):123–139, 2007.

[3] A. Buades, J.-L. Lisani, and M. Miladinovic. Patch-based video denoising with optical flow estimation. *IEEE Transactions on Image Processing*, 25(6):2573–2586, 2016.

[4] K. Dabov, A. Foi, and K. Egiazarian. Video denoising by sparse 3d transform-domain collaborative filtering. In *Signal Processing Conference, 2007 15th European*, pages 145–149, Sept 2007.

[5] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, 16(8):2080–2095, 2007.

[6] W. Dong, G. Li, G. Shi, X. Li, and Y. Ma. Low-rank tensor approximation with laplacian scale mixture modeling for multiframe image denoising. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 442–449, 12 2015.

[7] W. Dong, X. Li, L. Zhang, and G. Shi. Sparsity-based image denoising via dictionary learning and structural clustering. In *IEEE Conf. Comput. Vision and Pattern Recognition (CVPR 2011)*, pages 457–464, June 2011.

[8] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, 15(12):3736–45, Dec. 2006.

[9] H. Guo and N. Vaswani. Video denoising via online sparse and low-rank matrix decomposition. In *2016 IEEE Statistical Signal Processing Workshop (SSP)*, pages 1–5, 6 2016.

[10] H. Hu, J. Froment, and Q. Liu. Patch-based low-rank minimization for image denoising. *arXiv preprint arXiv:1506.08353*, 2015.

[11] M. Maggioni, G. Boracchi, A. Foi, and K. Egiazarian. Video denoising, deblocking, and enhancement through separable 4-d nonlocal spatiotemporal transforms. *IEEE Transactions on Image Processing*, 21(9):3952–3966, 2012.

[12] M. Maggioni, V. Katkovnik, K. Egiazarian, and A. Foi. Nonlocal transform-domain filter for volumetric data denoising and reconstruction. *IEEE Transactions on Image Processing*, 22(1):119–133, 2013.

[13] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-local sparse models for image restoration. In *IEEE 12th Int. Conf. Comput. Vision (ICCV 2009)*, pages 2272–2279, Sept 2009.

[14] B. Natarajan. Sparse approximate solutions to linear systems. *SIAM journal on computing*, 24(2):227–234, 1995.

[15] D. Needell and J. Tropp. CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. *Appl. Comput. Harmon. Anal.*, 26(3):301–321, May 2009.

[16] Y. Pati, R. Rezaiifar, and P. Krishnaprasad. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers*, pages 40–44, 1993.

[17] Y. Peng, D. Meng, Z. Xu, C. Gao, Y. Yang, and B. Zhang. Decomposable nonlocal tensor dictionary learning for multispectral image denoising. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2949–2956, 6 2014.

[18] N. Rajpoot, Z. Yao, and R. Wilson. Adaptive wavelet restoration of noisy video sequences. In *Proc. IEEE Int. Conf. Image Proc., ICIP*, volume 2, pages 957–960, 2004.

[19] S. Ravishankar and Y. Bresler. Closed-form solutions within sparsifying transform learning. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5378–5382. IEEE, 2013.

[20] S. Ravishankar and Y. Bresler. Learning sparsifying transforms. *IEEE Transactions on Signal Processing*, 61(5):1072–1086, 2013.

[21] S. Ravishankar and Y. Bresler. Sparsifying transform learning with efficient optimal updates and convergence guarantees. *IEEE Transactions on Signal Processing*, 63(9):2389–2404, 2015.

[22] S. Ravishankar, B. Wen, and Y. Bresler. Online sparsifying transform learningpart i: Algorithms. *IEEE Journal of Selected Topics in Signal Processing*, 9(4):625–636, 2015.

[23] R. Rubinstein, M. Zibulevsky, and M. Elad. Double sparsity: Learning sparse dictionaries for sparse signal approximation. *IEEE Transactions on Signal Processing*, 58(3):1553–1564, 2010.

[24] D. Rusanovskyy and K. Egiazarian. Video denoising algorithm in sliding 3d dct domain. In *Proc. Advanced Concepts for Intelligent Vision Systems*, pages 618–625, 2005.

[25] P. Seeling and M. Reisslein. Video traffic characteristics of modern encoding standards: H.264/AVC with SVC and MVC extensions and h.265/HVEC. *The Scientific World Journal*, 2014:1–16, 2014.

[26] I. Tosic and P. Frossard. Dictionary learning. *IEEE Signal Processing Magazine*, 28(2):27–38, Mar 2011.

[27] B. Wen, S. Ravishankar, and Y. Bresler. Structured overcomplete sparsifying transform learning with convergence guarantees and applications. *International Journal of Computer Vision*, 114(2-3):137–167, 2015.

[28] B. Wen, S. Ravishankar, and Y. Bresler. Video denoising by online 3d sparsifying transform learning. In *IEEE International Conference on Image Processing (ICIP)*, pages 118–122. IEEE, 2015.

[29] B. Wen, S. Ravishankar, and Y. Bresler. Learning flipping and rotation invariant sparsifying transforms. In *Image Processing (ICIP), 2016 IEEE International Conference on*, pages 3857–3861. IEEE, 2016.

[30] J. Zhang, D. Zhao, and W. Gao. Group-based sparse representation for image restoration. *IEEE Transactions on Image Processing*, 23(8):3336–3351, Aug 2014.