ADAPTIVE NONLOCAL AND STRUCTURED SPARSE SIGNAL MODELING AND APPLICATIONS

BY

BIHAN WEN

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2018

Urbana, Illinois

ABSTRACT

Features based on sparse representation, especially using the synthesis dictionary model, have been heavily exploited in signal processing and computer vision. Many applications such as image and video denoising, inpainting, demosaicing, super-resolution, magnetic resonance imaging (MRI), and computed tomography (CT) reconstruction have been shown to benefit from adaptive sparse signal modeling. However, synthesis dictionary learning typically involves expensive sparse coding and learning steps. Recently, sparsifying transform learning received interest for its cheap computation and its optimal updates in the alternating algorithms. Prior works on transform learning have certain limitations, including (1) limited model richness and structure for handling diverse data, (2) lack of non-local structure, and (3) lack of effective extension to high-dimensional or streaming data. This dissertation focuses on advanced data-driven sparse modeling techniques, especially with nonlocal and structured sparse signal modeling.

In the first work of this dissertation, we propose a methodology for learning, dubbed Flipping and Rotation Invariant Sparsifying Transforms (FRIST), to better represent natural images that contain textures with various geometrical directions. The proposed alternating FRIST learning algorithm involves efficient optimal updates. We provide a convergence guarantee, and demonstrate the empirical convergence behavior of the proposed FRIST learning approach. Preliminary experiments show the promising performance of FRIST learning for image sparse representation, segmentation, denoising, robust inpainting, and compressed sensing-based magnetic resonance image reconstruction.

Next, we present an online high-dimensional sparsifying transform learning method for spatio-temporal data, and demonstrate its usefulness with a novel video denoising framework, dubbed VIDOSAT. The proposed method is based on our previous work on online sparsifying transform learning, which has low computational and memory costs, and can potentially handle streaming video. By combining with a block matching (BM) technique, the learned model can effectively adapt to video data with various motions. The patches are constructed either from corresponding 2D patches in successive frames or using an online block matching technique. The proposed online video denoising requires little memory and offers efficient processing. Numerical experiments are used to analyze the contribution of the various components of the proposed video denoising scheme by "switching off" these components - for example, fixing the transform to be 3D DCT, rather than a learned transform. Other experiments compare to the performance of prior schemes such as dictionary learning-based schemes, and the state-of-the-art VBM3D and VBM4D on several video data sets, demonstrating the promising performance of the proposed methods.

In the third part of the dissertation, we propose a joint sparse and low-rank model, dubbed STROLLR, to better represent natural images. Patch-based methods exploit local patch sparsity, whereas other works apply low-rankness of grouped patches to exploit image non-local structures. However, using either approach alone usually limits performance in image restoration applications. In order to fully utilize both the local and non-local image properties, we develop an image restoration framework using a transform learning scheme with joint low-rank regularization. The approach owes some of its computational efficiency and good performance to the use of transform learning for adaptive sparse representation rather than the popular synthesis dictionary learning algorithms, which involve approximation of NP-hard sparse coding and expensive learning steps. We demonstrate the proposed framework in various applications to image denoising, inpainting, and compressed sensing based magnetic resonance imaging. Results show promising performance compared to state-of-the-art competing methods.

Last, we extend the effective joint sparsity and low-rankness model from image to video applications. 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 the best of our knowledge, this is the first work that combines adaptive sparsity and low-rankness for video denoising, and the first work that solves 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.

To my wife and parents, for their love and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Sparse Modeling

Sparse representation of natural signals in a certain transform domain or dictionary has been widely exploited. Various sparse signal models, such as the synthesis model [1, 2] and the transform model [3], have been studied. It has been shown in many image processing and vision tasks, including image and video denoising, that data-adaptive representations usually lead to superior performance [4–6].

1.1.1 Synthesis Model

The popular *synthesis model* suggests that a signal $y \in \mathbb{R}^n$ can be sparsely represented as $y = Dx + \eta$, where $D \in \mathbb{R}^{n \times m}$ is a synthesis dictionary, $x \in \mathbb{R}^m$ is a sparse code, and η is small approximation error in the signal domain. Synthesis dictionary learning methods [7, 8] that adapt the dictionary based on training data typically involve a synthesis sparse coding step which is, however, NP-hard [9], so that approximate solutions [10–12] are widely used.

Various dictionary learning algorithms [7, 13–15] have been proposed and are popular in numerous applications such as denoising, inpainting, deblurring, and demosaicing [4,16,17]. For example, the well-known K-SVD method [8] generalizes the K-means clustering process to a dictionary learning algorithm, and alternates between updating the sparse codes of training signals (sparse coding step) and the dictionary (dictionary or codebook update step). K-SVD updates both the dictionary atoms (columns) and the non-zero entries in the sparse codes (with fixed support) in the dictionary update step using singular value decompositions (SVD). However, the dictionary learning algorithms such as K-SVD are usually computationally expensive for large-scale problems. Moreover, methods such as KSVD lack convergence guarantees, and can get easily caught in local minima, or saddle points [18].

1.1.2 Transform Model and Transform Learning

The alternative *transform model* suggests that the signal y is approximately sparsifiable using a transform $W \in \mathbb{R}^{m \times n}$, i.e., $Wy = x + e$, with $x \in \mathbb{R}^m$ sparse and e a small approximation error in the transform domain (rather than in the signal domain). It is well known that natural images are sparsifiable by analytical transforms such as the discrete cosine transform (DCT), or wavelet transform [19]. Furthermore, recent works proposed learning square sparsifying transforms (SST) [20], which turn out to be advantageous in various applications such as image denoising, magnetic resonance imaging (MRI), and computed tomography (CT) [5, 20–22]. Alternating minimization algorithms for learning SST have been proposed with cheap and closed-form updates [23].

Since SST learning is restricted to one adaptive square transform for all the data, the diverse patches of natural images, or dynamically changing data, may not be sufficiently sparsified in the SST model.

1.2 Learning Structured Transform for Diverse Data

Recent work focused on learning a union of unstructured sparsifying transforms [5, 24], dubbed OCTOBOS (for OverComplete TransfOrm with BlOck coSparsity constraint – cf. [5]), to sparsify images with diverse contents, features and textures. Given a signal $y \in \mathbb{R}^n$ and a union of transforms $\{W_k\}_{k=1}^K$, where each $W_k \in \mathbb{R}^{m \times n}$, the OCTOBOS model selects the best matching transform for y as the one providing the minimum modeling error. The OCTOBOS sparse coding problem is the following:

$$\text{(P0)} \quad \min_{1 \le k \le K} \min_{x^k} \left\| W_k \, y - x^k \right\|_2^2 \;\; s.t. \;\; \left\| x^k \right\|_0 \le s \;\; \forall \; k,$$

where x^k denotes the sparse representation for y in the transform W_k, the ℓ_0 "norm" counts the number of non-zeros in a vector, and s is a given sparsity level. However, learning such an unstructured OCTOBOS model [5] (that has many free parameters), especially from noisy or limited data, could suffer from overfitting to noise/artifacts, thereby degrading performance in various applications and inverse problem settings.

Instead, we consider the use of transformation symmetries to constrain the multiple learned transforms, thus reducing the number of free parameters and avoiding overfitting. While previous works exploited transformation symmetries in synthesis model sparse coding [25], and applied rotational operators with analytical transforms [26], the usefulness of the rotational invariance property in learning adaptive sparse signal models has not been explored. Here, we propose a Flipping and Rotation Invariant Sparsifying Transform (FRIST) learning scheme, and show that it can provide better sparse representations by capturing the "optimal" orientations of patches in natural images. As such, it serves as an effective regularizer for image recovery in various inverse problems. Preliminary experiments in this dissertation show the usefulness of adaptive sparse representation by FRIST for image sparse representation, segmentation, denoising, robust inpainting, and compressed sensing-based magnetic resonance image (MRI) reconstruction with promising performances.

1.3 Online Transform Learning for High-dimensional Data

Prior work on batch transform learning [5, 20, 23, 24, 27] adapted the transform using all the training data, which usually renders the batch method computationally and memory inefficient. Our recent work [28, 29] introduced online sparsifying transform learning (OSTL) which iteratively adapts the sparsifying transform and sparse codes for signals that arrive sequentially. Preliminary results demonstrated the usefulness of online adaptive 2D transform for large-scale image denoising [28]. Compared to the work of online synthesis dictionary learning [15], the online adaptation of the sparsifying transform allows for much cheaper computations, and thus is more suitable of extending to high-dimensional data applications.

While the data-driven adaptation of synthesis dictionaries for the purpose of denoising image sequences or volumetric data [30, 31] has been studied in some recent papers, the usefulness of learned sparsifying transforms has not been explored in these applications. Video data typically contains correlation along the temporal dimension, which will not be captured by learning sparsifying transforms for the 2D patches of the video frames. We focus on video denoising using high-dimensional online transform learning. We refer to our proposed framework as VIdeo Denoising by Online SpArsifying Transform learning (VIDOSAT). Spatio-temporal (3D) patches are constructed using local 2D patches of the corrupted video, and the sparsifying transform is adapted to these 3D patches on-the-fly. To the best of our knowledge, this is the first video denoising method using online sparse signal modeling, by applying high-dimensional sparsifying transform learning for spatio-temporal data.

1.4 Image Restoration via Joint Sparsifying Transform Learning and Low-Rank Modeling

Image reconstruction refers to the process of forming an image from a collection of measurements. Despite today's vast improvement in camera sensors, digital images are often still corrupted by severe noise in low-light conditions. Furthermore, in modern computed imaging applications, in order to reduce the system complexity, data-acquisition time, or radiation dose, it is usually required to reconstruct high-quality images from incomplete or corrupted measurements. Under such settings, image reconstruction corresponds to a challenging inverse problem. We aim to estimate the underlying image x from its degraded / noisy measurement y, which has the general form of $y = A\,x + e$, where A and e denote the sensing operator and additive noise, respectively. This framework encompasses various important problems, including image denoising, deblurring, inpainting, super-resolution, compressed sensing (CS), and more advanced linear computed imaging modalities. For such problems, and especially for those that are ill-posed, an effective regularizer is key to a successful image reconstruction algorithm. Most of the popular methods take advantage of either sparsity or non-local image structures.

It is well known that natural images contain local structures, such that image patches are typically sparsifiable or compressible under certain transforms, or over certain dictionaries. Early works exploited the sparsity in a fixed transform domain, e.g. discrete cosine transform (DCT) [32, 33] and wavelets [34]. Comparing to fixed sparse models, recent works have shown that the data-driven adaption of sparse signal models leads to promising results in various image recovery problems [5, 35–39]. Among them, synthesis dictionary learning [35, 36] is the most popular adaptive sparse modeling technique. However synthesis model based methods typically involve the approximate solution of an NP-hard sparse coding step [9]. The widely used approximate methods [36, 40] are not efficient for large-scale problems.

As an alternative, the *transform model* provides cheap and exact sparse coding. Recent works on sparsifying transform learning proposed efficient learning algorithms with convergence guarantees [5, 38], which turn out to be advantageous in applications including video denoising [6, 41], magnetic resonance imaging (MRI) [42–44], and computational tomography (CT) [45], with state-of-the-art performances.

Apart from sparsity, images also contain non-local structures such as self-similarity: patches are typically similar to other non-local structures within the same image. Recent image restoration algorithms investigated the grouping of similar patches, and exploited the correlation within each group [32, 37, 46–52]. Among them, the algorithms based on low-rank (LR) modeling have demonstrated superior performance in image recovery tasks [48, 50, 51]. A successful approach of this nature comprises four major steps:

1. For each overlapping image patch u_i, apply block matching to find its similar patches.

2. Construct a data matrix U_i whose columns are the vectorized patches closest to u_i.

3. Denoise U_i by calculating its low-rank approximation.

4. Aggregate the denoised U_i's to form the image estimate.

Similar to adopting sparsity-promoting "norms" as penalties in sparse coding, several types of norms have been introduced to impose low-rankness, including nuclear norm, Schattern p-norm [51], weighted nuclear norm [50], etc.

In summary, transform learning, and low-rankness in groups of similar patches, respectively, capture the sparsity, and non-local self-similarity in natural images. Each of them has been applied as an effective regularizer in various image restoration algorithms. However, to the best of our knowledge, no 2D image recovery algorithm has to date utilized both transform learning and low-rank approximation jointly. In this work, we propose a flexible Sparsifying TRansfOrm Learning and Low-Rank (STROLLR) model [39,53]. Instead of using nuclear norms, we directly minimize the rank of data matrix as the regularization term, which leads to a more efficient algorithm.

1.5 Online Tensor Reconstruction Scheme for Video Denoising by Joint Adaptive Sparsity and Low-Rankness

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). 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 [54, 55].

Video denoising presents challenges that are distinct from other multi-frame image data, such as volumetric data (e.g., 3D medical image) or hyperspectral data. Hyperspectral images, in particular, typically exhibit strong correlation in a small spatial window along the spectral dimension [56–58]. In video, however, objects can move throughout or exit the scene, and such long-term correlations may not exist [59]. Furthermore, 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 [6].

Most video denoising methods take advantage of local or non-local structures present in video data. While some of the previous 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 [4–6]. Synthesis dictionary learning is the most well-known adaptive representation learning scheme [4, 8, 60]. Unfortunately, dictionary learning features typically NP-hard sparse coding steps [61], for which commonly-used greedy approximate algorithms still involve relatively expensive computations [40, 62]. 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 [5, 27, 63]. The recent online variants of the transform learning [28] are especially applicable to streaming large scale, or high-dimensional data, and have demonstrated promising performance for video denoising [6].

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 [39, 57]. However, to the best of our knowledge, no video denoising algorithm has to date utilized both data-adaptive sparse and low-rank priors. We introduce an online video denoising scheme called Sparse And Low-rank Tensor (SALT) reconstruction, which exploits both local and non-local structures.

1.6 Organization

The rest of the dissertation is organized as follows. In Chapter 2, we present the FRIST model, and the FRIST learning formulations, associated with efficient learning algorithms to solve the proposed learning problem along with convergence analysis. In Chapter 3, we describe various applications based on FRIST learning, including image denoising, inpainting and compressed sensing-based (MR) image reconstruction. Experimental results demonstrating the promise of FRIST learning, including for image segmentation, sparse representation, denoising, inpainting, and MRI are illustrated. In Chapter 4, we introduce an online denoising formulation based on the online and mini-batch transform learning [28, 64]. We then propose a video denoising method using high-dimensional online transform learning, which we refer to as VIdeo Denoising by Online SpArsifying Transform learning (VIDOSAT).

In Chapter 5, we propose an image restoration framework using multiple complementary regularizers. We propose a joint adaptive patch sparse and group low-rank model, dubbed STROLLR, for better representing natural images using transform learning. In Chapter 6, we introduce an online video denoising scheme called Sparse And Low-rank Tensor (SALT) reconstruction, which exploits both local and non-local structures, and achieves state-of-the-art video denoising results.

CHAPTER 2

FRIST: FLIPPING AND ROTATION INVARIANT SPARSIFYING TRANSFORM LEARNING

2.1 FRIST Model and Its Learning Formulation

2.1.1 FRIST Model

The learning of the sparsifying transform model [20] has been proposed recently. Extending this approach, we propose a *FRIST model* that first applies a flipping and rotation (FR) operator $\Phi \in \mathbb{R}^{n \times n}$ to a signal $y \in \mathbb{R}^n$, and models Φy as approximately sparsifiable by some sparsifying transform $W \in \mathbb{R}^{m \times n}$, i.e., $W \Phi y = x + e$, with $x \in \mathbb{R}^m$ sparse, and e is a small deviation term. A finite set of flipping and rotation operators $\{\Phi_k\}_{k=1}^{K}$ is considered, and the sparse coding problem in the FRIST model is as follows:

$$(\text{P1}) \quad \min_{1 \leq k \leq K} \, \min_{x^k} \, \left\| W \, \Phi_k \, y - x^k \right\|_2^2 \;\; \text{s.t.} \;\; \left\| x^k \right\|_0 \leq s \;\; \forall \, k.$$

Thus, x^k denotes the sparse code of $\Phi_k \, y$ in the transform W domain, with maximum sparsity s. Equivalently, the optimal $\hat{x}^{\hat{k}}$ is called the sparse code in the FRIST domain. We further decompose the FR matrix as $\Phi_k \triangleq G_q \, F$, where F can be either an identity matrix or (for 2D signals) a left-to-right flipping permutation matrix. Though there are various methods of formulating the rotation operator G with arbitrary angles [65, 66], rotating image patches by an angle θ that is not a multiple of 90° requires interpolation, and may result in misalignment with the pixel grid. Here, we adopt the matrix $G_q \triangleq G(\theta_q)$ that permutes the pixels in an image patch approximating rotation by angle θ_q without interpolation. Constructions of such $\{G_q\}$ have been proposed before [26, 67, 68]. With such implementation, the number of possible permutations $\{G_q\}$ denoted by $\tilde{Q}$ is finite and grows linearly with the signal dimension n. Accounting for the flipping operation, the total number of possible FR operators is $\tilde{K} = 2\tilde{Q}$.

In practice, one can select a subset $\{\Phi_k\}_{k=1}^{K}$, containing $K < \tilde{K}$ of FR candidates, from which the optimal $\hat{\Phi} = \Phi_{\hat{k}}$ is chosen in (P1). For each Φ_k, the optimal sparse code $\hat{x}^k$ in Problem (P1) can be solved exactly as $\hat{x}^k = H_s(W\Phi_k y)$, where $H_s(\cdot)$ is the projector onto the s-ℓ_0 ball [69], i.e., $H_s(b)$ zeros out all but the s elements of largest magnitude in $b \in \mathbb{R}^m$. The optimal FR operator $\Phi_{\hat{k}}$ is selected to provide the smallest sparsification (modeling) error $\|W\,\Phi_k\,y - H_s(W\Phi_k y)\|_2^2$ over k in Problem (P1).

The FRIST model can be interpreted as a structured union-of-transforms model, or a structured OCTOBOS model [5, 24]; i.e., compared to a general overcomplete dictionary or OCTOBOS, FRIST is much more constrained, with fewer free parameters. In particular, the OCTOBOS model involves a collection (or union) of sparsifying transforms $\{W_k\}_{k=1}^{K}$ such that for each candidate signal, there is a transform in the collection that is best matched (or that provides the lowest sparsification error) to the signal. The FRIST model involves a collection of transforms $\{W\Phi_k\}_{k=1}^{K}$ (as in Problem (P1)) that are related to each other by rotation and flip operators (and involving a single parent transform W). The transforms in the collection all share a common transform W. We call the shared common transform W the **parent transform**, and each generated $W_k = W\Phi_k$ is called a **child transform**. Clearly, the collection of transforms in FRIST is more constrained than in the OCTOBOS model. The constraints that are imposed by FRIST are devised to reflect commonly observed properties of natural image patches, i.e., image patches tend to have edges and features at various orientations, and optimally rotating (or flipping) each patch would allow it to be well-sparsified by a common sparsifying transform W (as in (P1)). This property turns out to be useful in inverse problems such as denoising and inpainting, preventing the overfitting of the model in the presence of limited or highly corrupted data or measurements.

Problem (P1) is similar to the OCTOBOS sparse coding problem [5], where each W_k corresponds to a block of OCTOBOS. Similar to the clustering procedure in OCTOBOS, Problem (P1) matches a signal y to a particular child transform W_k with its directional FR operator Φ_k. Thus, FRIST is potentially capable of automatically clustering a collection of signals (e.g., image patches), but according to their geometric orientations. When the parent transform W is unitary, FRIST is also equivalent to an overcomplete synthesis dictionary with block sparsity [70], with W_k^T denoting the kth block of the equivalent overcomplete dictionary.

2.1.2 FRIST Learning Formulation

Generally, the parent transform W can be overcomplete [5, 20, 21]. In this work, we restrict ourselves to learning FRIST with a square parent transform W (i.e., $m = n$), which leads to a highly efficient learning algorithm with optimal updates. Note that the FRIST model is still overcomplete, even with a square parent W, because of the additional FR operators. Given the training data $Y \in \mathbb{R}^{n \times N}$, we formulate the FRIST learning problem as follows:

$$\text{(P2)} \min_{W, \{X_i\}, \{C_k\}} \sum_{k=1}^{K} \left\{ \sum_{i \in C_k} \|W\Phi_k Y_i - X_i\|_2^2 \right\} + \lambda Q(W)$$
$$\text{s.t.} \;\; \|X_i\|_0 \leq s \;\; \forall\, i, \;\; \{C_k\} \in \Gamma\,,$$

where $\{X_i\}$ represent the FRIST-domain sparse codes of the corresponding columns $\{Y_i\}$ of Y, and $X \in \mathbb{R}^{n \times N}$ with columns X_i denotes the sparse code matrix of Y. The $\{C_k\}_{k=1}^{K}$ indicate a clustering of the signals $\{Y_i\}_{i=1}^{N}$ such that C_j contains the indices of signals in the jth cluster (corresponding to the child transform $W\Phi_j$), and each signal Y_i is associated with exactly one FR operator Φ_k. The set Γ is the set of all possible partitions (into K subsets) of the set of integers $\{1, 2, ..., N\}$, which enforces all of the C_k's to be disjoint [5].

Problem (P2) is to minimize the FRIST learning objective that includes the modeling or sparsification error $\sum_{k=1}^{K} \left\{ \sum_{i \in C_k} \|W\Phi_k Y_i - X_i\|_2^2 \right\}$ for Y as well as the regularizer $Q(W) = -\log\, |\det W| + \|W\|_F^2$ to prevent trivial solutions [20]. Here, the log-determinant penalty $-\log\, |\det W|$ enforces full rank on W, and the $\|W\|_F^2$ penalty helps remove a 'scale ambiguity' in the solution. The regularizer $Q(W)$ fully controls the condition number and scaling of the learned parent transform [20]. The regularizer weight λ is chosen as $\lambda = \lambda_0 \|Y\|_F^2$, in order to scale with the first term in (P2). Previous works [20] showed that the condition number and spectral norm of the optimal parent transform $\hat{W}$ approach 1 and $1/\sqrt{2}$ respectively, as $\lambda_0 \to \infty$ in (P2).

Problem (P2) imposes an ℓ_0 sparsity constraint $\|X_i\|_0 \leq s$ on the sparse code of each signal or image patch. One can also impose an overall sparsity constraint on the entire sparse code matrix X to allow variable sparsity levels across the signals (see Section 3.3). Alternatively, a sparsity penalty method can be used, instead of imposing sparsity constraints, which also leads to efficient algorithms (see Section 3.2.2).

Note that in the overcomplete synthesis dictionary model, sparse coding with an ℓ_0 "norm" constraint is NP-hard in general, and convex ℓ_1 relaxations of the synthesis sparse coding problem have been popular, and solving such an ℓ_1 (relaxed) problem is known to provide the sparsest solution under certain conditions on the dictionary. On the other hand, in the sparsifying transform model (including in the FRIST model), the sparse coding problem can be solved exactly and cheaply by thresholding operations, irrespective of whether an ℓ_0 penalty or constraint (resulting in hard thresholding-type solution) or an ℓ_1 penalty (resulting in soft thresholding solution) is used. Thus there is not really a computational benefit for employing the ℓ_1 norm in the case of the transform model. More importantly, in practice, we have observed that transform learning with ℓ_0 sparsity leads to better performance in applications compared to ℓ_1 norm-based learning.

2.2 FRIST Learning Algorithm and Convergence Analysis

2.2.1 FRIST Learning Algorithm

We propose an efficient algorithm for solving (P2), which alternates between a *sparse coding and clustering* step and a *transform update* step.

Sparse Coding and Clustering

Given the training matrix Y, and a fixed parent transform W, we solve the following Problem (P3) for the sparse codes and clusters:

$$\text{(P3)} \quad \min_{\{C_k\},\{X_i\}} \sum_{k=1}^{K} \sum_{i \in C_k} \left\| W \Phi_k Y_i - X_i \right\|_2^2 \quad s.t. \;\; \left\| X_i \right\|_0 \leq s \;\; \forall\, i, \;\; \{C_k\} \in \Gamma.$$

The modeling error $\|W\Phi_k Y_i - X_i\|_2^2$ serves as the clustering measure corresponding to signal Y_i, where the best sparse code with FR permutation Φ_k [1] is $X_i = H_s(W\Phi_k Y_i)$. Problem (P3) is clearly equivalent to finding the "optimal" FR permutation $\Phi_{\hat{k}_i}$ independently for each data vector Y_i by solving the following optimization problem:

$$\min_{1 \leq k \leq K} \|W\Phi_k Y_i - H_s(W\Phi_k Y_i)\|_2^2 \quad \forall\, i\,, \tag{2.1}$$

where the minimization over k for each Y_i determines the optimal $\Phi_{\hat{k}_i}$, or the cluster $C_{\hat{k}_i}$ to which Y_i belongs. The corresponding optimal sparse code for Y_i in (P3) is thus $\hat{X}_i = H_s(W\Phi_{\hat{k}_i} Y_i)$. Given the sparse code, [2] one can also easily recover a least squares estimate of each signal as $\hat{Y}_i = \Phi_{\hat{k}_i}^T W^{-1}\hat{X}_i$. Since the Φ_k's are permutation matrices, applying and computing Φ_k^T (which is also a permutation matrix) is cheap.

Transform Update Step

Here, we solve for W in (P2) with fixed $\{C_k\}$ and $\{X_i\}$, which leads to the following problem:

$$\text{(P4)} \quad \min_{W} \left\| W\tilde{Y} - X \right\|_F^2 + \lambda Q(W)\,,$$

where $\tilde{Y} = \left[\Phi_{\hat{k}_1} Y_1 \mid \Phi_{\hat{k}_2} Y_2 \mid \ \ldots \ \mid \Phi_{\hat{k}_N} Y_N\right]$ contains signals after applying their optimal (as determined in the preceding sparse coding and clustering step) FR operations, and the columns of X are the corresponding sparse codes X_i's. Problem (P4) has a simple solution involving a singular value decomposition (SVD), which is similar to the transform update step in SST [23]. We first decompose the positive-definite matrix $\tilde{Y}\tilde{Y}^T + \lambda I_n = UU^T$ (e.g., using Cholesky decomposition). Then, denoting the full singular value decomposition (SVD) of the matrix $U^{-1}\tilde{Y}X^T = S\Sigma V^T$, where $S, \Sigma, V \in \mathbb{R}^{n\times n}$, an optimal transform $\hat{W}$ in (P4) is

$$\hat{W} = 0.5V\left(\Sigma + (\Sigma^2 + 2\lambda I_n)^{\frac{1}{2}}\right) S^T U^{-1}\,, \tag{2.2}$$

[1]The FR operator is $\Phi_k = G_q F$, where both G_q and F are permutation matrices. Therefore the composite operator Φ_k is a permutation matrix.

[2]The sparse code includes the value of $\hat{X}_i$, as well as the membership index $\hat{k}_i$ which adds just $\log_2 K$ bits to the code storage.

where $(\cdot)^{\frac{1}{2}}$ above denotes the positive definite square root, and I_n is the $n \times n$ identity.

Initialization Insensitivity and Cluster Elimination

Unlike the previously proposed OCTOBOS learning algorithm [5], which requires initialization of the clusters using heuristic methods such as K-means, the FRIST learning algorithm only needs initialization of the parent transform W. In Section 3.4.1, numerical results demonstrate the fast convergence of the proposed FRIST learning algorithm, which is typically insensitive to the parent transform initialization. In practice, we apply a heuristic cluster elimination strategy in the FRIST learning algorithm, to select the desired K FR operators. In the first iteration, all possible FR operators Φ_k's [26,67] (i.e., all possible child transforms W_k's) are considered for sparse coding and clustering. After each clustering step, the learning algorithm eliminates half of the operators with smallest cluster sizes, until the number of selected operators drops to K, which only takes a few iterations. For the rest of the iterations, the algorithm only considers the selected K Φ_k's in the sparse coding and clustering steps.

Computational Cost Analysis

The *sparse coding and clustering* step computes the optimal sparse codes and clusters, with $O(Kn^2N)$ cost. In the transform update step, we compute the optimal solution for the square parent transform in (P4). The cost of computing this solution scales as $O(n^2N)$, assuming $N \gg n$, which is cheaper than the sparse coding step. Thus, the overall computational cost per iteration of FRIST learning using the proposed alternating algorithm scales as $O(Kn^2N)$, which is typically lower than the cost per iteration of the overcomplete K-SVD learning algorithm [8], with the number of the dictionary atoms $m = Kn$. FRIST learning also converges quickly in practice as illustrated later in Section 3.4.1. The computational costs per iteration of SST, OCTOBOS, FRIST, and KSVD learning are summarized in Table 2.1.

Table 2.1: Computational cost comparison between SST ($W \in \mathbb{R}^{n\times n}$), OCTOBOS ($K$ clusters, each $W_k \in \mathbb{R}^{n\times n}$), FRIST and KSVD ($D \in \mathbb{R}^{n\times m}$) learning. N is the amount of training data.

	SST.	OCTOBOS	FRIST	KSVD
Cost	$O(n^2N)$	$O(Kn^2N)$	$O(Kn^2N)$	$O(mn^2N)$

2.2.2 Convergence Analysis

We analyze the convergence behavior of the proposed FRIST learning algorithm for (P2), assuming that every step in the algorithms (such as SVD) is computed exactly.

Notation

Problem (P2) is formulated with sparsity constraints, which is equivalent to an unconstrained formulation with sparsity barrier penalties $\phi(X_i)$ (which equals $+\infty$ when the constraint is violated, and is zero otherwise). Thus, the objective function of Problem (P2) can be rewritten as

$$f\left(W, X, \Lambda\right) = \sum_{k=1}^{K} \sum_{i \in C_k} \left\{ \|W\Phi_k Y_i - X_i\|_2^2 + \phi(X_i) \right\} + \lambda Q(W)\,, \tag{2.3}$$

where $\Lambda \in \mathbb{R}^{1\times N}$ is the vector whose ith element $\Lambda_i \in \{1, .., K\}$ denotes the cluster label k corresponding to the signal Y_i, i.e., $i \in C_k$. We use $\{W^t, X^t, \Lambda^t\}$ to denote the output in each iteration (consisting of the sparse coding and clustering, and transform update steps) t, generated by the proposed FRIST learning algorithm.

Main Results

Since FRIST can be interpreted as a structured OCTOBOS, the convergence results of the FRIST learning algorithm take a form similar to those obtained for the OCTOBOS learning algorithm [5] in our recent work. The convergence result for the FRIST learning algorithm for (P2) is summarized in the following theorem and corollaries.

Theorem 1. *For each initialization (W^0, X^0, Λ^0), the following conclusions hold:*

(i) The objective sequence $\{f^t = f(W^t, X^t, \Lambda^t)\}$ in the FRIST learning algorithm is monotone decreasing, and converges to a finite value, $f^* = f^*(W^0, X^0, \Lambda^0)$.

(ii) The iterate sequence $\{W^t, X^t, \Lambda^t\}$ is bounded, with all of its accumulation points equivalent, i.e., achieving the exact same value f^*.

(iii) Every accumulation point (W, X, Λ) of the iterate sequence satisfies the following partial global optimality conditions:

$$(X, \Lambda) \in \underset{\tilde{X},\tilde{\Lambda}}{\arg\min}\ f\left(W, \tilde{X}, \tilde{\Lambda}\right) \tag{2.4}$$

$$W \in \underset{\tilde{W}}{\arg\min}\ f\left(\tilde{W}, X, \Lambda\right) . \tag{2.5}$$

(iv) For each accumulation point (W, X, Λ), there exists $\epsilon = \epsilon(W) > 0$ such that

$$f\left(W + dW, X + \Delta X, \Lambda\right) \geq f\left(W, X, \Lambda\right) = f^*, \tag{2.6}$$

which holds for all $dW \in \mathbb{R}^{n\times n}$ satisfying $\|dW\|_F \leq \epsilon$, and all $\Delta X \in \mathbb{R}^{n\times N}$ satisfying $\|\Delta X\|_\infty < \min_k \min_{i\in C_k} \{\psi_s(W\Phi_k Y_i) : \|W\Phi_k Y_i\|_0 > s\}$. Here, we define $\|\Delta X\|_\infty \triangleq \max_{i,j} |\Delta X_{i,j}|$, and the operator $\psi_s(\cdot)$ returns the s*th largest magnitude in a vector.*

Conclusion *(iv)* provides a partial local optimality condition for each accumulation point with respect to (W, X), where the local perturbation dW in Equation (2.6) is sufficiently small, and ΔX is specified by a finite region, which is determined by a scalar κ that limits the amplitudes of entries in ΔX (i.e., $\|\Delta X\|_\infty < \kappa$). Here, we have $\kappa = \min_k \kappa_k$, and each $\kappa_k = \min_{i\in C_k} \{\psi_s(W\Phi_k Y_i) : \|W\Phi_k Y_i\|_0 > s\}$ is computed by *(i)* choosing the vectors with sparsity $> s$ from $\{W\Phi_k Y_i\}$ where $i \in C_k$, *(ii)* selecting the sth largest magnitude in each of those vectors, and *(ii)* returning the smallest of those values.

Corollary 1. *For a particular initial (W^0, X^0, Λ^0), the iterate sequence in the FRIST learning algorithm converges to an equivalence class of accumulation points, which are also partial minimizers satisfying* (2.4)*,* (2.5)*, and* (2.6)*.*

Corollary 2. *The iterate sequence $\{W^t, X^t, \Lambda^t\}$ in the FRIST learning algorithm is globally convergent (i.e., it converges from any initialization) to the set of partial minimizers of the non-convex objective $f(W, X, \Lambda)$.*

For reasons of space, we only provide an outline of proofs. The conclusion (i) in Theorem 1 is obvious, as the proposed alternating algorithm solves the sub-problem in each step exactly. The proof of Conclusion (ii) follows the same arguments as in the proofs in Lemma 3 and Lemma 5 in [5]. In Conclusion (iii), Condition (2.4) can be proved using the arguments for Lemma 7 from [5], while Condition (2.5) can be proved with the arguments for Lemma 6 from [23]. The last conclusion in Theorem 1 can be shown using arguments similar to those in the proof of Lemma 9 in [23].

Theorem 1 and Corollaries 1 and 2 establish that with any initialization (W^0, X^0, Λ^0), the iterate sequence $\{W^t, X^t, \Lambda^t\}$ generated by the FRIST learning algorithm converges to an equivalence class (corresponding to a common objective value f^* – that may depend on initialization) of partial minimizers of the objective. Note that no assumptions are made about the initialization to establish these results. We leave for future work the investigation of stronger convergence results (e.g., convergence to global minima) with additional assumptions, including on the algorithm initialization, or using a probabilistic analysis framework.

CHAPTER 3

FRIST APPLICATIONS OF INVERSE PROBLEMS

Natural or biomedical images typically contain a variety of directional features and edges; thus, the FRIST model is particularly appealing for applications in image processing and inverse problems. In this chapter, we consider three such applications, namely image denoising, image inpainting, and blind compressed sensing (BCS)-based magnetic resonance imaging (MRI).

3.1 Image Denoising

Image denoising is one of the most fundamental inverse problems in image processing. The goal is to reconstruct a 2D image represented as a vector $y \in \mathbb{R}^P$, from its measurement $z = y + h$, corrupted by a noise vector h. Various denoising algorithms have been proposed recently, with state-of-the-art performance [32, 33]. Similar to previous dictionary and transform learning based image denoising methods [4,5], we propose the following patch-based image denoising formulation using FRIST learning:

$$\text{(P5)} \quad \min_{W, \{y_i, x_i, C_k\}} \sum_{k=1}^{K} \sum_{i \in C_k} \left\{ \|W \Phi_k y_i - x_i\|_2^2 + \tau \, \|R_i \, z - y_i\|_2^2 \right\} + \lambda \, Q(W)$$
$$s.t. \;\; \|x_i\|_0 \leq s_i \;\; \forall \, i, \;\; \{C_k\} \in \Gamma \, ,$$

where $R_i \in \mathbb{R}^{n \times P}$ denotes the patch extraction operator, i.e., $R_i z \in \mathbb{R}^n$ represents the ith overlapping patch of the corrupted image z as a vector. We assume N overlapping patches in total. The data fidelity term $\tau \, \|R_i \, z - y_i\|_2^2$ measures the discrepancy between the observed patch $R_i z$ and the (unknown) noiseless patch y_i, and uses a weight $\tau = \tau_0 / \sigma$ that is inversely proportional to the given noise standard deviation σ [4, 20], and $\tau_0 > 0$. The vector $x_i \in \mathbb{R}^n$ represents the sparse code of y_i in the FRIST domain, with an a priori unknown sparsity level s_i. We follow the previous SST-based and OCTOBOS-based denoising methods [5,69], and impose a sparsity constraint on each y_i.

We propose a simple iterative denoising algorithm based on (P5). Each iteration involves the following steps: (*i*) sparse coding and clustering, (*ii*) sparsity level update, and (*iii*) transform update. Once the iterations complete, we have a denoised image reconstruction step. We initialize the $\{y_i\}$ in (P5) using the noisy image patches $\{R_i z\}$. Step (*i*) is the same as described in Chapter 2. We then update the sparsity levels s_i for all i, similar to the SST learning-based denoising algorithm [69]. With fixed W and clusters $\{C_k\}$, we solve for y_i ($i \in C_k$) in (P5) in the least squares sense,

$$y_i = \Phi_k^T \begin{bmatrix} \sqrt{\tau}\, I \\ W \end{bmatrix}^{\dagger} \begin{bmatrix} \sqrt{\tau}\, v_i \\ H_{s_i}(W v_i) \end{bmatrix} = G_1 v_i + G_2 H_{s_i}(W v_i)\,, \tag{3.1}$$

where G_1 and G_2 are appropriate matrices in the above decomposition, and $v_i \triangleq \Phi_k R_i\, z$ are the rotated/flipped noisy patches, which can be pre-computed in each iteration. We choose the optimal s_i to be the smallest integer that makes the reconstructed y_i in (3.1) satisfy the error condition $\|R_i z - y_i\|_2^2 \leq nC^2\sigma^2$, where C is a constant parameter [69]. Once step (*ii*) is completed, we proceed to the transform update based on the method in Chapter 2. The algorithm alternates between steps (*i*)-(*iii*) for a fixed number of iterations, and eventually the denoised image patches $\{y_i\}$ are obtained using (3.1). Each pixel in the reconstructed patch is projected onto the underlying intensity range (image pixel is typically stored as 8-bit integer, which corresponds to the intensity range $[0, 255]$). The denoised image is reconstructed by averaging the overlapping denoised patches at their respective image locations.

For improved denoising, the algorithm for (P5) is repeated for several passes by replacing z with the most recent denoised image estimate in each pass. The noise standard deviation σ decreases gradually in each such pass, and is found (tuned) empirically [5].

3.2 Image Inpainting

The goal of image inpainting is to recover missing pixels in an image. The given image measurement, with missing pixel intensities set to zero, is denoted as $z = \Xi y + \varepsilon$, where ε is the additive noise on the available pixels, and $\Xi \in \mathbb{R}^{P \times P}$ is a diagonal binary matrix with zeros at locations corresponding to missing pixels. We propose the following patch-based image inpainting formulation using FRIST learning:

$$\text{(P6)} \min_{W,\{y_i,x_i,C_k\}} \sum_{k=1}^{K} \sum_{i \in C_k} \left\{ \|W\Phi_k y_i - x_i\|_2^2 + \tau^2 \|x_i\|_0 + \gamma \|P_i y_i - z_i\|_2^2 \right\} + \lambda\, Q(W),$$

where $z_i = R_i z$ and $y_i = R_i y$. The diagonal binary matrix $P_i \in \mathbb{R}^{n \times n}$ captures the available (non-missing) pixels in z_i. The sparsity penalty $\tau^2 \|x_i\|_0$ is used, which leads to an efficient algorithm. The fidelity term $\gamma \|P_i y_i - z_i\|_2^2$ for the ith patch has the coefficient γ that is chosen inversely proportional to the noise standard deviation σ (in the measured pixels). The parameter τ is chosen proportional to the number of pixels that are missing in z.

Our proposed iterative algorithm for solving (P6) involves the following steps: (i) sparse coding and clustering, and (ii) transform update. Once the iterations complete, we have a (iii) patch reconstruction step. The sparse coding problem with a sparsity penalty has closed-form solution [28], and thus step (i) is equivalent to solving the following problem:

$$\min_{1 \le k \le K} \|W\Phi_k y_i - T_\tau(W\Phi_k y_i)\|_2^2 \quad \forall\, i\,, \tag{3.2}$$

where the hard thresholding operator $T_\tau(\cdot)$ is defined as

$$(T_\tau(b))_j = \begin{cases} 0 & , \ |b_j| < \tau \\ b_j & , \ |b_j| \ge \tau \end{cases}, \tag{3.3}$$

where the vector $b \in \mathbb{R}^n$, and the subscript j indexes its entries. Step (ii) is similar to that in the denoising algorithm in Section 3.1. In the following, we discuss step (iii) by considering two cases.

3.2.1 Ideal Image Inpainting without Noise

In the ideal case when the noise ε is absent, i.e., $\sigma = 0$, the coefficient of the fidelity term $\gamma \to \infty$. Thus the fidelity term can be replaced with hard constraints $P_i\, y_i = z_i\ \forall\, i$. In the noiseless reconstruction step, with fixed $\{x_i, C_k\}$ and W, we first reconstruct each image patch y_i by solving the following linearly constrained least squares problem:

$$\min_{y_i} \|W\Phi_{k_i} y_i - x_i\|_2^2 \ \ s.t.\ P_i\, y_i = z_i\,. \tag{3.4}$$

We define $z_i = P_i y_i \triangleq y_i - e_i$, where $e_i = (I_n - P_i) y_i$. Because Φ_k only rearranges pixels, $\Phi_k e_i$ has the support $\Omega_i = \mathrm{supp}(\Phi_k e_i) = \{j|\,(\Phi_k e_i)_j \neq 0\}$, which is complementary to $\mathrm{supp}(\Phi_k z_i)$. Since the constraint leads to the relationship $y_i = z_i + e_i$ with z_i given, we solve the equivalent minimization problem over e_i as follows:

$$\min_{e_i} \|W\Phi_{k_i} e_i - (x_i - W\Phi_{k_i}\, z_i)\|_2^2 \ \ s.t.\ \mathrm{supp}(\Phi_{k_i} e_i) = \Omega_i\,. \tag{3.5}$$

Here, we define W_{Ω_i} to be the submatrix of W formed by columns indexed in Ω_i, and $(\Phi_k e_i)_{\Omega_i}$ to be the vector containing the non-zero entries of $\Phi_k e_i$. Thus, $W\Phi_k e_i = W_{\Omega_i} (\Phi_k e_i)_{\Omega_i}$, and we define $\xi^i \triangleq \Phi_k e_i$. The reconstruction problem is then re-written as the following unconstrained problem:

$$\min_{\xi^i_{\Omega_i}} \left\|W_{\Omega_i} \xi^i_{\Omega_i} - (x_i - W\Phi_k\, z_i)\right\|_2^2 \ \ \forall\, i\,. \tag{3.6}$$

The above least squares problem has a simple solution given as $\hat{\xi}^i_{\Omega_i} = W^{\dagger}_{\Omega_i}(x_i - W\Phi_k z_i)$. Accordingly, we can calculate $\hat{e}_i = \Phi_k^T \hat{\xi}^i$, and thus the reconstructed patches $\hat{y}_i = \hat{e}_i + z_i$.

3.2.2 Robust Image Inpainting

We now consider the case of noisy z, and propose a robust inpainting algorithm (i.e., for the aforementioned Step (iii)). This is useful because real image measurements are inevitably corrupted with noise [16]. The robust reconstruction step for each patch is to solve the following problem:

$$\min_{y_i} \|W\Phi_{k_i}y_i - x_i\|_2^2 + \gamma \|P_i y_i - z_i\|_2^2 \,. \tag{3.7}$$

Let $\tilde{z}_i \triangleq \Phi_{k_i}z_i$, $u_i \triangleq \Phi_{k_i}y_i$, and $\tilde{P}_i \triangleq \Phi_{k_i}P_i\Phi_{k_i}^T$, where Φ_{k_i} is a permutation matrix. The rotated solution $\hat{u}_i$ in optimization problem (3.7) is equivalent to

$$\hat{u}_i = \arg\min_{u_i} \|Wu_i - x_i\|_2^2 + \gamma \left\|\tilde{P}_i u_i - \tilde{z}_i\right\|_2^2 \,, \tag{3.8}$$

which has a least squares solution $\hat{u}_i = (W^TW + \gamma\tilde{P}_i)^{-1}(W^Tx_i + \gamma\tilde{P}_i\tilde{z}_i)$. As the matrix inversion $(W^TW + \gamma\tilde{P}_i)^{-1}$ is expensive with a cost of $O(n^3)$ for each patch reconstruction, we apply the Woodbury matrix identity and rewrite the solution to (3.8) as

$$\hat{u}_i = [B - B_{\Upsilon_i}^T(\frac{1}{\gamma}I_{q^i} + \Psi_i)^{-1}B_{\Upsilon_i}](W^Tx_i + \gamma\tilde{P}_i\tilde{z}_i) \,, \tag{3.9}$$

where $B \triangleq (W^TW)^{-1}$ can be pre-computed, and the support of $\tilde{z}_i$ is denoted as $\Upsilon_i \triangleq \text{supp}(\tilde{z}_i)$. The scalar $q^i = |\Upsilon_i|$ counts the number of available pixels in z_i. Here, B_{Υ_i} is the submatrix of B formed by B_{Υ_i}-indexed rows, while Ψ_i is the submatrix of B_{Υ_i} formed by B_{Υ_i}-indexed columns. Thus, the matrix inversion $(\frac{1}{\gamma}I_{q^i} + \Psi_i)^{-1}$ has cost of $O((q^i)^3)$, compared to computing $(B + \gamma\tilde{P}_i)^{-1}$ with cost of $O(n^3)$ for the reconstruction of each patch. For an inpainting problem with most pixels missing ($q^i \ll n$), this represents significant savings. On the other hand, with few pixels missing ($q^i \approx n$), a similar procedure can be used with $I - P_i$ replacing P_i. Once $\hat{u}_i$ is computed, the patch in (3.7) is recovered as $\hat{y}_i = \Phi_{k_i}^T\hat{u}_i$.

Similar to Section 3.1, each pixel in the reconstructed patch is projected onto the (underlying) intensity range (e.g., $[0, 255]$ for image pixel stored using 8-bit integer). Eventually, we output the inpainted image by averaging the reconstructed patches at their respective image locations. We perform multiple passes in the inpainting algorithm for (P6) for improved inpainting. In each pass, we initialize $\{y_i\}$ using patches extracted from the most recent inpainted image. By doing so, we indirectly reinforce the dependency between overlapping patches in each pass.

3.3 Blind Compressed Sensing Based MRI

Compressed sensing (CS) enables accurate MRI reconstruction from limited k-space or Fourier measurements [44, 71, 72]. However, CS-based MRI suffers from various artifacts at high undersampling factors, when using non-adaptive or analytical sparsifying transforms [73]. Recent works [44] proposed blind compressed sensing (BCS)-based MR image reconstruction methods using learned signal models, and achieved superior reconstruction results. The terminology *blind compressed sensing* (or image model-blind compressed sensing) is used because the dictionary or sparsifying transform for the underlying image patches is assumed unknown a priori, and is learned simultaneously with the image from the undersampled (compressive) measurements themselves. MR image patches also typically contain variously oriented features [26], which have recently been shown to be well sparsifiable by directional wavelets [68]. As an alternative to approaches involving directional analytical transforms, here, we propose an adaptive FRIST-based approach that can adapt a parent transform W while clustering image patches simultaneously based on their geometric orientations. This leads to more accurate modeling of MR image features.

Similar to the previous TL-MRI work [44], we propose a BCS-based MR image reconstruction scheme using the (adaptive) FRIST model, dubbed FRIST-MRI. For computational efficiency, we restrict the parent W to be a unitary transform in the following. The FRIST-blind image recovery problem with a sparsity constraint is formulated as

$$\text{(P7)} \qquad \min_{W,y,\{x_i,C_k\}} \mu \left\| F_u y - z \right\|_2^2 + \sum_{k=1}^{K} \sum_{i \in C_k} \left\| W \Phi_k R_i y - x_i \right\|_2^2$$
$$s.t.\;\; W^H W = I,\;\; \|X\|_0 \leq s,\;\; \|y\|_2 \leq L,\;\; \{C_k\} \in \Gamma\,,$$

where $W^H W = I$ is the unitary constraint, $y \in \mathbb{C}^P$ is the MR image to be reconstructed, and $z \in \mathbb{C}^M$ denotes the measurements with the sensing matrix $F_u \in \mathbb{C}^{M \times P}$, which is the undersampled (single-coil) Fourier encoding matrix. Here $M \ll P$, so Problem (P7) is aimed to reconstruct an MR image y from highly undersampled measurements z. The constraint $\|y\|_2 \leq L$ with $L > 0$, represents prior knowledge of the signal energy/range. The sparsity term $\|X\|_0$ counts the number of non-zeros in the entire sparse code matrix

X, whose columns are the sparse codes $\{x_i\}$. This sparsity constraint enables variable sparsity levels for individual patches [44].

We use a block coordinate descent approach [44] to solve Problem (P7). The proposed algorithm alternates between (i) sparse coding and clustering, (ii) parent transform update, and (iii) MR image update. We initialize this algorithm, dubbed FRIST-MRI, with an image (estimate for y) such as the zero-filled Fourier reconstruction $F_u^H z$. Step (i) solves Problem (P7) for $\{x_i, C_k\}$ with fixed W and y as

$$\min_{\{x_i, C_k\}} \sum_{k=1}^{K} \sum_{i \in C_k} \|W\Phi_k R_i y - x_i\|_2^2 \quad s.t. \;\; \|X\|_0 \leq s, \;\; \{C_k\} \in \Gamma\,. \tag{3.10}$$

The exact solution to Problem (3.10) requires calculating the sparsification error (objective) for each possible clustering. The cost of this scales as $O(Pn^2K^P)$ for P patches,[1] which is computationally infeasible. Instead, we present an approximate solution here, which we observed to work well in our experiments. In this method, we first compute the total sparsification error SE_k, associated with each Φ_k, by solving the following problem:

$$SE_k = \sum_{i=1}^{P} SE_k^i \triangleq \min_{\left\{\beta_i^k\right\}} \sum_{i=1}^{P} \left\|W\Phi_k R_i y - \beta_i^k\right\|_2^2 \quad s.t. \quad \left\|B^k\right\|_0 \leq s\,, \tag{3.11}$$

where the columns of B^k are $\left\{\beta_i^k\right\}$. The optimal B^k above is obtained by thresholding the matrix with columns $\left\{W\Phi_k R_i y\right\}_{i=1}^{P}$ and retaining the s largest magnitude elements. The clusters $\{C_k\}$ in (3.10) are approximately computed by assigning $i \in C_{\hat{k}}$ where $\hat{k} = \arg\min_k SE_k^i$. Once the clusters are computed, the corresponding sparse codes $\hat{X}$ in (3.10) (for fixed clusters) are easily found by thresholding the matrix $\left[W\Phi_{\hat{k}_1} R_1 y \mid \ldots \mid W\Phi_{\hat{k}_P} R_P y\right]$ and retaining the s largest magnitude elements [44].

Step (ii) updates the parent transform W with the unitary constraint (and with other variables fixed). The optimal solution, which is similar to previous work [23], is computed as follows. First, we calculate the full SVD $AX^H = \tilde{S}\tilde{\Sigma}\tilde{V}^H$, where the columns of A are $\{\Phi_k R_i y\}_{i=1}^{P}$. The optimal unitary parent transform is then $\hat{W} = \tilde{V}\tilde{S}^H$.

[1]The number of patches is P when we use a patch overlap stride of 1 and include patches at image boundaries by allowing them to 'wrap-around' on the opposite side of the image [35].

Step (iii) solves for y with fixed W and $\{x_i, C_k\}$ as

$$\min_y \sum_{k=1}^{K} \sum_{i \in C_k} \|W\Phi_k R_i y - x_i\|_2^2 + \mu \|F_u y - z\|_2^2 \quad s.t. \;\; \|y\|_2 \leq L. \tag{3.12}$$

As Problem (3.12) is a least squares problem with an ℓ_2 constraint, it can be solved exactly using the Lagrange multiplier method [74]. Thus (3.12) is equivalent to

$$\min_y \sum_{k=1}^{K} \sum_{i \in C_k} \|W\Phi_k R_i y - x_i\|_2^2 + \mu \|F_u y - z\|_2^2 + \rho(\|y\|_2^2 - L)\,, \tag{3.13}$$

where $\rho \geq 0$ is the optimally chosen Lagrange multiplier. An alternative approach to solving Problem (3.12) is by employing the iterative projected gradient method. However, because of the specific structure of the matrices (e.g., partial Fourier sensing matrix in MRI) involved, (3.12) can be solved much more easily and efficiently with the Lagrange multiplier method as discussed next.

Similar to the previous TL-MRI work [44], the normal equation for Problem (3.13) (for known multiplier ρ) can be simplified as follows, where F denotes the full Fourier encoding matrix assumed normalized ($F^H F = I$):

$$(FEF^H + \mu F F_u^H F_u F^H + \rho I)Fy = F \sum_{k=1}^{K} \sum_{i \in C_k} R_i^H \Phi_k^H W^H x_i + \mu F F_u^H z \tag{3.14}$$

where $E \triangleq \sum_{k=1}^{K} \sum_{i \in C_k} R_i^H \Phi_k^H W^H W \Phi_k R_i = \sum_{i=1}^{P} R_i^H R_i$. When the patch overlap stride is 1 and all wrap-around patches are included, $E = nI$, with I the $P \times P$ identity. Since FEF^H, $\mu F F_u^H F_u F^H$ [44], and ρI are all diagonal matrices, the matrix pre-multiplying Fy in (3.14) is diagonal and invertible. Hence, (3.14) can be solved cheaply. Importantly, using a unitary constraint for W leads to an efficient update in (3.14). In particular, the matrix E is not easily diagonalizable when W is not unitary. The problem of finding the optimal Lagrange multiplier reduces to solving a simple scalar equation (see, for example, equation (3.17) in [44]) that can be solved using Newton's method. Thus, the approach based on the Lagrange multiplier method is much simpler compared to an iterative projected gradient scheme to estimate

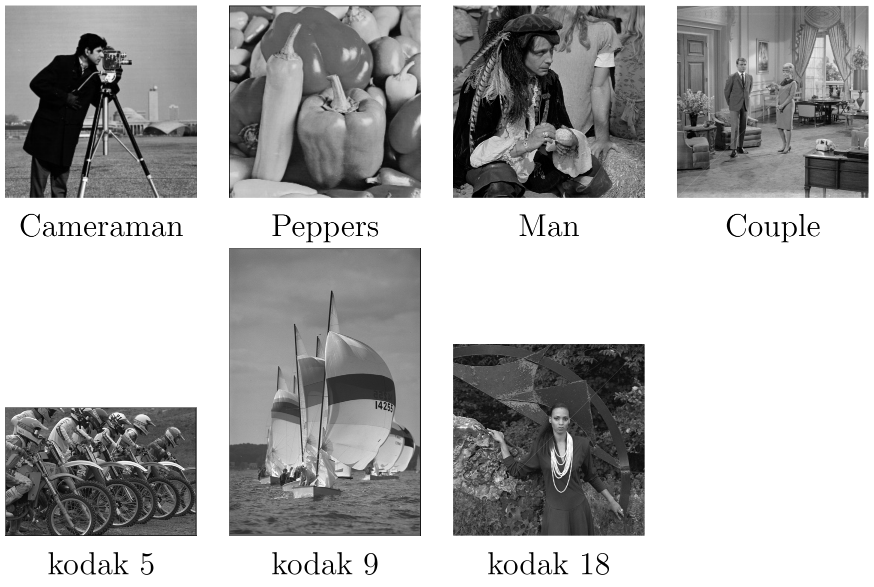


Figure 3.1: Testing images used in the image denoising and image inpainting experiments.

the (typically large) vector-valued image in Problem (3.12).

3.4 Experiment Results

We present numerical convergence results for the FRIST learning algorithm along with image segmentation examples, as well as some preliminary results demonstrating the promise of FRIST learning in applications including image sparse representation, denoising, robust inpainting, and MRI reconstruction. We work with 8×8 non-overlapping patches for the study of convergence and sparse representation, 8×8 overlapping patches for image segmentation, denoising, and robust inpainting, and 6×6 overlapping patches (including patches at image boundaries that 'wrap around' on the opposite side of the image) for the MRI experiments. Figure 3.1 lists the testing images that are used in the image denoising and inpainting experiments.

3.4.1 Empirical Convergence Results

We first illustrate the convergence behavior of FRIST learning. We randomly extract 10^4 non-overlapping patches from the 44 images in the USC-

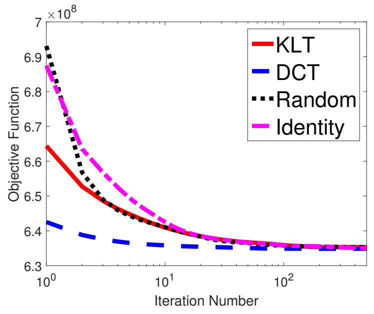
SIPI database [75] (the color images are converted to gray-scale images), and learn a FRIST model, with a 64×64 parent transform W, from the randomly selected patches using fixed sparsity level $s = 10$. We set $K = 2$, and $\lambda_0 = 3.1 \times 10^{-3}$ for visualization simplicity. In the experiment, we initialize the learning algorithm with different square 64×64 parent transform W's, including the (*i*) Karhunen-Loève Transform (KLT), (*ii*) 2D DCT, (*iii*) random matrix with i.i.d. Gaussian entries (zero mean and standard deviation 0.2), and (*iv*) identity matrix.

Figures 3.2(a) and 3.2(d) illustrate that the objective and sparsification error in (P2) converge to similar values from various initializations of W, indicating that the algorithm is reasonably insensitive or robust to initializations in practice. Figures 3.2(b) and 3.2(c) show the cluster size changes over iterations for the 2D DCT and KLT initializations. The final values of the cluster sizes are also similar (although, not necessarily identical) for various initializations. Figure 3.2(e) and 3.2(f) display the learned FRIST parent W's with DCT and random matrix initializations. They are non-identical, and capture features that sparsify the image patches equally well. Thus we consider such learned transforms to be essentially equivalent as they achieve similar objective values and sparsification errors for the training data and are similarly conditioned (the learned parent W's with the DCT and random matrix initializations have condition numbers 1.04 and 1.06, respectively).

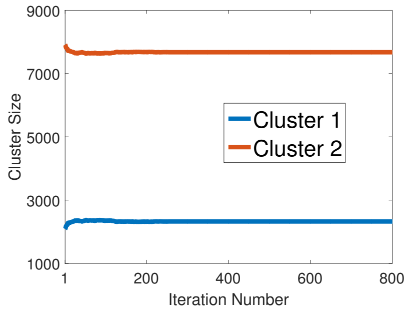
The numerical results demonstrate that our FRIST learning algorithm is reasonably robust, or insensitive to initialization. Good initialization for the parent transform W, such as the DCT, leads to faster convergence during learning. Thus, we initialize the parent transform W using the 2D DCT in the rest of the experiments.

3.4.2 Image Segmentation and Clustering Behavior

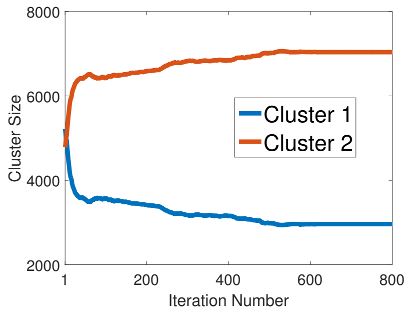
The FRIST learning algorithm is capable of clustering image patches according to their orientations. In this subsection, we illustrate the FRIST clustering behavior by image segmentation experiments. We consider the images *Wave* (512×512) and *Field* (512×512) shown in Fig. 3.3(a) and Fig. 3.4(a) as inputs. Both images contain directional textures, and we aim to cluster the pixels of the images into one of four classes, which represent



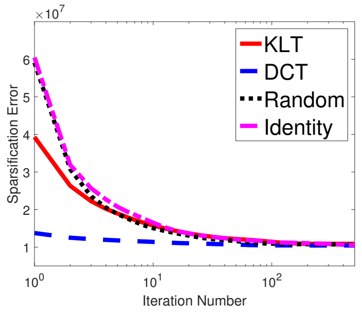
(a) FRIST Objective



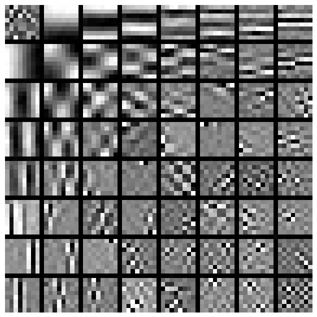
(b) DCT initialization



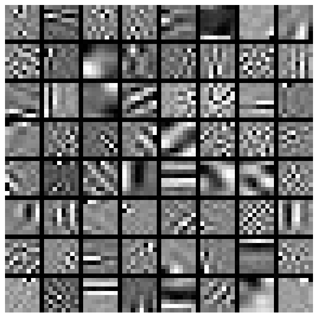
(c) KLT initialization



(d) FRIST
Sparsification Error

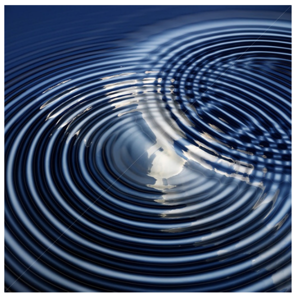


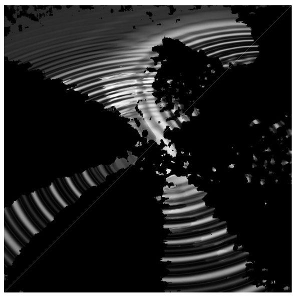
(e) FRIST parent W with DCT initialization

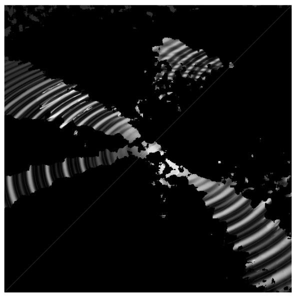


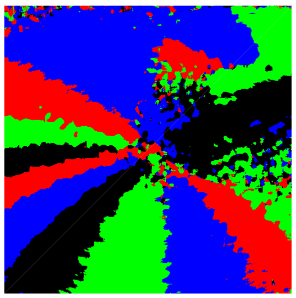
(f) FRIST parent W with random initialization

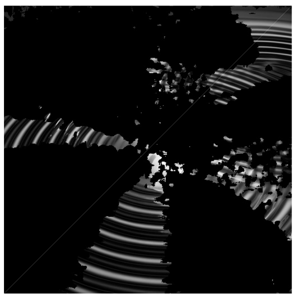
Figure 3.2: Convergence of the FRIST objective, sparsification error, and cluster size with various parent transform initializations, as well as the visualizations of the learned FRIST parent transforms with DCT and random initializations.


(a) *Wave*


(c) Class 1


(e) Class 3


(b) Pixel memberships


(d) Class 2

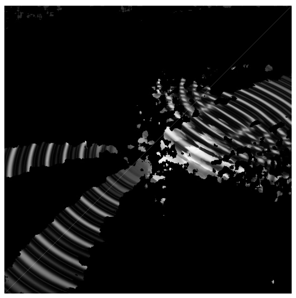
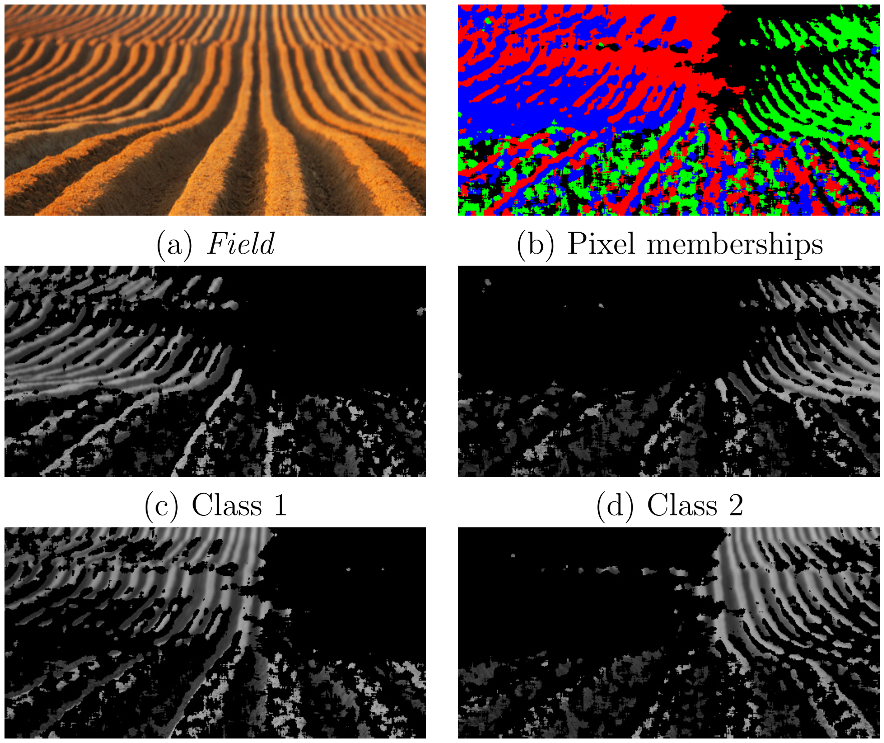

(f) Class 4

Figure 3.3: Image segmentation result of *Wave* (512×512) using FRIST learning on the gray-scale version of the image. The colors red, green, blue, and black in (b) represent pixels that belong to the four classes. Pixels that are clustered into a specific class are shown in gray-scale (using intensities in the original gray-scale image), while pixels that are not clustered into that class are shown in black for (c)-(f).



(a) *Field* (b) Pixel memberships

(c) Class 1 (d) Class 2

(e) Class 3 (f) Class 4

Figure 3.4: Image segmentation result of *Field* (256 $\times$ 512) using FRIST learning on the gray-scale version of the image. The colors red, green, blue, and Bblack in (b) represent pixels that belong to the four classes. Pixels that are clustered into a specific class are shown in gray-scale (using intensities in the original gray-scale image), while pixels that are not clustered into that class are shown in black for (c)-(f).

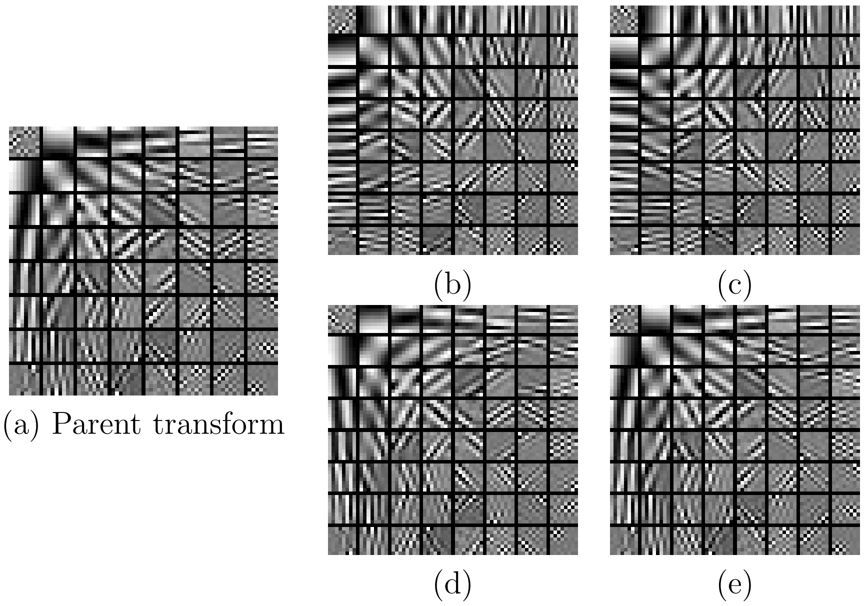


Figure 3.5: Visualization of the learned (a) parent transform, and (b)-(e) children transforms in FRIST for the image *Wave*. The rows of each child transform are displayed as 8×8 patches.

different orientations or flips. For each input image, we convert it into gray-scale, extract the overlapping mean-subtracted patches, and learn a FRIST while clustering the patches using the algorithm in Chapter 2. As overlapping patches are used, each pixel in the image belongs to several overlapping patches. We cluster a pixel into a particular class by majority voting among the patches that contain it.

We set $s = 10$, and $K = 4$ in the clustering experiments. Figures 3.3 and 3.4 illustrate the segmentation results of images *Wave* and *Field*, respectively. Figures 3.3(b) and 3.4(b) illustrate the pixel memberships with four different colors (blue, red, green, and black, for classes 1 to 4, respectively). Figures 3.3(c)-(f) and 3.4(c)-(f) each visualize the image pixels clustered into a specific class in gray-scale, and the pixels that are not clustered into that class are shown in black. Each class captures edges at specific orientations.

The parent transform W and its children transforms W_k's in the learned FRIST for the *Wave* image are visualized in Fig. 3.5 with the rows of each W_k displayed as 8×8 patches. We observe that each child transform contains distinct directional features that were adaptively learned to sparsify edges with specific orientations better. The parent W turns out to be identical to

Table 3.1: PSNR values for reconstruction of images from sparse representations obtained using the 2D DCT, learned SST and OCTOBOS, square and overcomplete K-SVD, and learned FRIST. The first row of the table provides average PSNR values computed over the 44 images from the USC-SIPI database. The best PSNR values are marked in bold.

Methods	2D DCT	SST	OCTOBOS	K-SVD		**FRIST**
Model Size		64×64	128×64	64×64	64×128	64×64
USC-SIPI	34.36	34.20	33.62	34.11	35.08	**35.14**
Cameraman	29.49	29.43	29.03	29.09	30.16	**30.63**
House	36.89	36.36	35.38	36.31	37.41	**37.71**

the child transform shown in Fig. 3.4(e), implying that the corresponding FR operator is the identity matrix.

The preliminary image segmentation results here demonstrate some potential for the FRIST scheme for directional classification or segmentation. More importantly, we wish to illustrate why FRIST can provide improvements over SST or OCTOBOS in various inverse problems. As natural images usually contain a variety of directional features and edges, FRIST is capable of grouping those patches with similar orientations/flips, and thus provides better sparsification in each cluster using directional children transforms, even while learning only a single small parent transform W (which could be learned even in cases of very limited or corrupted data).

3.4.3 Sparse Image Representation

Most of the popular image compression methods make use of analytical sparsifying transforms. In particular, the commonly used JPEG uses the 2D DCT to sparsify image patches. Data-driven adaptation of dictionaries using the K-SVD scheme has also been shown to be beneficial for image compression, compared to fixed analytical transforms [76]. In this section, we show that the proposed FRIST learning scheme provides improved sparse representations of images compared to related adaptive sparse modeling methods. While we focus here on a simple study of the sparse representation abilities of adaptive FRIST, the investigation of a complete adaptive image compression framework based on FRIST and its comparison to benchmarks is left for future work.

We learn a FRIST, with a 64×64 parent transform W, from the 10^4 randomly selected patches (from USC-SIPI images) used in Section 3.4.1. We set $K = 32$, $s = 10$ and $\lambda_0 = 3.1 \times 10^{-3}$. We compare the learned FRIST with other popular adaptive sparse signals models. In particular, we train a 64×64 SST [23], a 128×64 OCTOBOS [5], as well as a 64×64 square (synthesis) dictionary and a 64×128 overcomplete dictionary using KSVD [4], using the same training patches and sparsity level as for FRIST.

With the learned models, we represent each image from the USC-SIPI database as well as some other standard images. Each image is represented compactly by storing its sparse representation including (*i*) non-zero coefficient values in the sparse codes of the 8×8 non-overlapping patches, (*ii*) locations of the non-zeros (plus the cluster membership if necessary) in the sparse codes for each patch and (*iii*) the adaptive sparse signal model (e.g., the dictionary or transform matrix – this would typically involve negligible overhead). For each method, the patch sparsity (or equivalently, the number of non-zero coefficients per patch) is set to $s = 10$ (same as during training). The adaptive SST, square KSVD, and adaptive FRIST methods store only a 64×64 square matrix in (*iii*) above, whereas the overcomplete KSVD and OCTOBOS methods store a 128×64 matrix.

The images (i.e., their non-overlapping patches) are reconstructed from their sparse representations in a least squares sense, and the reconstruction quality for each image is evaluated using the peak-signal-to-noise ratio (PSNR), expressed in decibels (dB). We use the average of the PSNR values over all 44 USC-SIPI images as the indicator of the quality of sparse representation of the USC-SIPI database.

Table 3.1 lists the sparse representation reconstruction results for the USC-SIPI database and the images *Cameraman* (256×256) and *House* (256×256). We observe that the learned FRIST model provides the best reconstruction quality compared to other adaptive sparse signal models or the analytical 2D DCT, for both the USC-SIPI images and the external images. Compared to unstructured overcomplete models such as KSVD and OCTOBOS, the proposed FRIST provides improved PSNRs, while achieving potentially fewer bits for sparse representation.[2] Additionally, dictionary learning based repre-

[2]Assuming for simplicity that L bits are used to describe each non-zero coefficient value in each model, the total number of bits for storing the sparse code (non-zero locations,

sentation requires synthesis sparse coding, which is more expensive than the cheap and exact sparse coding in the transform model-based methods [20]. As mentioned before, the investigation of an image compression system based on learned FRIST models, and its analysis as well as quantitative comparison to other compression benchmarks, are left for future work.

3.4.4 Image Denoising

We present denoising results using our FRIST-based framework in Section 3.1. We simulate i.i.d. Gaussian noise at four different noise levels ($\sigma = 5$, 10, 15, 20) for seven standard images in Fig. 3.1. Denoising results obtained by our proposed algorithm in Section 3.1 are compared with those obtained by the adaptive overcomplete K-SVD denoising scheme [4], adaptive SST denoising scheme [23] and the adaptive OCTOBOS denoising scheme [5]. We also compare to the denoising result using the SST method, but with fixed 2D DCT (i.e., no learning).

We set $K = 64$, $n = 64$, $C = 1.04$ for the FRIST denoising method. For the adaptive SST and OCTOBOS denoising methods, we follow the same parameter settings as used in the previous works [5, 23]. The same parameter settings as for the SST method are used for the DCT-based denoising algorithm. A 64×256 learned synthesis dictionary is used in the synthesis K-SVD denoising method, and for the OCTOBOS denoising scheme we use a corresponding 256×64 learned OCTOBOS. For the K-SVD, adaptive SST, and adaptive OCTOBOS denoising methods, we used the publicly available implementations [77, 78] in this experiment.

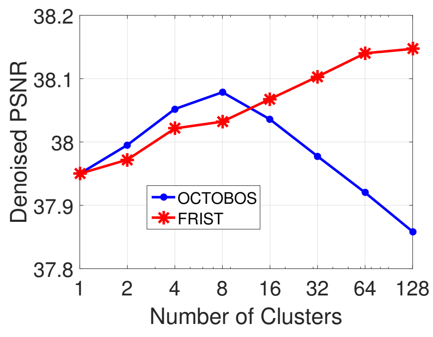
Table 3.2 lists the denoised image PSNR values for the various methods for the seven tested images at several noise levels. The proposed FRIST scheme provides consistently better PSNRs compared to the other fixed or adaptive sparse modeling methods including DCT, SST, K-SVD, and OCTOBOS. The average denoising PSNR improvements provided by adaptive FRIST over DCT, adaptive SST, K-SVD, and adaptive OCTOBOS are 0.48 dB, 0.26 dB, 0.47 dB, and 0.04 dB respectively, and the standard deviations in these improvements are 0.26 dB, 0.11 dB, 0.22 dB, and 0.03 dB, respectively.

non-zero coefficient values, cluster membership) of a patch is $6s + Ls + 5$ in 64×64 FRIST ($K = 32$), and $7s + Ls$ in 64×128 KSVD. For the setting $s = 10$, FRIST requires 5 fewer bits per patch compared to KSVD.

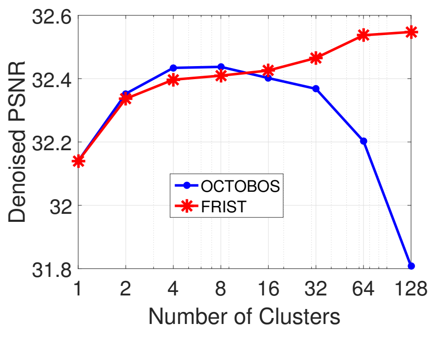
Table 3.2: PSNR values (in dB) for denoising with 64×64 adaptive FRIST along with the corresponding PSNR values for denoising using the 64×64 2D DCT, the 64×64 adaptive SST, the 64×256 overcomplete K-SVD, and the 256×64 learned OCTOBOS. The best PSNR values are marked in bold.

Image	σ	Noisy PSNR	DCT	SST	K-SVD	OCTOBOS	FRIST
Peppers (256×256)	5	34.14	37.70	37.95	37.78	38.09	**38.16**
	10	28.10	34.00	34.37	34.24	34.57	**34.68**
	15	24.58	31.83	32.14	32.18	32.43	**32.54**
	20	22.12	30.06	30.62	30.80	30.97	**31.02**
Cameraman (256×256)	5	34.12	37.77	38.01	37.82	**38.16**	**38.16**
	10	28.14	33.63	33.90	33.72	34.13	**34.16**
	15	24.61	31.33	31.65	31.51	31.95	**31.97**
	20	22.10	29.81	29.91	29.82	30.24	**30.33**
Man (768×768)	5	34.15	36.59	36.64	36.47	36.73	**36.82**
	10	28.13	32.86	32.95	32.71	32.98	**33.06**
	15	24.63	30.88	30.96	30.78	31.07	**31.10**
	20	22.11	29.42	29.58	29.40	29.74	**29.76**
Couple (512×512)	5	34.16	37.25	37.32	37.29	37.40	**37.43**
	10	28.11	33.48	33.60	33.50	33.73	**33.78**
	15	24.59	31.35	31.47	31.44	**31.71**	**31.71**
	20	22.11	29.82	30.01	30.02	30.34	**30.36**
Kodak 5 (768×512)	5	34.17	36.72	36.96	36.32	37.10	**37.17**
	10	28.12	32.03	32.33	31.86	32.57	**32.62**
	15	24.60	29.51	29.84	29.49	30.13	**30.16**
	20	22.13	27.79	28.09	27.89	28.40	**28.47**
Kodak 9 (512×768)	5	34.14	39.35	39.45	38.85	**39.53**	**39.53**
	10	28.15	35.66	35.98	35.39	36.23	**36.26**
	15	24.60	33.36	33.89	33.39	34.27	**34.28**
	20	22.11	31.66	32.30	31.90	32.73	**32.76**
Kodak 18 (512×768)	5	34.17	36.75	36.72	36.50	**36.83**	**36.83**
	10	28.12	32.40	32.44	32.20	**32.59**	**32.59**
	15	24.62	30.02	30.06	29.88	30.27	**30.31**
	20	22.12	28.42	28.49	28.35	28.72	**28.77**

Performance vs. Number of Clusters. In applications such as image denoising, when OCTOBOS or FRIST are learned from limited noisy



(a) $\sigma = 5$



(b) $\sigma = 15$

Figure 3.6: Denoising PSNR for *Peppers* as a function of the number of clusters (including flipping and rotations) K.

patches, OCTOBOS with many more degrees of freedom is more likely to overfit the data and learn noisy features, which can degrade the denoising performance. Figure 3.6 provides an empirical illustration of this behavior, and plots the denoising PSNRs for *Peppers* as a function of the number of child transforms or number of clusters K for $\sigma = 5$ and $\sigma = 15$. In both cases, the denoising PSNRs of the OCTOBOS and FRIST schemes increase with K initially. However, beyond an optimal value of K, the OCTOBOS denoising scheme suffers from overfitting the noise. Thus the OCTOBOS performance in Fig. 3.6 quickly degrades as the number of transforms (in the collection/union) or clusters to be learned from a set of noisy image patches is increased [5]. In contrast, the structured FRIST-based denoising scheme (involving much fewer degrees of freedom) is more robust or resilient to noise. As K increases, adaptive FRIST denoising provides continually monotonically increasing denoising PSNR in Fig. 3.6. For example, while the FRIST PSNR achieves a peak value for $K = 128$, the PSNR for adaptive OCTOBOS denoising is significantly lower at such a large K.

Although we focused our comparisons here on related adaptive sparse modeling methods, a very recent work [39] shows that combining transform learning based denoising with non-local similarity models leads to better denoising performance, and outperforms the state-of-the-art BM3D denoising method [32]. A further extension of the work in [39] to include FRIST learning is of interest and could potentially provide even greater advantages, but we leave this detailed investigation to future work.

Table 3.3: PSNR values for image inpainting, averaged over six images, using the proposed adaptive FRIST based method, along with the corresponding values obtained using cubic interpolation (Cubic), patch smooth ordering (Smooth), patch-based DCT, adaptive SST, and adaptive OCTOBOS based methods, for various fractions of available pixels and noise levels. The best PSNR value in each row is marked in bold.

Avail. pixels	σ	Corrupt. PSNR	Cubic	Smooth	DCT	SST	OCTOBOS	FRIST
20%	0	6.41	25.86	27.99	28.32	28.49	28.60	**28.65**
	5	6.39	6.40	27.86	28.26	28.44	28.53	**28.61**
	10	6.36	6.37	26.46	27.46	27.98	28.25	**28.41**
	15	6.33	6.33	25.02	26.60	27.38	27.71	**27.92**
10%	0	5.89	23.19	24.87	25.21	25.25	25.25	**25.31**
	5	5.88	5.89	24.81	24.98	25.19	25.30	**25.38**
	10	5.86	5.87	24.10	24.40	24.44	24.69	**24.80**
	15	5.81	5.82	23.28	23.62	23.87	24.11	**24.22**

3.4.5 Image Inpainting

We present preliminary results for our adaptive FRIST-based inpainting framework (based on (P6)). We randomly remove 80% and 90% of the pixels of the entire images in Fig. 3.1, and simulate i.i.d. additive Gaussian noise for the sampled pixels with $\sigma = 0$, 5, 10, and 15. We set $K = 64$, $n = 64$, and apply the proposed adaptive FRIST inpainting algorithm to reconstruct the images from the corrupted and noisy measurements. For comparison, we replace the adaptive FRIST in the proposed inpainting algorithm with the fixed 2D DCT, adaptive SST [23], and adaptive OCTOBOS [5] respectively, and evaluate the inpainting performance of these alternatives. The image inpainting results obtained by the FRIST based methods are also compared with those obtained by the cubic interpolation [79, 80] and patch smoothing [81] methods. We used the Matlab function "griddata" to implement the cubic interpolation, and we used the publicly available implementation of the patch smoothing method. For the DCT, SST, OCTOBOS, and FRIST based methods, we initialize the image patches using the cubic interpolation method in noiseless cases, and using the patch smoothing method in noisy cases.

Table 3.4: Comparison of the PSNRs corresponding to the zero-filling, Sparse MRI, DL-MRI, PBDWS, PANO, TL-MRI, and the proposed FRIST-MRI reconstructions for various images, sampling schemes, and undersampling factors. The best PSNR for each MRI image is marked in bold.

Im.	Sampl. Scheme	Under-sampl.	Zero-filling	Sparse MRI	DL-MRI	PBD -WS	PANO	TL-MRI	FRIST-MRI
1	Cartesian	7×	27.9	28.6	30.9	31.1	31.1	31.2	**31.4**
2	Random	5×	26.9	27.9	30.5	30.3	30.4	30.6	**30.7**
3	Cartesian	2.5×	24.9	29.9	36.6	35.8	34.8	36.3	**36.7**

Table 3.3 lists the image inpainting PSNR results, averaged over the images shown in Fig. 3.1, for various fractions of sampled pixels and noise levels. The proposed adaptive FRIST inpainting scheme provides better PSNRs compared to the other inpainting methods based on interpolation, transform-domain sparsity, and spatial similarity. The average inpainting PSNR improvements achieved by FRIST over DCT, SST, and OCTOBOS are 0.56 dB, 0.28 dB, and 0.11 dB respectively, and the standard deviations in these improvements are 0.39 dB, 0.16 dB, and 0.05 dB respectively. Importantly, adaptive FRIST provides bigger improvements than the other methods including the learned OCTOBOS, at higher noise levels. Figure 3.7 provides an illustration of the inpainting results, with regional zoom-in for visual comparisons. We observe that the cubic interpolation produces blur in various locations. The FRIST result is much improved, and also shows fewer artifacts compared to the patch smoothing [81] and adaptive SST results. Table 3.3 shows that the cubic Interpolation method is extremely sensitive to noise, whereas the FRIST based method is the most robust. These results indicate the benefits of adapting the highly constrained yet overcomplete FRIST data model.

3.4.6 MRI Reconstruction

We present preliminary MRI reconstruction results using the proposed FRIST-MRI algorithm. The three complex-valued images and the corresponding k-space sampling masks used in this section are shown in Fig. 3.8, Fig.

3.9(a), and Fig. 3.9(b).[3] We retrospectively undersample the k-space of the reference images using the displayed sampling masks. We set $K = 32$, the sparsity level $s = 0.05 \times nP$, and the other parameters were set similarly as for TL-MRI in [44]. We used a higher sparsity level $s = 0.085 \times nN$ for reconstructing Image ***3***, which worked well. To speed up convergence, lower sparsity levels are used in the initial iterations [44]. We compare our FRIST-MRI reconstruction results to those obtained using conventional or popular methods, including naive Zero-filling, Sparse MRI [72], DL-MRI [73], PBDWS [82], PANO [83], and TL-MRI [44]. The parameter settings for these methods are as mentioned in [44]. We separately tuned the sparsity parameter for TL-MRI [44] for reconstructing Image ***3***.[4] The reconstruction PSNRs (computed for image magnitudes) for various approaches are compared in Table 3.4.

First, the proposed FRIST-MRI algorithm provides significant improvements over the naive zero-filling reconstruction (the initialization of the algorithm) with 6.4 dB better PSNR on average, as well as 4.2 dB better PSNR (on average) over the non-adaptive sparse MRI reconstructions. Compared to recently proposed popular MRI reconstruction methods, the FRIST-MRI algorithm demonstrates reasonably better performance for each testing case, with an average PSNR improvement of 0.8 dB, 0.5 dB, and 0.3 dB over the non-local patch similarity-based PANO method, the partially adaptive PBDWS method, and the adaptive dictionary-based DL-MRI method.

The proposed FRIST-MRI reconstruction quality is 0.2 dB better than TL-MRI on average. As we followed a reconstruction framework and parameters similar to those used by TL-MRI [44], the quality improvement obtained with FRIST-MRI is solely because the learned FRIST can serve as a better regularizer for MR image reconstruction compared to the single adaptive square transform in TL-MRI. Figure 3.9 visualizes the reconstructions and reconstruction errors (magnitude of the difference between the magnitudes of the reconstructed and reference images) for FRIST-MRI and TL-MRI. The FRIST-MRI reconstruction error map clearly shows fewer artifacts, especially along the boundaries of the circles, compared to TL-MRI.

[3]The testing image data in this section were used and included in previous works [26,44] with the data sources.

[4]We observed improved reconstruction PSNR compared to the result obtained using the sparsity settings in [44].

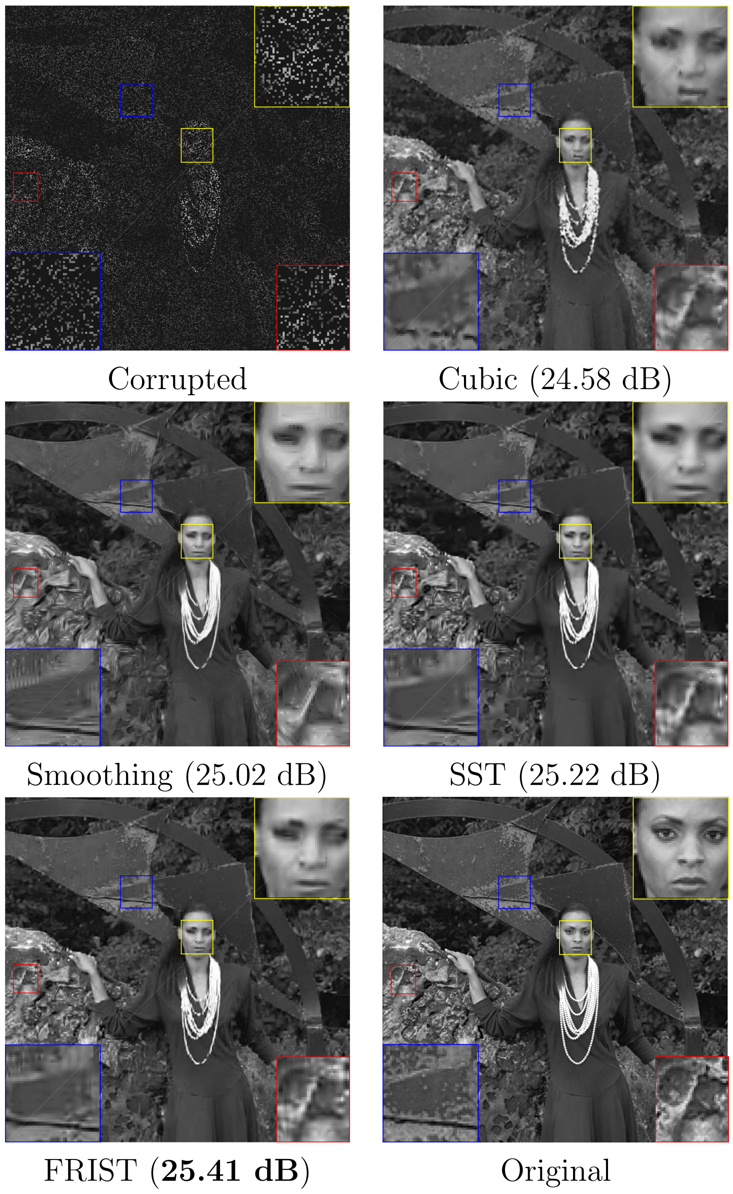


Figure 3.7: Illutration of image inpainting results for the *Kodak 18* (80% pixels are missing) with regional zoom-in comparisons.

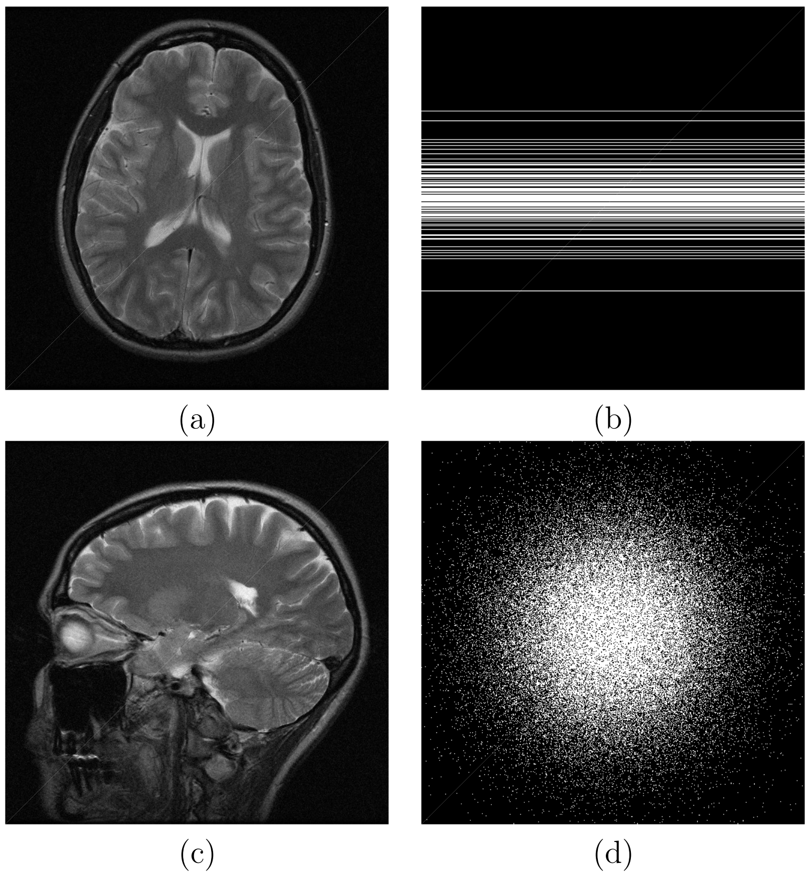


Figure 3.8: Testing MRI images and their k-space sampling masks: (a) Image ***1***; (b) k-space sampling mask (Cartesian with 7$\times$ undersampling) for Image ***1***; (c) Image ***2***; (d) k-space sampling mask (2D random with 5$\times$ undersampling) for Image ***2***.

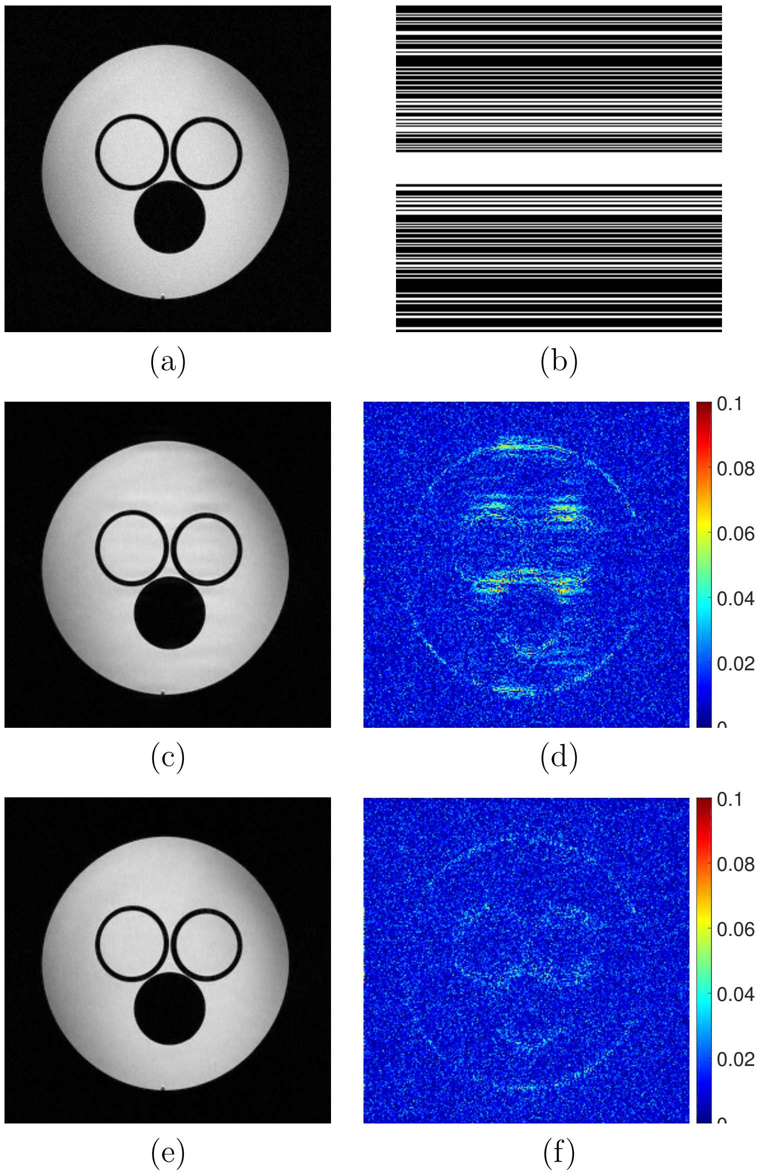


Figure 3.9: Visualization of the reconstruction of Image ***3*** using Cartesian sampling and 2.5× undersampling: (a) Image ***3***; (b) sampling mask in k-space; (c) TL-MRI reconstruction (36.3 dB); (d) magnitude of TL-MRI reconstruction error; (e) FRIST-MRI reconstruction (36.7 dB); and (f) magnitude of FRIST-MRI reconstruction error.

CHAPTER 4

HIGH-DIMENSIONAL SPARSIFYING TRANSFORM LEARNING FOR ONLINE VIDEO DENOISING

4.1 Introduction to Video Denoising

This chapter presents novel online data-driven video denoising techniques based on learning sparsifying transforms for appropriately constructed spatio-temporal patches of videos. This new framework provides high quality video restoration from highly corrupted data. In the following, we briefly review the background on video denoising and sparsifying transform learning, before discussing the contributions of this work.

4.1.1 Video Denoising

Denoising is one of the most important problems in video processing. The ubiquitous use of relatively low-quality smart phone cameras has also led to the increasing importance of video denoising. Recovering high-quality video from noisy footage also improves robustness in high-level vision tasks [37,84].

Though image denoising algorithms, such as the popular BM3D [32], can be applied to process each video frame independently, most of the video denoising techniques exploit the spatio-temporal correlation of the tensor 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) [85] and wavelets [86]. Prior works exploited this fact and proposed video (or high-dimensional data) denoising algorithms by coefficient shrinkage, e.g., sparse approximation [31] or Wiener filtering [87]. Since there are typically motions involved in videos, objects can move throughout the scene. Thus, state-of-the-art video and image denoising algorithms also combine block matching (BM) to group local patches, and apply denoising jointly [32, 59, 87].

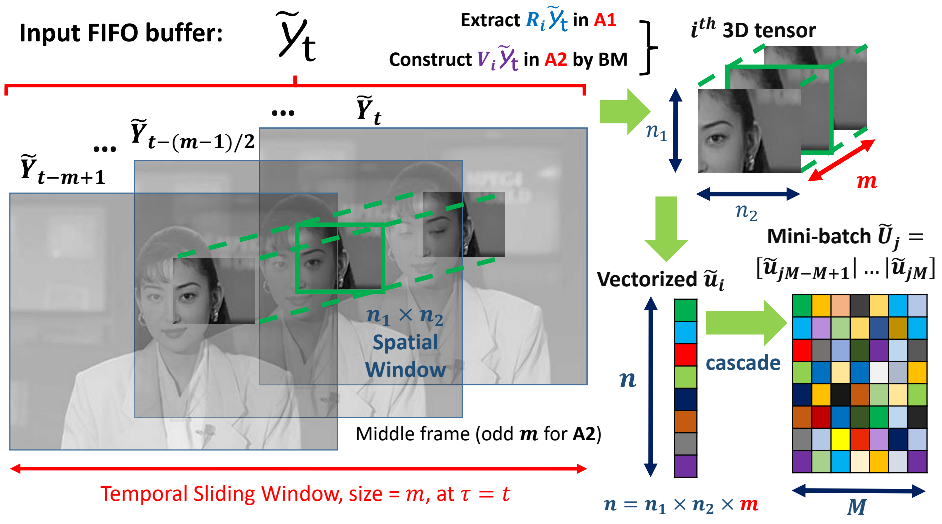

Input FIFO buffer: $\tilde{y}_t$
Extract $R_i \tilde{y}_t$ in A1
Construct $V_i \tilde{y}_t$ in A2 by BM
i^{th} 3D tensor
$\tilde{Y}_{t-m+1}$
$\tilde{Y}_{t-(m-1)/2}$
$\tilde{Y}_t$
$n_1 \times n_2$ Spatial Window
Middle frame (odd m for A2)
Temporal Sliding Window, size = m, at $\tau = t$
n_1
n_2
m
Vectorized $\tilde{u}_i$
Mini-batch $\tilde{U}_j = [\tilde{u}_{jM-M+1}| \dots |\tilde{u}_{jM}]$
cascade
n
$n = n_1 \times n_2 \times m$
M

Figure 4.1: Illustration of video streaming, tensor construction and vectorization.

4.1.2 Sparsifying Transform Learning

Most of the aforementioned video denoising methods exploit sparsity in fixed transform domain (e.g., DCT) as part of their framework. It has been shown that the adaptation of sparse models based on training signals usually leads to superior performance over fixed sparse representation in many applications. Synthesis dictionary learning is the most well-known adaptive sparse representation scheme [4, 8]. However, the synthesis model sparse coding problem is NP-hard. The commonly used approximate sparse coding algorithms still involve relatively expensive computations. As an alternative, the transform model suggests that the signal u is approximately sparsifiable using a transform $W \in \mathbb{R}^{m \times n}$, i.e., $Wu = x + e$, with $x \in \mathbb{R}^m$ sparse and e a small approximation error in the transform domain (rather than in the signal domain). Recent works proposed sparsifying transform learning [20, 23] with cheap and exact sparse coding steps, which turn out to be advantageous in various applications such as natural data representations, image denoising, inpainting, segmentation, magnetic resonance imaging (MRI), and computed tomography (CT) [5, 21, 22, 27, 44, 88].

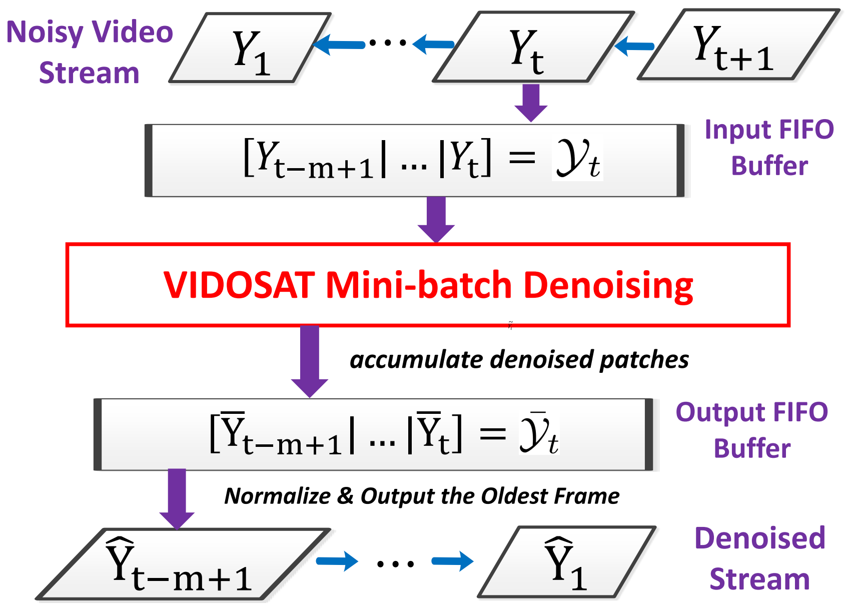

Noisy Video Stream
Y_1
Y_t
Y_{t+1}
$[Y_{t-m+1}| \dots |Y_t] = \mathcal{Y}_t$
Input FIFO Buffer
VIDOSAT Mini-batch Denoising
accumulate denoised patches
$[\bar{Y}_{t-m+1}| \dots |\bar{Y}_t] = \bar{\mathcal{Y}}_t$
Output FIFO Buffer
Normalize & Output the Oldest Frame
$\hat{Y}_{t-m+1}$
$\hat{Y}_1$
Denoised Stream

Figure 4.2: Illustration of online video streaming and denoising framework.

4.1.3 Contribution

While the data-driven adaptation of synthesis dictionaries for the purpose of denoising image sequences or volumetric data [30, 31] has been studied in some recent papers, the usefulness of learned sparsifying transforms has not been explored in these applications. Video data typically contains correlation along the temporal dimension, which will not be captured by learning sparsifying transforms for the 2D patches of the video frames. Thus in this work, we focus on video denoising using high-dimensional OSTL. We propose the method of VIdeo Denoising by Online SpArsifying Transform learning (VIDOSAT). The sparsifying transform is adapted to the tensors formed by the local patches of the corrupted video on-the-fly. Figure 4.1 illustrates how the spatio-temporal tensors are constructed and vectorized from the streaming video, and Fig. 4.2 is a flow-chart of the proposed VIDOSAT framework.

To our knowledge, this is the first video denoising method using online sparse signal modeling, by applying high-dimensional sparsifying transform learning for spatio-temporal data. Table 4.1 summarizes the key attributes of some of the aforementioned related video denoising algorithm representatives,

Table 4.1: Comparison between video denoising methods, including fBM3D, 3D DCT, sKSVD, VBM3D, VBM4D, as well as VIDOSAT and VIDOSAT-BM proposed here. fBM3D is applying BM3D algorithm for denoising each frame, and the 3D DCT method is applying the VIDOSAT framework but using the fixed 3D DCT transform.

Methods	Sparse Signal Model			BM	Temporal Correlation
	Fixed	Adaptive	Online		
fBM3D	✓			✓	
3D DCT	✓				✓
sKSVD		✓			✓
VBM3D	✓			✓	✓
VBM4D	✓			✓	✓
VIDOSAT		✓	✓		✓
VIDOSAT -BM		✓	✓	✓	✓

as well as the proposed VIDOSAT algorithms. Our contributions can be summarized as follows:

- We propose a video denoising framework, which processes noisy frames in an online fashion. Within the framework, we present two methods of spatio-temporal tensor construction, one of which utilizes block matching (BM) for motion compensation.

- We apply the OSTL for reconstruction of sequentially arrived tensors, whose spatio-temporal structure is exploited using the adaptive 3D transform-domain sparsity. The denoised tensors are aggregated to reconstruct the streaming video frames.

- We evaluate the video denoising performance of the proposed algorithms, which outperform the competing methods over several public video datasets.

4.2 Signal Denoising via Online Transform Learning

The goal in denoising is to recover an estimate of a signal $\tilde{\mathbf{u}} \in \mathbb{R}^n$ from the measurement $\mathbf{u} = \tilde{\mathbf{u}} + \mathbf{e}$, corrupted by additive noise $\mathbf{e}$. Here, we consider a time sequence of noisy measurements $\{\mathbf{u}_t\}$, with $\mathbf{u}_t = \tilde{\mathbf{u}}_t + \mathbf{e}_t$. We assume noise $\mathbf{e}_t \in \mathbb{R}^n$ whose entries are independent and identically distributed (i.i.d.) Gaussian with zero mean and possibly time-varying but known variance σ_t^2. Online denoising is to recover the estimates $\hat{\mathbf{u}}_t$ for $\tilde{\mathbf{u}}_t \; \forall \, t$ sequentially. Such time-sequential denoising with low memory requirements would be especially useful for streaming data applications. We assume that the underlying signals $\{\tilde{\mathbf{u}}_t\}$ are approximately sparse in an (unknown, or to be estimated) transform domain.

4.2.1 Online Transform Learning

In prior work [28], we proposed an online signal denoising methodology based on sparsifying transform learning, where the transform is adapted based on sequentially processed data. For time $t = 1, 2, 3$, etc., the problem of updating the adaptive sparsifying transform and sparse code (i.e., the sparse representation in the adaptive transform domain) to account for the new noisy signal $\mathbf{u}_t \in \mathbb{R}^n$ is

$$\left\{\hat{\mathbf{W}}_t, \hat{\mathbf{x}}_t\right\} = \underset{\mathbf{W}, \mathbf{x}_t}{\arg\min} \frac{1}{t} \sum_{\tau=1}^{t} \left\{ \|\mathbf{W}\mathbf{u}_\tau - \mathbf{x}_\tau\|_2^2 + \lambda_\tau \nu(\mathbf{W}) \right\}$$
$$+ \frac{1}{t} \sum_{\tau=1}^{t} \alpha_\tau^2 \|\mathbf{x}_\tau\|_0 \quad s.t. \;\; \mathbf{x}_\tau = \hat{\mathbf{x}}_\tau, \; 1 \le \tau \le t-1 \;\; \text{(P1)},$$

where the ℓ_0 "norm" counts the number of nonzeros in x_τ, which is the sparse code of $\mathbf{u}_\tau$. Thus $\|\mathbf{W}\mathbf{u}_\tau - \mathbf{x}_\tau\|_2^2$ is the sparsification error (i.e., the modeling error in the transform model) for $\mathbf{u}_\tau$ in the transform $\mathbf{W}$. The term $\nu(\mathbf{W}) = -\log\, |\det \mathbf{W}| + \|\mathbf{W}\|_F^2$ is a transform learning regularizer [20], $\lambda_\tau = \lambda_0 \|\mathbf{u}_\tau\|_2^2$ with $\lambda_0 > 0$ allows the regularizer term to scale with the first term in the cost, and the weight α_τ is chosen proportional to σ_τ (the standard deviation of noise in $\tilde{\mathbf{u}}_\tau$). Matrix $\hat{\mathbf{W}}_t$ in (P1) is the optimal transform at time t, and $\hat{\mathbf{x}}_t$ is the optimal sparse code for $\mathbf{u}_t$.

Note that at time t, only the latest optimal sparse code $\hat{\mathbf{x}}_t$ is updated in

(P1)[1] along with the transform $\hat{\mathbf{W}}_t$. The condition $\mathbf{x}_\tau = \hat{\mathbf{x}}_\tau,\ 1 \leq \tau \leq t-1$, is therefore assumed. For brevity, we will not explicitly restate this condition (or its variants) in the formulations in the rest of this chapter. Although at each time t the transform is updated based on all the past and present observed data, the online algorithm for (P1) [28] involves efficient operations based on a few matrices of modest size, accumulated sequentially over time.

The regularizer $\nu(\mathbf{W})$ in (P1) prevents trivial solutions and controls the condition number and scaling of the learned transform [20]. The condition number $\kappa(\mathbf{W})$ is upper bounded by a monotonically increasing function of $\nu(\mathbf{W})$ [20]. In the limit $\lambda_0 \to \infty$ (and assuming the $\mathbf{u}_\tau$, $1 \leq \tau \leq t$, are not all zero), the condition number of the optimal transform in (P1) tends to 1. The specific choice of λ_0 (and hence the condition number) depends on the application.

Denoising

Given the optimal transform $\hat{\mathbf{W}}_t$ and the sparse code $\hat{\mathbf{x}}_t$, a simple estimate of the denoised signal is obtained as $\hat{\mathbf{u}}_t = \hat{\mathbf{W}}_t^{-1}\hat{\mathbf{x}}_t$. Online transform learning can also be used for patch-based denoising of large images [28]. Overlapping patches of the noisy images are processed sequentially (e.g., in raster scan order) via (P1), and the denoised image is obtained by averaging together the denoised patches at their respective image locations.

Forgetting Factor

For non-stationary or highly dynamic data, it may not be desirable to uniformly fit a single transform $\mathbf{W}$ to all the $\mathbf{u}_\tau$, $1 \leq \tau \leq t$, in (P1). Such data can be handled by introducing a forgetting factor $\rho^{t-\tau}$ (with a constant $0 < \rho < 1$) that scales the terms in (P1) [28]. The forgetting factor diminishes the influence of "old" data. The objective function in this case is modified as

$$\frac{1}{C_t}\sum_{\tau=1}^{t} \rho^{t-\tau}\left\{\|\mathbf{W}\mathbf{u}_\tau - \mathbf{x}_\tau\|_2^2 + \lambda_\tau \nu(\mathbf{W}) + \alpha_\tau^2 \|\mathbf{x}_\tau\|_0\right\}, \tag{4.1}$$

[1]This is because only the signal $\tilde{\mathbf{u}}_t$ is assumed to be stored in memory at time t for the online scheme.

where $C_t = \sum_{\tau=1}^{t} \rho^{t-\tau}$ is the normalization factor.

4.2.2 Mini-Bsatch Learning

Another useful variation of Problem (P1) involves *mini-batch* learning, where a block (group), or *mini-batch* of signals is processed at a time [28]. Assuming a fixed mini-batch size M, the Lth ($L \geq 1$) mini-batch of signals is $\mathbf{U}_L = \left[\mathbf{u}_{LM-M+1} \mid \mathbf{u}_{LM-M+2} \mid \;\; \ldots \;\; \mid \mathbf{u}_{LM}\right]$. For $L = 1, 2, 3$, etc., the mini-batch sparsifying transform learning problem is

$$\text{(P2)} \quad \left\{\hat{\mathbf{W}}_L, \hat{\mathbf{X}}_L\right\} = \underset{\mathbf{W}, \mathbf{X}_L}{\arg\min} \frac{1}{LM} \sum_{j=1}^{L} \|\mathbf{W}\mathbf{U}_j - \mathbf{X}_j\|_F^2 \\ + \frac{1}{LM} \sum_{l=1}^{LM} \alpha_l^2 \|\mathbf{x}_l\|_0 + \frac{1}{LM} \sum_{j=1}^{L} \Lambda_j \, \nu(\mathbf{W}),$$

where the regularizer weight is $\Lambda_j = \lambda_0 \left\|\mathbf{U}_j\right\|_F^2$, and the matrix $\mathbf{X}_L = \left[\mathbf{x}_{LM-M+1} \mid \mathbf{x}_{LM-M+2} \mid \;\; \ldots \;\; \mid \mathbf{x}_{LM}\right]$ contains the block of sparse codes corresponding to $\mathbf{U}_L$.

Since we only consider a finite number of frames or patches in practice (e.g., in the proposed VIDOSAT algorithms), the normalizations by $1/t$ in (P1), $1/C_t$ in (4.1), and $1/LM$ in (P2) correspondingly have no effect on the optimum $\left\{\hat{\mathbf{W}}_t, \hat{\mathbf{X}}_t\right\}$ or $\left\{\hat{\mathbf{W}}_L, \hat{\mathbf{X}}_L\right\}$. Thus we drop, for clarity,[2] normalization factors from (P3) and all subsequent expressions for the cost functions.

Once (P2) is solved, a simple denoised estimate of the noisy block of signals in $\mathbf{U}_L$ is obtained as $\hat{\mathbf{U}}_L = \hat{\mathbf{W}}_L^{-1}\hat{\mathbf{X}}_L$. The mini-batch transform learning Problem (P2) is a generalized version of (P1), with (P2) being equivalent to (P1) for $M = 1$. Similar to (4.1), (P2) can be modified to include a forgetting factor. Mini-batch learning can provide potential speedups over the $M = 1$ case in applications, but this comes at the cost of higher memory requirements and latency (i.e., delay in producing output) [28].

[2]In practice, such normalizations may still be useful to control the dynamic range of various internal variables in the algorithm.

4.3 VIDOSAT Framework and Formulations

Prior work on adaptive sparsifying transform-based image denoising [5,23,28] adapted the transform operator to 2D image patches. However, in video denoising, exploiting the sparsity and redundancy in both the spatial and temporal dimensions typically leads to better performance than denoising each frame separately [31]. We therefore propose an online approach to video denoising by learning a sparsifying transform on appropriate 3D spatio-temporal patches.

4.3.1 Video Streaming and Denoising Framework

Figure 4.2 illustrates the framework of our proposed online denoising scheme for streaming videos. The frames of the noisy video (assumed to be corrupted by additive i.i.d. Gaussian noise) denoted as $\mathbf{Y}_\tau \in \mathbb{R}^{a\times b}$ arrive at $\tau = 1, 2, 3$, etc. At time $\tau = t$, the newly arrived frame $\mathbf{Y}_t$ is added to a fixed-size FIFO (first in first out) buffer (i.e., queue) that stores a block of m consecutive frames $\left\{\mathbf{Y}_i\right\}_{i=t-m+1}^{t}$. The oldest (leftmost) frame is dropped from the buffer at each time instant. We denote the spatio-temporal tensor or 3D array obtained by stacking noisy frames along the temporal dimension as $\mathcal{Y}_t = \left[\mathbf{Y}_{t-m+1} \mid \; ... \; \mid \mathbf{Y}_t\right] \in \mathbb{R}^{a\times b\times m}$. We denoise the noisy array $\mathcal{Y}_t$ using the proposed VIDOSAT mini-batch denoising algorithms (denoted by the red box in Fig. 4.2) that are discussed in Sections 4.3.2 and 4.4. These algorithms denoise groups (mini-batches) of 3D patches sequentially and adaptively, by learning sparsifying transforms. Overlapping patches are used in our framework.

The patches output by the mini-batch denoising algorithms are deposited at their corresponding spatio-temporal locations in the fixed-size FIFO output $\bar{\mathcal{Y}}_t = \left[\bar{\mathbf{Y}}_{t-m+1} \mid \; ... \; \mid \bar{\mathbf{Y}}_t\right]$ by adding them to the contents of $\bar{\mathcal{Y}}_t$. We call this process *patch aggregation*. The streaming scheme then outputs the oldest frame $\bar{\mathbf{Y}}_{t-m+1}$. The denoised estimate $\hat{\mathbf{Y}}_{t-m+1}$ is obtained by normalizing $\bar{\mathbf{Y}}_{t-m+1}$ pixel-wise by the number of occurrences of each pixel in the aggregated patches (see Section 4.4 for details).

Though any frame could be denoised and output from $\bar{\mathcal{Y}}_t$ instantaneously, we observe improved denoising quality by averaging over multiple denoised estimates at different time. Figure 4.3 illustrates how the output buffer varies

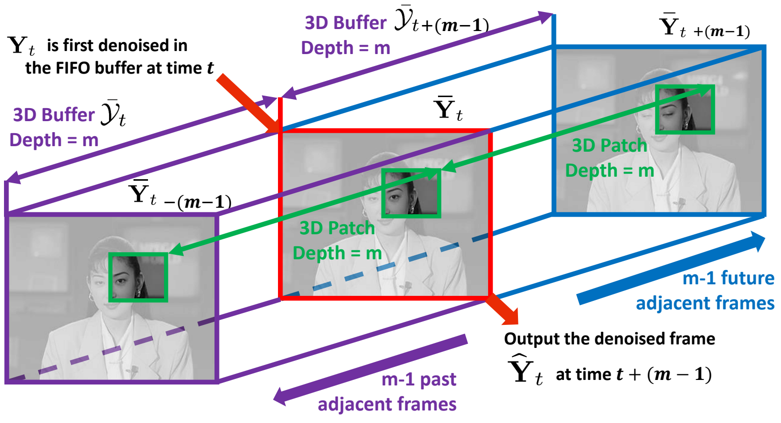

$\mathbf{Y}_t$ is first denoised in the FIFO buffer at time t
3D Buffer $\bar{\mathcal{Y}}_{t+(m-1)}$ Depth = m
$\bar{\mathbf{Y}}_{t+(m-1)}$
3D Buffer $\bar{\mathcal{Y}}_t$ Depth = m
$\bar{\mathbf{Y}}_t$
$\bar{\mathbf{Y}}_{t-(m-1)}$
3D Patch Depth = m
3D Patch Depth = m
m-1 future adjacent frames
Output the denoised frame $\hat{\mathbf{Y}}_t$ at time $t+(m-1)$
m-1 past adjacent frames

Figure 4.3: Illustration of the output buffer from time t to $t + (m - 1)$ for generating the denoised frame output $\hat{\mathbf{Y}}_t$.

from time t to $t+(m-1)$, to output the denoised $\hat{\mathbf{Y}}_t$. In practice, we set the length of the output buffer $\bar{\mathcal{Y}}$ to be the same as the 3D patch depth m, such that each denoised frame $\hat{\mathbf{Y}}_t$ is output by averaging over its estimates from all 3D patches that group the tth frame with $m-1$ adjacent frames. We refer to this scheme as "two-sided" denoising, since the tth frame is denoised together with both past and future adjacent frames ($m-1$ frames on each side), which are highly correlated. Now, data from frame $\mathbf{Y}_t$ is contained in 3D patches that also contain data from frame $\mathbf{Y}_{t+m-1}$. Once these patches are denoised, they will contribute (by aggregation into the output buffer) to the final denoised frame $\hat{\mathbf{Y}}_t$. Therefore, we must wait for frame $\mathbf{Y}_{t+m-1}$ before producing the final estimate $\hat{\mathbf{Y}}_t$. Thus there is a delay of $m-1$ frames between the arrival of the noisy $\mathbf{Y}_t$ and the generation of its final denoised estimate $\hat{\mathbf{Y}}_t$.

4.3.2 VIDOSAT Mini-Batch Denoising Formulation

Here, we discuss the mini-batch denoising formulation that is a core part of the proposed online video denoising framework. For each time instant t, we denoise P partially overlapping size $n_1 \times n_2 \times m$ 3D patches of $\mathcal{Y}_t$ whose vectorized versions are denoted as $\left\{\mathbf{v}_p^t\right\}_{p=1}^P$, with $\mathbf{v}_p^t \in \mathbb{R}^n$, $n = mn_1n_2$. We sequentially process disjoint groups of M such patches, and the groups or mini-batches of patches (total of N mini-batches, where $P = MN$) are denoted as $\left\{\mathbf{U}_{L_k^t}\right\}_{k=1}^{N}$, with $\mathbf{U}_{L_k^t} \in \mathbb{R}^{n\times M}$. Here, k is the *local* mini-batch index within the set of P patches of $\mathcal{Y}_t$, whereas $L_k^t \triangleq N \times (t-1) + k$ is the *global* mini-batch index, identifiying the mini-batch in both time t and location within the set of P patches of $\mathcal{Y}_t$.

For each t, we solve the following online transform learning problem for each $k = 1, 2, 3, ..., N$, to adapt the transform and sparse codes sequentially to the mini-batches in $\mathcal{Y}_t$:

$$\text{(P3)} \quad \left\{\hat{\mathbf{W}}_{L_k^t}, \hat{\mathbf{X}}_{L_k^t}\right\} = \underset{\mathbf{W},\mathbf{X}_{L_k^t}}{\arg\min} \sum_{j=1}^{L_k^t} \rho^{L_k^t - j} \left\|\mathbf{W}\mathbf{U}_j - \mathbf{X}_j\right\|_F^2$$
$$+ \sum_{j=1}^{L_k^t} \rho^{L_k^t - j} \left\{ \Lambda_j \, \nu(\mathbf{W}) + \sum_{i=1}^{M} \alpha_{j,i}^2 \left\|\mathbf{x}_{j,i}\right\|_0 \right\}.$$

Here, the transform is adapted based on patches from *all* the *observed* $\mathbf{Y}_\tau$, $1 \leq \tau \leq t$. The matrix $\mathbf{X}_j = \left[\mathbf{x}_{j,1} \mid ... \mid \mathbf{x}_{j,M}\right] \in \mathbb{R}^{n\times M}$ denotes the transform sparse codes corresponding to the mini-batch $\mathbf{U}_j$. The sparsity penalty weight $\alpha_{j,i}^2$ in (P3) controls the number of non-zeros in $\mathbf{x}_{j,i}$. We set $\alpha_{j,i} = \alpha_0 \sigma_{j,i}$, where α_0 is a constant and $\sigma_{j,i}$ is the noise standard deviation for each patch. We use a forgetting factor $\rho^{L_k^t - j}$ in (P3) to diminish the influence of old frames and old mini-batches.

Once (P3) is solved, the denoised version of the current noisy mini-batch $\hat{\mathbf{U}}_{L_k^t}$ is computed. The columns of the denoised $\hat{\mathbf{U}}_{L_k^t}$ are tensorized and aggregated at the corresponding spatial and temporal locations in the output FIFO buffer. Section 4.4 next discusses the proposed VIDOSAT algorithms in full detail.

4.4 Denoising Algorithms

We now discuss two video denoising algorithms, namely VIDOSAT and VIDOSAT-BM. VIDOSAT-BM uses block matching to generate the 3D patches from $\mathcal{Y}_t$. Though these methods differ in the way they construct the 3D patches, and the way the denoised patches are aggregated in the output FIFO, they both denoise groups of 3D patches sequentially by solving (P3). The VIDOSAT denoising algorithm (without BM) is summarized in Algorithm 4.1.[3] The VIDOSAT-BM algorithm, a modified version of Algorithm 4.1, is discussed in Section 4.4.2.

4.4.1 VIDOSAT

As discussed in Section 4.3.2, the VIDOSAT algorithm processes each mini-batch $\mathbf{U}_j$ in $\mathcal{Y}_\tau$ sequentially. We solve the mini-batch transform learning problem (P3) using a simple alternating minimization approach, with one alternation per mini-batch, which works well and saves computation. Initialized with the most recently estimated transform (warm start), we perform two steps for (P3): Sparse Coding and Mini-batch Transform Update, which compute $\hat{\mathbf{X}}_j$ and update $\hat{\mathbf{W}}_j$, respectively. Then, we compute the denoised mini-batch $\hat{\mathbf{U}}_j$, and aggregate the denoised patches into the output buffer $\bar{\mathcal{Y}}_\tau$.

The major steps of the VIDOSAT algorithm 4.1 for denoising the kth mini-batch $\mathbf{U}_{L_k^t}$ at time t and further processing these denoised patches are described below. To facilitate the exposition and interpretation in terms of the general online denoising algorithm described, various quantities (such as positions of 3D patches in the video stream) are indexed in the text with respect to absolute time t. On the other hand, to emphasize the streaming nature of Algorithm 4.1 and its finite (and modest) memory requirements, indexing of internal variables in the statement of the algorithm is local.

Noisy Mini-Batch Formation

To construct each mini-batch $\mathbf{U}_{L_k^t}$, partially overlapping size $n_1 \times n_2 \times m$ 3D patches of $\mathcal{Y}_t$ are extracted sequentially in a spatially contiguous order (raster scan order with direction reversal on each line).[4] Let $R_p\mathcal{Y}_t$ denote the pth vectorized 3D patch of $\mathcal{Y}_t$, with R_p being the patch-extraction operator. Considering the patch indices $S_k = \Big\{M(k-1)+1, ..., Mk\Big\}$ for the kth mini-batch, we extract $\Big\{\mathbf{v}_p^t = \text{vec}(R_p\mathcal{Y}_t)\Big\}_{p \in S_k}$ as the patches in the mini-batch. Thus $\mathbf{U}_{L_k^t} = \Big[\mathbf{v}_{M(k-1)+1}^t \mid ... \mid \mathbf{v}_{Mk}^t\Big]$. To impose spatio-temporal contiguity

[3]In practice, we wait for the first m frames to be received, before starting Algorithm 4.1, to avoid zero frames in the input FIFO buffer.

[4]We did not observe any marked improvement in denoising performance, when using other scan orders such as raster or Peano-Hilbert scan [89].

```
Algorithm 4.1: VIDOSAT Denoising Algorithm
Input: The noisy frames $\mathbf{Y}_\tau$ ($\tau = 1, 2, 3$, etc.), and the initial transform $\mathbf{W}_0$ (e.g., 3D DCT).
Initialize: $\hat{\mathbf{W}} = \mathbf{W}_0$, $\mathbf{\Gamma} = \mathbf{\Theta} = \mathbf{0}$, $\beta = 0$,
and output buffer $\bar{\mathcal{Y}} = 0$.
For $\tau = 1, 2, 3$, etc., Repeat
The newly arrived frame $\mathbf{Y}_\tau \rightarrow$ latest frame in the input FIFO frame buffer $\mathcal{Y}$.
For $k = 1, ..., N$ Repeat
  Indices of patches in $\mathcal{Y}$:   $S_k = \{M(k-1)+1, ..., Mk\}$.
  1. Noisy Mini-Batch Formation:
      (a) Patch Extraction: $\mathbf{v}_p = \text{vec}(R_p \mathcal{Y}) \quad \forall p \in S_k$.
      (b) $\mathbf{U} = \left[\mathbf{u}_1 \mid ... \mid \mathbf{u}_M\right] \leftarrow \left[\mathbf{v}_{Mk-M+1} \mid ... \mid \mathbf{v}_{Mk}\right]$.
  2. Sparse Coding: $\hat{\mathbf{x}}_i = H_{\alpha_i}(\hat{\mathbf{W}}\mathbf{u}_i) \quad \forall i \in \{1, ..., M\}$.
  3. Mini-batch Transform Update:
      (a) Define $\Lambda \triangleq \lambda_0 \|\mathbf{U}\|_F^2$ and $\hat{\mathbf{X}} \triangleq \left[\hat{\mathbf{x}}_1 \mid ... \mid \hat{\mathbf{x}}_M\right]$.
      (b) $\mathbf{\Gamma} \leftarrow \rho\mathbf{\Gamma} + \mathbf{U}\mathbf{U}^T$.
      (c) $\mathbf{\Theta} \leftarrow \rho\mathbf{\Theta} + \mathbf{U}\hat{\mathbf{X}}^T$.
      (d) $\beta \leftarrow \rho\beta + \Lambda$.
      (e) Matrix square root: $\mathbf{Q} \leftarrow (\mathbf{\Gamma} + \beta\mathbf{I})^{1/2}$.
      (f) Full SVD: $\mathbf{\Phi\Sigma\Psi}^T \leftarrow \text{SVD}(\mathbf{Q}^{-1}\mathbf{\Theta})$.
      (g) $\hat{\mathbf{W}} \leftarrow 0.5\mathbf{\Psi}\left(\mathbf{\Sigma} + (\mathbf{\Sigma}^2 + 2\beta\mathbf{I})^{\frac{1}{2}}\right)\mathbf{\Phi}^T\mathbf{Q}^{-1}$.
  4. 3D Denoised Patch Reconstruction:
      (a) Update Sparse Codes: $\hat{\mathbf{x}}_i = H_{\alpha_i}(\hat{\mathbf{W}}\mathbf{u}_i) \quad \forall i$.
      (b) Denoised mini-batch: $\hat{\mathbf{U}} = \hat{\mathbf{W}}^{-1}\hat{\mathbf{X}}$.
      (c) $\left[\hat{\mathbf{v}}_{M(k-1)+1} \mid ... \mid \hat{\mathbf{v}}_{Mk}\right] \leftarrow \hat{\mathbf{U}}$
      (d) Tensorization: $\hat{\mathcal{V}}_p = \text{vec}^{-1}(\hat{\mathbf{v}}_p) \; \forall p \in S_k$.
  5. Aggregation: Aggregate patches $\{\hat{\mathcal{V}}_p\}$ at corresponding locations:
     $\bar{\mathcal{Y}} \leftarrow \sum_{p \in S_k} R_p^* \hat{\mathcal{V}}_p$.
End
Output: The oldest frame in $\bar{\mathcal{Y}}$ after normalization $\rightarrow$ the denoised frame $\hat{\mathbf{Y}}_{\tau-m+1}$.
End
```

of 3D patches extracted from two adjacent stacks of frames, we reverse the raster scan order (of patches) between $\mathcal{Y}_t$ and $\mathcal{Y}_{t+1}$.

Sparse Coding

Given the sparsifying transform $\mathbf{W} = \hat{\mathbf{W}}_{L_k^t - 1}$ estimated for the most recent mini-batch, we solve Problem (P3) for the sparse coefficients $\hat{\mathbf{X}}_{L_k^t}$:

$$\hat{\mathbf{X}}_{L_k^t} = \underset{\mathbf{X}}{\arg\min} \left\|\mathbf{W}\mathbf{U}_{L_k^t} - \mathbf{X}\right\|_F^2 + \sum_{i=1}^{M} \alpha_{L_k^t,i}^2 \left\|\mathbf{x}_i\right\|_0 . \tag{4.2}$$

A solution for (4.2) is given in closed-form as $\hat{\mathbf{x}}_{L_k^t,i} = H_{\alpha_{L_k^t,i}}(\hat{\mathbf{W}}_{L_k^t}\mathbf{u}_{L_k^t,i})$ $\forall$ i [28]. Here, the hard thresholding operator $H_\alpha(\cdot) : \mathbb{R}^n \to \mathbb{R}^n$ is applied to a vector element-wise, as defined by

$$\left(H_\alpha(\mathbf{d})\right)_r = \begin{cases} 0 & , \ |d_r| < \alpha \\ d_r & , \ |d_r| \geq \alpha \end{cases} . \tag{4.3}$$

This simple hard thresholding operation for transform sparse coding is similar to traditional techniques involving analytical sparsifying transforms [19].

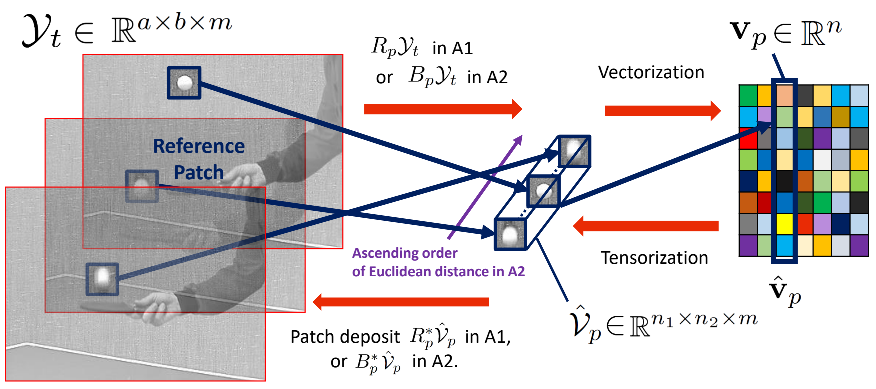
Mini-batch Transform Update

We solve Problem (P3) for $\mathbf{W}$ with fixed $\mathbf{X}_j = \hat{\mathbf{X}}_j$, $1 \leq j \leq L_k^t$, as follows:

$$\min_{\mathbf{W}} \sum_{j=1}^{L_k^t} \rho^{L_k^t - j} \left\{ \left\|\mathbf{W}\mathbf{U}_j - \mathbf{X}_j\right\|_F^2 + \Lambda_j \nu(\mathbf{W}) \right\} . \tag{4.4}$$

This problem has a simple solution (similar to Section III-B2 in [28]). Set index $J = L_t^k$, and define the following quantities: $\mathbf{\Gamma}_J \triangleq \sum_{j=1}^{J} \rho^{J-j} \mathbf{U}_j \mathbf{U}_j^T$, $\mathbf{\Theta}_J \triangleq \sum_{j=1}^{J} \rho^{J-j} \mathbf{U}_j \hat{\mathbf{X}}_j^T$, and $\beta_J \triangleq \sum_{j=1}^{J} \rho^{J-j} \Lambda_j$. Let $\mathbf{Q} \in \mathbb{R}^{n \times n}$ be a square root (e.g., Cholesky factor) of $(\mathbf{\Gamma}_J + \beta_J \mathbf{I})$, i.e., $\mathbf{Q}\mathbf{Q}^T = \mathbf{\Gamma}_J + \beta_J \mathbf{I}$. Denoting the full singular value decomposition (SVD) of $\mathbf{Q}^{-1}\mathbf{\Theta}_J$ as $\mathbf{\Phi}\mathbf{\Sigma}\mathbf{\Psi}^T$, we then have that the closed-form solution to (4.4) is

$$\hat{\mathbf{W}}_J = 0.5 \mathbf{\Psi} \left(\mathbf{\Sigma} + \left(\mathbf{\Sigma}^2 + 2\beta_J \mathbf{I} \right)^{\frac{1}{2}} \right) \mathbf{\Phi}^T \mathbf{Q}^{-1} , \tag{4.5}$$



$\mathcal{Y}_t \in \mathbb{R}^{a\times b\times m}$
$R_p\mathcal{Y}_t$ in A1
or $B_p\mathcal{Y}_t$ in A2
Vectorization
$\mathbf{v}_p \in \mathbb{R}^n$
Reference Patch
Ascending order of Euclidean distance in A2
Tensorization
$\hat{\mathbf{v}}_p$
$\hat{\mathcal{V}}_p \in \mathbb{R}^{n_1\times n_2\times m}$
Patch deposit $R_p^*\hat{\mathcal{V}}_p$ in A1,
or $B_p^*\hat{\mathcal{V}}_p$ in A2.

Figure 4.4: Patch deposit $R_p^* \operatorname{vec}^{-1}(\hat{\mathbf{v}}_p)$ (resp. $B_p^* \operatorname{vec}^{-1}(\hat{\mathbf{v}}_p)$) as an adjoint of patch extraction operator in **4.1** (resp. an adjoint of BM operator in 4.2).

where $\mathbf{I}$ denotes the identity matrix, and $(\cdot)^{\frac{1}{2}}$ denotes the positive definite square root of a positive definite (diagonal) matrix. The quantities $\mathbf{\Gamma}_J$, $\mathbf{\Theta}_J$, and β_J are all computed sequentially over time t and mini-batches k [28].

3D Denoised Patch Reconstruction

We denoise $\mathbf{U}_{L_k^t}$ using the updated transform. First, we repeat the sparse coding step using the updated $\hat{\mathbf{W}}_{L_k^t}$ as $\hat{\mathbf{x}}_{L_k^t,i} = H_{\alpha_{L_k^t,i}}(\hat{\mathbf{W}}_{L_k^t}\mathbf{u}_{L_k^t,i})\ \forall\ i$. Then, with fixed $\hat{\mathbf{W}}_{L_k^t}$ and $\hat{\mathbf{X}}_{L_k^t}$, the denoised mini-batch is obtained in the least squares sense under the transform model as

$$\hat{\mathbf{U}}_{L_k^t} = \hat{\mathbf{W}}_{L_k^t}^{-1}\hat{\mathbf{X}}_{L_k^t}. \tag{4.6}$$

The denoised mini-batch is used to update the denoised (vectorized) 3D patches as $\hat{\mathbf{v}}^t_{M(k-1)+i} = \hat{\mathbf{u}}_{L_k^t,i}\ \forall i$. All reconstructed vectors $\left\{\hat{\mathbf{v}}_p^t\right\}_{p\in S_k}$ from the kth mini-batch denoising result are tensorized as $\left\{\operatorname{vec}^{-1}(\hat{\mathbf{v}}_p^t)\right\}_{p\in S_k}$.

Aggregation

The denoised 3D patches $\left\{\operatorname{vec}^{-1}(\hat{\mathbf{v}}_p^t)\right\}_{p\in S_k}$ from each mini-batch are sequentially aggregated at their corresponding spatial and temporal locations in the output FIFO buffer as $\sum_{p\in S_k} R_p^* \operatorname{vec}^{-1}(\hat{\mathbf{v}}_p^t) \to \bar{\mathcal{Y}}_t \in \mathbb{R}^{a\times b\times m}$, where the

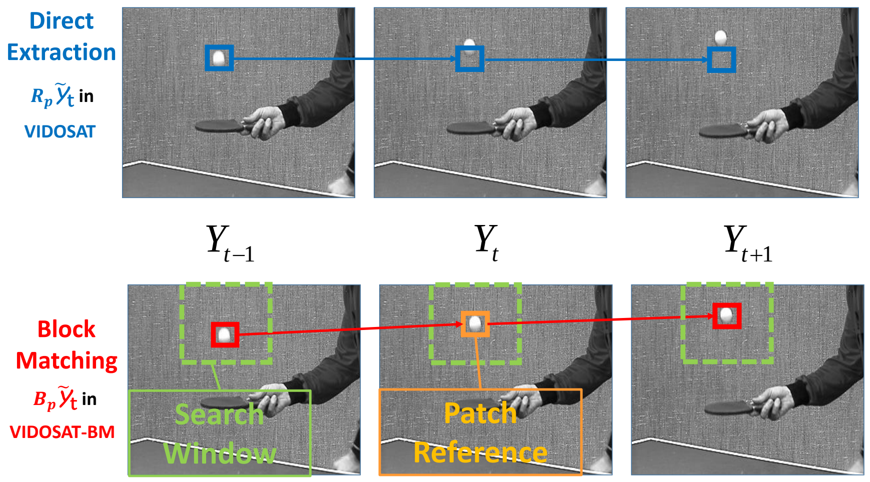


Figure 4.5: Illustration of the different 3D patch construction methods in VIDOSAT (blue) and VIDOSAT-BM (red). The 3D search window used in VIDOSAT-BM is illustrated in green.

adjoint R_p^* is the patch deposit operator. Figure 4.4 illustrates the patch deposit procedure for aggregation.

When all N denoised mini-batches for $\mathcal{Y}_t$ are generated, and the patch aggregation in $\bar{\mathcal{Y}}_t$ completes, the oldest frame in $\bar{\mathcal{Y}}_t$ is normalized pixel-wise by the number of occurrences (which ranges from $2m - 1$, for pixels at the corners of a video frame, to n for pixels away from the borders of a video frame) of that pixel among patches aggregated into the output buffer. This normalized result is output as the denoised frame $\hat{\mathbf{Y}}_{t-m+1}$.

4.4.2 VIDOSAT-BM

For videos with relatively static scenes, each extracted spatio-temporal tensor $R_p\mathcal{Y}_t$ in the VIDOSAT Algorithm 4.1 typically has high temporal correlation, implying high (3D) transform domain sparsity. However, highly dynamic videos usually involve various motions, such as translation, rotation, scaling, etc. Figure 4.5 demonstrates one example when the 3D patch construction strategy in the VIDOSAT denoising algorithm 4.1 fails to capture the properties of the moving object. Thus, Algorithm 4.1 could provide sub-optimal denoising performance for highly dynamic videos. We propose an alternative algorithm, dubbed VIDOSAT-BM, which improves VIDOSAT denoising by constructing 3D patches using block matching.

The proposed VIDOSAT-BM solves the online transform learning problem (P3) with a different methodology for constructing the 3D patches and each mini-batch. The Steps (2) − (4) in Algorithm 4.1 remain the same for VIDOSAT-BM. We now discuss the modified Steps (1) and (5) in the VIDOSAT-BM denoising algorithm, to which we also refer as Algorithm 4.2.

3D Patch and Mini-Batch Formation in VIDOSAT-BM: Here, we use a small and odd-valued sliding (temporal) window size m (e.g., we set $m = 9$ in the video denoising experiments in Section 4.5, which corresponds to ~ 0.2 s buffer duration for a video with 40 Hz frame rate). Within the m-frame input FIFO buffer $\mathcal{Y}_t$, we approximate the various motions in the video using simple (local) translations [90].

We consider the middle frame $\mathbf{Y}_{t-(m-1)/2}$ in the input FIFO buffer $\mathcal{Y}_t$, and sequentially extract all 2D overlapping patches $\mathbf{Z}_p^t \in \mathbb{R}^{n_1 \times n_2}$, $1 \leq p \leq P$ in $\mathbf{Y}_{t-(m-1)/2}$, in a 2D spatially contiguous (raster scan) order. For each $\mathbf{Z}_p^t$, we form a $h_1 \times h_2 \times m$ pixel local search window centered at the center of $\mathbf{Z}_p^t$ (see the illustration in Fig. 4.5). We apply a spatial BM operator, denoted B_p, to find (using exhaustive search) the $(m-1)$ patches, one for each neighboring frame in the search window, that are most similar to $\mathbf{Z}_p^t$ in Euclidean distance. The operator B_p stacks the $\mathbf{Z}_p^t$, followed by the $(m-1)$ matched patches, in an ascending order of their Euclidean distance to $\mathbf{Z}_p^t$, to form the pth 3D patch $B_p\mathcal{Y}_t \in \mathbb{R}^{n_1 \times n_2 \times m}$. Similar BM approaches have been used in prior works on video compression (e.g., MPEG) for motion compensation [90], and in recent works on spatiotemporal medical imaging [51]. The coordinates of all selected 2D patches are recorded to be used later in the denoised patch aggregation step. Instead of constructing the 3D patches from 2D patches in corresponding locations in contiguous frames (i.e., $R_p\mathcal{Y}_t$ in Algorithm 4.1), we form the patches using BM and work with the vectorized $\mathbf{v}_p^t = \text{vec}(B_p\mathcal{Y}_t) \in \mathbb{R}^n$ in VIDOSAT-BM. The k-th mini-batch is defined, as in Algorithm 4.1 as $\mathbf{U}_{L_k^t} = \left[\mathbf{v}_{M(k-1)+1}^t \mid ... \mid \mathbf{v}_{Mk}^t\right]$.

Aggregation: Each denoised 3D patch (tensor) of $\left\{\text{vec}^{-1}(\hat{\mathbf{v}}_p^t)\right\}_{p \in S_k}$ contains the matched (and denoised) 2D patches. They are are sequentially aggregated at their recorded spatial and temporal locations in the output FIFO buffer $\bar{\mathcal{Y}}_t$ as $\sum_{p \in S_k} B_p^* \text{vec}^{-1}(\hat{\mathbf{v}}_p^t) \to \bar{\mathcal{Y}}_t \in \mathbb{R}^{a \times b \times m}$, where the adjoint B_p^*

Table 4.2: Comparison of video denoising PSNR values (in dB), averaged over the ASU dataset, for the proposed VIDOSAT, VIDOSAT-BM, and other competing methods. For each dataset and noise level, the best denoising PSNR is marked in bold. For each method, we list Δ PSNR, which denotes the average PSNR difference (with its standard deviation included in parentheses) relative to the proposed VIDOSAT-BM (highlighted in bold).

Data	**ASU** Dataset (26 videos)					ΔPSNR
σ	5	10	15	20	50	(std.)
fBM3D [32]	38.78	34.66	32.38	30.82	26.13	3.89 (1.41)
sKSVD [31]	41.27	37.37	35.15	33.59	28.79	1.20 (0.34)
3D DCT	41.26	37.14	34.73	33.03	27.59	1.69 (0.78)
VBM3D [87]	41.10	37.82	35.78	34.25	28.65	0.92 (0.72)
VBM4D [59]	41.42	37.59	35.30	33.64	27.76	1.30 (0.86)
VIDOSAT	41.94	38.32	36.13	34.60	29.87	0.27 (0.13)
VIDOSAT -BM	**42.22**	**38.57**	**36.42**	**34.88**	**30.09**	0

is the patch deposit operator in 4.2. Fig. 4.4 illustrates the patch deposit procedure for aggregation in 4.2. Once the aggregation of $\bar{\mathcal{Y}}_t$ completes, the oldest frame in $\bar{\mathcal{Y}}_t$ is normalized pixel-wise by the number of occurrences of each pixel among patches in the denoising algorithm. Unlike Algorithm 4.1 where this number of occurrences is the same for all frames, in Algorithm 4.2 this number is data-dependent and varies from frame to frame and pixel to pixel. We record the number of occurrences of each pixel which is based on the recorded locations of the matched patches, and can be computed online as described. The normalized oldest frame is output by Algorithm 4.2 for each time instant.

Table 4.3: Comparison of video denoising PSNR values (in dB), averaged over the LASIP dataset, for the proposed VIDOSAT, VIDOSAT-BM, and other competing methods. For each dataset and noise level, the best denoising PSNR is marked in bold. For each method, we list Δ PSNR, which denotes the average PSNR difference (with its standard deviation included in parentheses) relative to the proposed VIDOSAT-BM (highlighted in bold).

Data	**LASIP** Dataset (8 videos)					ΔPSNR
σ	5	10	15	20	50	(std.)
fBM3D [32]	38.05	34.06	31.89	30.42	25.88	2.11 (1.03)
sKSVD [31]	38.87	34.95	32.80	31.33	26.89	1.21 (0.38)
3D DCT	38.01	33.60	30.44	28.50	22.31	3.60 (1.28)
VBM3D [87]	39.20	35.75	33.87	32.49	26.51	0.61 (0.51)
VBM4D [59]	39.37	35.73	33.70	32.24	26.68	0.63 (0.49)
VIDOSAT	39.56	35.75	33.54	31.98	27.29	0.55 (0.29)
VIDOSAT -BM	**39.95**	**36.11**	**34.05**	**32.60**	**28.15**	0

4.4.3 Computational Costs

In Algorithm 4.1, the computational cost of the sparse coding step is dominated by the computation of matrix-vector multiplication $\hat{\mathbf{W}}\mathbf{u}_i$, which scales as $O(Mn^2)$ [6, 28] for each mini-batch. The cost of mini-batch transform update step is $O(n^3+Mn^2)$, which is dominated by full SVD and matrix-matrix multiplications. The cost of the 3D denoised patch reconstruction step also scales as $O(n^3 + Mn^2)$ per mini-batch, which is dominated by the computation of matrix inverse $\hat{\mathbf{W}}^{-1}$ and multiplications. As all overlapping patches from a $a \times b \times T$ video are sequentially processed, the computational cost of Algorithm 4.1 scales as $O(abTn^3/M + abTn^2)$. We set $M = 15n$ in practice, so that the cost of 4.1 scales as $O(abTn^2)$. The cost of the additional BM step in Algorithm 4.2 scales as $O(abTmh_1h_2)$, where $h_1 \times h_2$ is the search window size. Therefore, the total cost of 4.2 scales as $O(abTn^2 + abTmh_1h_2)$, which is on par with the state-of-the-art video denoising algorithm VBM3D [87], which is not an online method.

4.5 Experiments

4.5.1 Implementation and Parameters

Testing Data

We present experimental results demonstrating the promise of the proposed VIDOSAT and VIDOSAT-BM online video denoising methods. We evaluate the proposed algorithms by denoising all 34 videos from 2 public datasets, including 8 videos from the LASIP video dataset [5] [59, 87], and 26 videos the Arizona State University (ASU) Video Trace Library [6] [91]. The testing videos contain 50 to 870 frames, with the frame resolution ranging from 176×144 to 720×576. Each video involves different types of motion, including translation, rotation, scaling (zooming), etc. The color videos are all converted to gray-scale. We simulated i.i.d. zero-mean Gaussian noise at 5 different noise levels (with standard deviation $\sigma = 5$, 10, 15, 20, and 50) for each video.

Implementation Details

We include several minor modifications of VIDOSAT and VIDOSAT-BM algorithms for improved performance. At each time instant t, we perform multiple passes of denoising for each $\mathcal{Y}_t$ by iterating over Steps (1) to (5) multiple times. In each pass, we denoise the output from the previous iteration [5, 28]. As the sparsity penalty weights are set proportional to the noise level, $\alpha_{j,i} = \alpha_0 \sigma$, the noise standard deviation σ in each such pass is set to an

[5]Available at `http://www.cs.tut.fi/~lasip/foi_wwwstorage/test_videos.zip`

[6]Available at `http://trace.eas.asu.edu/yuv/`. Only videos with less than 1000 frames are selected for our image denoising experiments.

empirical estimate [5, 6] of the remaining noise in the denoised frames from the previous pass. These multiple passes, although increasing the computation in the algorithm, do not increase the inherent latency $m-1$ of the single pass algorithm described earlier.

The following details are specifically for VIDOSAT-BM. First, instead of performing BM over the noisy input buffer $\mathcal{Y}_t$, we pre-clean $\mathcal{Y}_t$ using the VIDOSAT mini-batch denoising Algorithm 4.1, and then perform BM over the VIDOSAT denoised output. Second, when denoised 3D patches are aggregated to the output buffer, we assign them different weights, which are proportional to the sparsity level of their optimal sparse codes [39]. The weights are also accumulated and used for the output normalization.

Hyperparameters

We work with fully overlapping patches (spatial patch stride of 1 pixel) with spatial size $n_1 = n_2 = 8$, and temporal depth of $m = 9$ frames, which also corresponds to the depth of buffer $\mathcal{Y}$. It follows that for a video with $N_1 \times N_2$ frames, the buffer $\mathcal{Y}$ contains mN_1N_2 pixels, and $P = (N_1 - n_1 + 1)(N_2 - n_2 + 1)$ 3D patches. We set the sparsity penalty weight parameter $\alpha_0 = 1.9$, the transform regularizer weight constant $\lambda_0 = 10^{-2}$, and the mini-batch size $M = 15 \times mn_1n_2$. The transform $\mathbf{W}$ is initialized with the 3D DCT $\mathbf{W}_0$. For the other parameters, we adopt the settings in prior works [5,6,28], such as the forgetting factor $\rho = 0.68, 0.72, 0.76, 0.83, 0.89$, and the number of passes $L_p = 1, 2, 3, 3, 4$ for $\sigma = 5, 10, 15, 20, 50$, respectively. The values of ρ and L_p both increase as the noise level increases. The larger ρ helps prevent overfitting to noise, and the larger number of passes improves denoising performance at higher noise level. For VIDOSAT-BM, we set the local search window size $h_1 = h_2 = 21$.

4.5.2 Video Denoising Results

Competing Methods

We compare the video denoising results obtained using the proposed VIDOSAT and VIDOSAT-BM algorithms to several well-known alternatives, including the frame-wise BM3D denoising method (fBM3D) [32], the image sequence denoising method using sparse KSVD (sKSVD) [31], VBM3D [87] and VBM4D methods [59]. We used the publicly available implementations of these methods. Among these competing methods, fBM3D denoises each frame independently by applying a popular BM3D image denoising method; sKSVD exploits adaptive spatio-temporal sparsity but the dictionary is not learned online; and VBM3D and VBM4D are popular and state-of-the-art video denoising methods. Moreover, to better understand the advantages of the online high-dimensional transform learning, we apply the proposed video denoising framework, but fixing the sparsifying transform in VIDOSAT to 3D DCT, which is referred as the 3D DCT method.

Denoising Results

We present video denoising results using the proposed VIDOSAT and VIDOSAT-BM algorithms, as well as using the other aforementioned competing methods. To evaluate the performance of the various denoising schemes, we measure the peak signal-to-noise ratio (PSNR) in decibels (dB), which is computed between the noiseless reference and the denoised video.

Tables 4.2 and 4.3 list the video denoising PSNRs obtained by the two proposed VIDOSAT methods, as well as the five competing methods. It is clear that the proposed VIDOSAT and VIDOSAT-BM approaches both generate better denoising results with higher average PSNR values, compared to the competing methods. The VIDOSAT-BM denoising method provides average PSNR improvements (averaged over all 34 testing videos from both datasets and all noise levels) of 0.9 dB, 1.1 dB, 1.2 dB, 2.1 dB, and 3.5 dB, over the VBM3D, VBM4D, sKSVD, 3D DCT, and fBM3D denoising methods. Importantly, VIDOSAT-BM consistently outperforms all the competing methods for all testing videos and noise levels. Among the two proposed VIDOSAT algorithms, the average video denoising PSNR by VIDOSAT-BM is 0.3 dB higher than that using the VIDOSAT method, thanks to the use of the block matching for modeling dynamics and motion in video.

We illustrate the denoising results and improvements provided by VIDOSAT and VIDOSAT-BM with some examples.

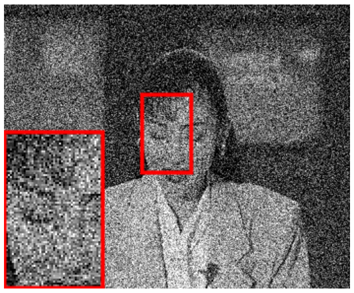
Figure 4.6 shows one denoised frame of the video *Akiyo* ($\sigma = 50$), which involves static background and a relatively small moving region (the magnitudes of error in Fig. 4.6 are clipped for viewing). The denoising results by VIDOSAT and VIDOSAT-BM both demonstrate similar visual quality improvements over the result by VBM3D. Figure 4.7(a) shows the frame-by-frame PSNRs of the denoised *Akiyo*, in which VIDOSAT and VIDOSAT-BM provide comparable denoising PSNRs, and both outperform the VBM3D and VBM4D schemes consistently by a sizable margin.

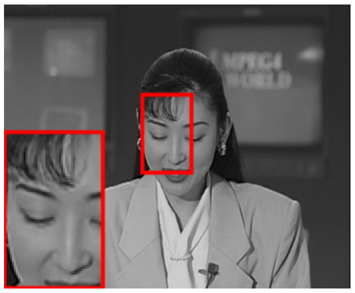
Figure 4.8 shows one denoised frame of the video *Salesman* ($\sigma = 20$) that involves occasional but fast movements (e.g., hand waving) in the foreground. The denoising result by VIDOSAT improves over the VBM4D result in general, but also shows some artifacts in regions with strong motion. Instead, the result by VIDOSAT-BM provides the best visual quality in both the static and the moving parts. Fig. 4.7(b) shows the frame-by-frame PSNRs of the denoised *Salesman*. VIDOSAT-BM provides large improvements over the other methods including VIDOSAT for most frames, and the PSNR is more stable (smaller deviations) over frames. Figure 4.9 shows example atoms (i.e., rows) of the initial 3D DCT transform, and the online learned transforms using (a) VIDOSAT and (b) VIDOSAT-BM at different times t. For the learned $\hat{\mathbf{W}}_t$'s using both VIDOSAT and VIDOSAT-BM, their atoms are observed to gradually evolve, in order to adapt to the dynamic video content. The learned transform atoms using VIDOSAT in Fig. 4.9(a) demonstrate linear shifting structure along the patch depth m, which is likely to compensate the video motion (e.g., translation). On the other hand, since the 3D patches are formed using BM in VIDOSAT-BM, such structure is not observed in Fig. 4.9(b) when $\hat{\mathbf{W}}_t$ is learned using VIDOSAT-BM.

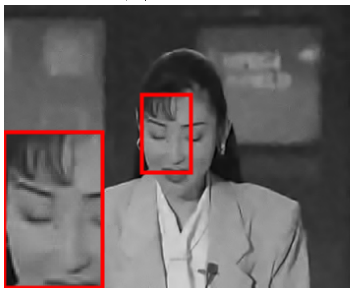
Figure 4.10 shows one denoised frame of the video *Bicycle* ($\sigma = 20$), which contains a large area of complex movements (e.g., rotations) throughout the video. In this case, the denoised frame using the VIDOSAT is worse than VBM4D. However, VIDOSAT-BM provides superior quality compared to all the methods. This example demonstrates the effectiveness of joint block matching and learning in the proposed VIDOSAT-BM scheme, especially when processing highly dynamic videos. Fig. 4.7(c) shows the frame-by-frame PSNRs of the denoised *Bicycle*, in which VIDOSAT-BM significantly improves over VIDOSAT, and also outperforms both VBM3D and VBM4D for all frames.

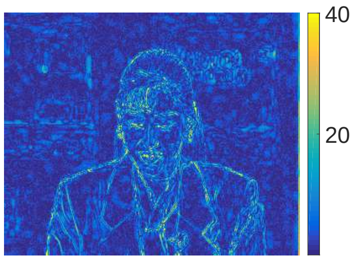
4.6 Conclusions

We presented a novel framework for online video denoising based on efficient high-dimensional sparsifying transform learning. The transforms are learned in an online manner from spatio-temporal patches. These patches are constructed either from corresponding 2D patches of consecutive frames or using an online block matching technique. The learned models effectively capture the dynamic changes in videos. We demonstrated the promising performance of the proposed video denoising schemes for several standard datasets. Our methods outperformed all compared methods, which included a version of the proposed video denoising scheme in which the learning of the sparsifying transform was eliminated and instead it was fixed to 3D DCT, as well as denoising using learned synthesis dictionaries, and the state-of-the-art VBM3D and VBM4D methods. While this work provides an initial study of the promise of the proposed data-driven online video denoising methodologies, we plan to study the potential implementation and acceleration of the proposed schemes for real-time video processing in future work.

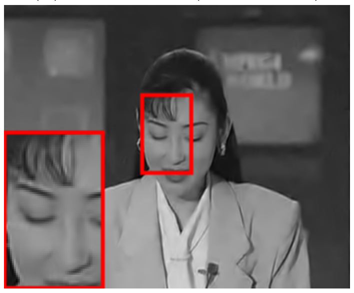

(a) Noisy

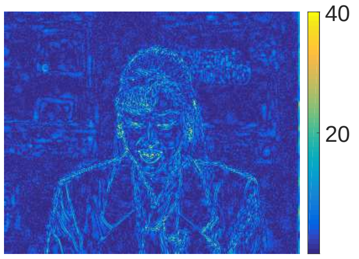

(b) Original


(c) VBM3D (33.30 dB)

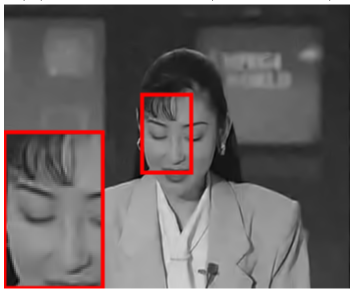


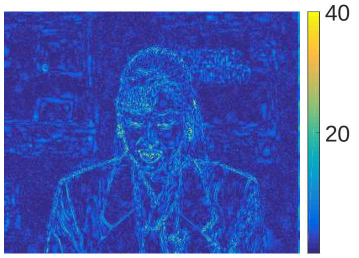
(d)


(e) VIDOSAT (35.84 dB)



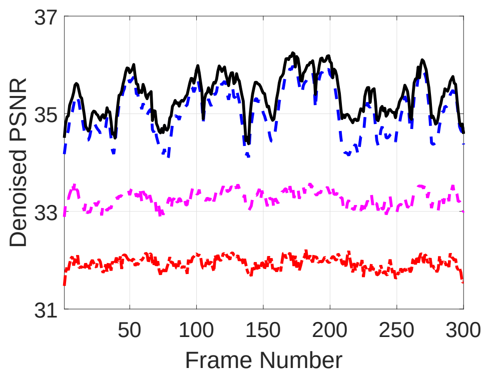
(f)


(g) VIDOSAT-BM (36.11 dB)

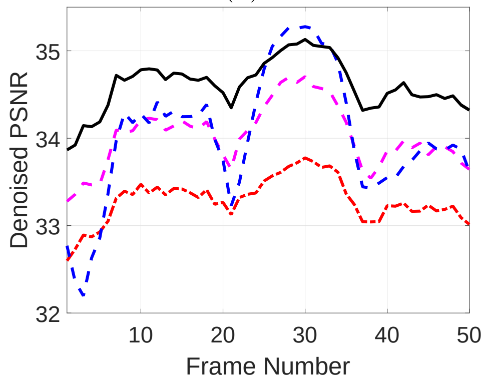


(h)

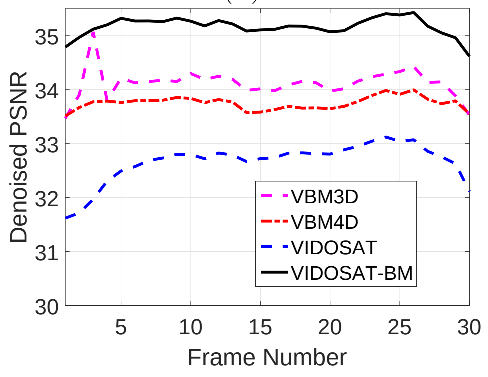
Figure 4.6: (a) The noisy version ($\sigma = 50$) of (b) one frame of the *Akiyo* ($288 \times 352 \times 300$) video. We show the comparison of the denoising results (resp. the magnitude of error in the denoised frame) using (c) VBM3D (33.30 dB), (e) VIDOSAT (35.84 dB) and (g) VIDOSAT-BM (36.11 dB) (resp. (d), (f) and (h)). The PSNR of the denoised frame is shown in the parentheses. The zoom-in region is highlighted using red box.



(a)

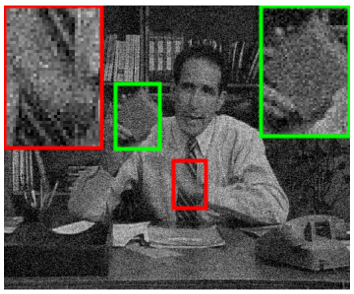


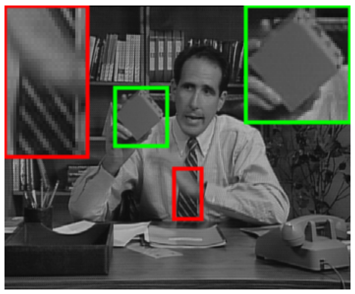
(b)

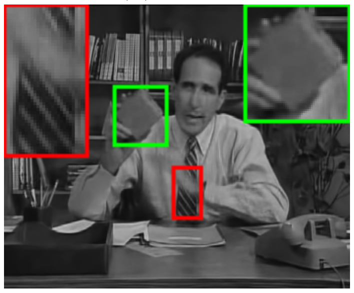


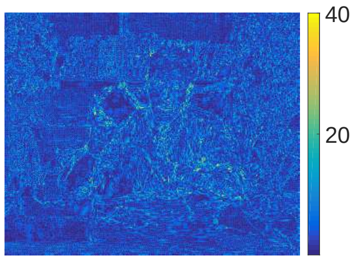
(c)

Figure 4.7: Frame-by-frame PSNR (dB) for (a) *Akiyo* with $\sigma = 50$, (b) *Salesman* with $\sigma = 20$, and (c) *Bicycle* with $\sigma = 20$, denoised by VBM3D, VBM4D, and the proposed VIDOSAT and VIDOSAT-BM schemes, respectively.

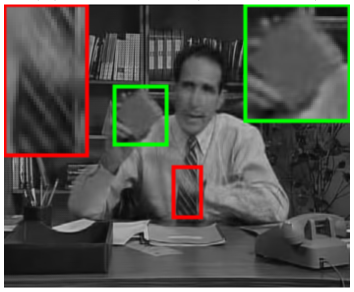

(a) Noisy

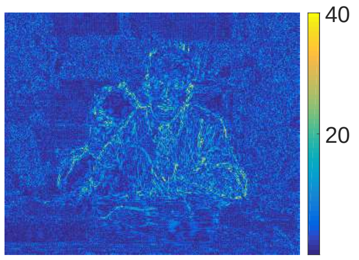

(b) Original


(c) VBM4D (33.04 dB)

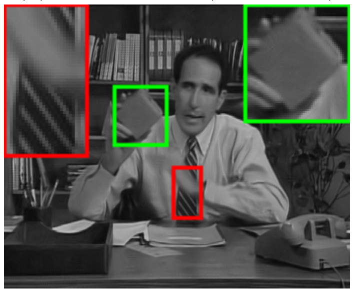


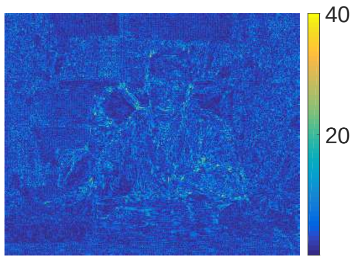
(d)


(e) VIDOSAT (33.43 dB)



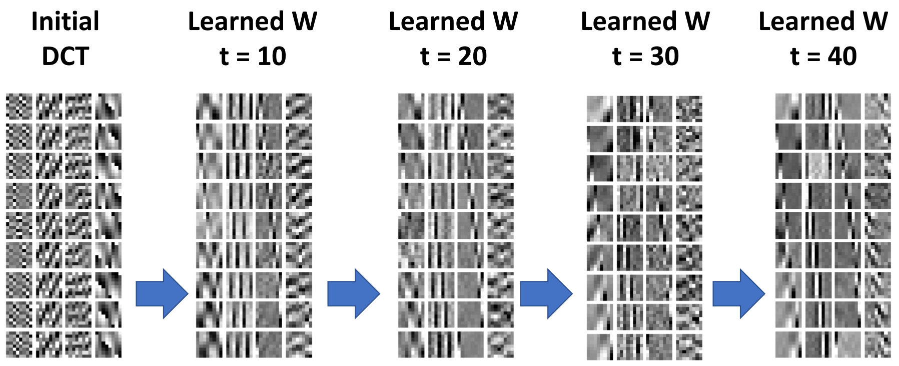
(f)


(g) VIDOSAT-BM (34.01 dB)

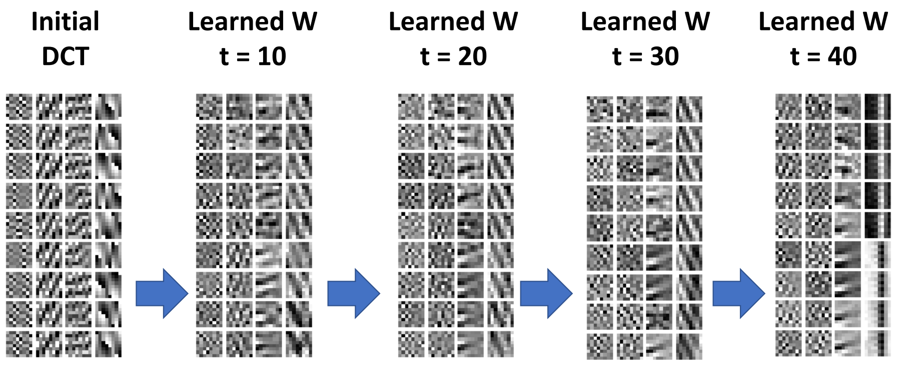


(h)

Figure 4.8: (a) The noisy version ($\sigma = 20$) of (b) one frame of the *Salesman* ($288 \times 352 \times 50$) video. We show the comparison of the denoising results (resp. the magnitude of error in the denoised frame) using (c) VBM4D (33.04 dB), (e) VIDOSAT (33.43 dB) and (g) VIDOSAT-BM (34.01 dB) (resp. (d), (f) and (h)). The PSNR of the denoised frame is shown in the parentheses. The zoom-in regions are highlighted using red and green boxes.

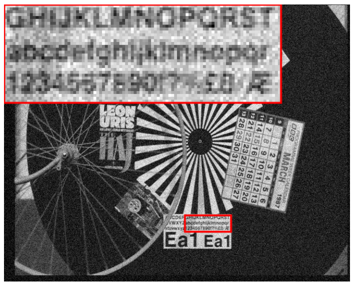


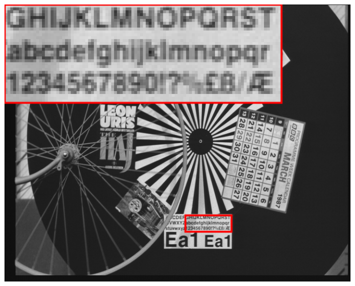
(a) $\hat{\mathbf{W}}_t$ learned using VIDOSAT

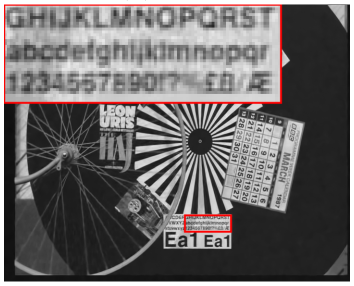


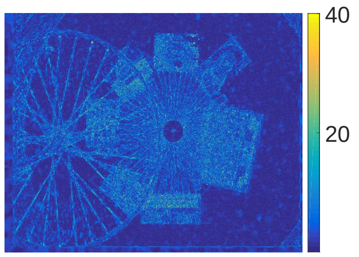
(b) $\hat{\mathbf{W}}_t$ learned using VIDOSAT-BM

Figure 4.9: Example atoms (i.e., 4 rows) of the initial 3D DCT (with depth $m = 9$), and the online learned 3D sparsifying transform using (a) VIDOSAT, and (b) VIDOSAT-BM, at times 10 to 40: the atoms (i.e., rows) of the learned $\hat{\mathbf{W}}$ are shown as $m = 9$ patches in each column. These 9 patches together form the $8 \times 8 \times 9$ 3D atoms.

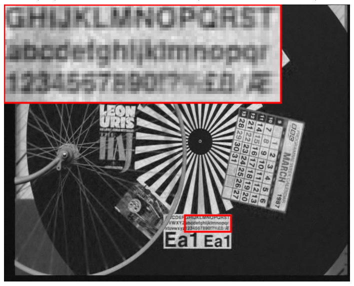

(a) Noisy

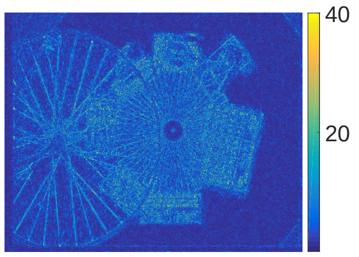

(b) Original


(c) VBM4D (34.00 dB)

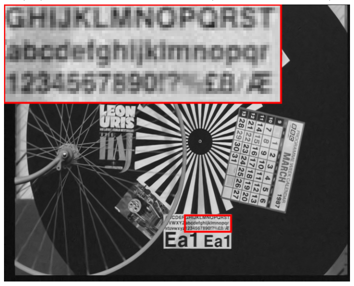


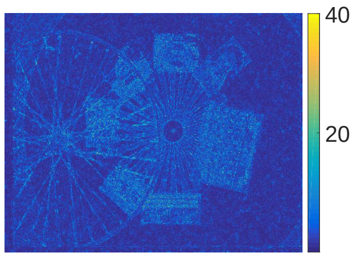
(d)


(e) VIDOSAT (32.07 dB)



(f)


(g) VIDOSAT-BM (35.33 dB)



(h)

Figure 4.10: (a) The noisy version ($\sigma = 20$) of (b) one frame of the *Bicycle* ($576 \times 720 \times 30$) video. We show the comparison of the denoising results (resp. the magnitude of error in the denoised frame) using (c) VBM4D (34.00 dB), (e) VIDOSAT (32.07 dB) and (g) VIDOSAT-BM (35.33 dB) (resp. (d), (f) and (h)). The PSNR of the denoised frame is shown in the parentheses. The zoom-in region is highlighted using red box.

CHAPTER 5

TRANSFORM LEARNING WITH NON-LOCAL LOW-RANK CONSTRAINT FOR IMAGE RESTORATION

5.1 Related Work

5.1.1 Image Denoising

Image denoising is one of the most important problems in image processing and low-level computer vision. It is dedicated to recovering high-quality images from their corrupted measurements, which also improves robustness in various high-level vision tasks [54]. The image denoising algorithms can be divided into *internal* and *external* methods [92–95].

Internal methods make use of only the noisy image to be reconstructed. Classical algorithms exploit image local structures using total variation (TV) [96, 97], or sparsity in fixed transforms [32–34]. Noise is reduced by various types of coefficient shrinkage, e.g. sparse coding of the compressed representation [34, 97]. More recently, data-driven approaches demonstrated promising results in image sparse modeling, including dictionary learning and transform learning, and thus lead to better denoising performance compared to those using analytical transforms [5,36,37,39]. Beyond these local structures, images also contain non-local structures, such as non-local self-similarity. Recent works proposed to group similar image patches, and denoise each group explicitly by applying collaborative filtering [32,46], group-based sparsity [37, 49, 52, 98], joint sparsity [47], low-rankness [48, 50, 51, 99, 100], etc. Yin et al. [101] proposed to use the row and column spaces of the stacked patch matrix to capture the local and non-local properties of the image, respectively, representing them by "convolution framelets" that capture both properties simultaneously. This formulation was used to interpret and improve upon the low dimensional manifold model (LDMM) [102]. However, in this formulation, the local structure is represented in a linear way (not by sparsity). This is different from our proposed approach, in which sparsity and low-rankness are simultaneously imposed in the image model. Table 5.1 summarizes the key attributes of some of the aforementioned related image denoising algorithm representatives, as well as the proposed STROLLR method.

In addition to exploiting image internal structures, *external* methods learn the image model using a corpus of clean training images. The well-known fields of experts (FoE) method [103], and the EPLL algorithm [104] proposed to restore an image using a probabilistic model for image patches, which is learned on a corpus of clean image patches. The PGPD [105] and PCLR [94] algorithms construct Gaussian mixture models (GMM) using patch groups from a training corpus, with additional sparsity and low-rank regularizers, respectively, which achieved improved denoising results. More recently, deep neural networks (DNN) have demonstrated remarkable potential to learn image models from training dataset with an end-to-end approach [54, 106–108]. The shrinkage field (SF) [107] and trainable nonlinear reaction diffusion (TNRD) [108] networks unrolled iterative denoising algorithms that are based on analysis sparse models. Besides networks derived by unrolling an iterative algorithm for a variational formulation, other popular neural networks structures, such as fully connected networks (FCN), convolutional neural networks (CNN), recurrent neural networks (RNN), and U-Net, have been applied to image restoration with state-of-the-art results [54, 106, 109, 110].

Although *external* methods often demonstrate superior denoising performance, they are supervised algorithms that require training on a corpus of images with distribution similar to the images to be denoised. It is expensive, or sometimes impossible, to obtain a reliable training set of this kind in applications such as remote sensing, biomedical imaging, scientific discovery, etc. In this work, we restrict our attention to *internal* image denoising algorithms, and leave the combination with external methods to future work.

5.1.2 Image Inpainting

The term *image inpainting* [111] refers to the process of recovering the missing pixels in an image. The inpainting problem is encountered in many image applications, including image restoration, editing (e.g., object removal), tex-

Table 5.1: Comparison between internal image denoising methods, including ODCT, KSVD, OCTOBOS, NLM, BM3D, GSR , SAIST, and STROLLR (this work).

Methods	Sparse Model		Collab.	Joint	Low-
	Fixed	Learned	Filtering	Sparse	Rank
ODCT	✓				
KSVD		✓			
OCTOBOS		✓			
NLM			✓		
BM3D	✓		✓		
GSR		✓		✓	
SAIST					✓
STROLLR		✓			✓

ture synthesis, content-aware image resizing (e.g., image enlargement), etc. In this chapter, we restrict to inpainting problems in image recovery application, in which the missing region is generally small (e.g., random pixels missing), and the goal of inpainting is to estimate the underlying complete image.

Similar to denoising, successful image inpainting algorithms exploit sparsity or non-local image structures. Popular inpainting methods exploit the structure in a local neighborhood of the missing pixels. Bertalmio et al. pioneered the work based on partial differential equations (PDEs) to propagate image local structures from known region to missing pixels [111]. In addition, classical inpainting algorithms also applied TV as image regularizers [112, 113]. Sparse priors have also been applied in inpainting problems, assuming the unknown and known parts of the image share the same sparse model [114]. Recent works proposed image inpainting methods based on patch sparsity, using dictionary learning [115, 116] and transform learning [39], demonstrating promising performance. On the other hand, *non-local* methods group similar image components and exploit their correlation. Ram

Table 5.2: Comparison between various MRI reconstruction methods, including SparseMRI, PBDWS, DLMRI, TLMRI, FRIST-MRI, PANO and STROLLR-MRI (this work).

Methods	Sparse Model			Non-	Super-
	Fixed	Direct.	Learned	Local	vised
Sparse MRI	✓				
PBDWS	✓	✓			
DLMRI			✓		
TLMRI			✓		
FRIST-MRI		✓	✓		
PANO	✓			✓	
ADMM-Net			✓		✓
STROLLR-MRI			✓	✓	

et al. [81] proposed to order image patches in a shortest path followed by collaborative filtering for inpainting. Li [117] proposed to iteratively cluster similar patches and reconstruct each cluster via sparse approximation. Jin and Ye [118] proposed inpainting algorithm using low-rank Hankel structured matrix completion. More recently, non-local algorithms [49, 119] applied dictionary learning within each group of similar patches, for improved sparse representation in inpainting problems. We refer the readers to a comprehensive review of various recent image inpainting approaches [120].

5.1.3 Compressed Sensing MRI

In modern imaging applications, the image recovery problem from the sparsely sampled measurements is often ill-posed. A popular approach to recover high-quality images is to use regularizers based on image priors that penalize the undesired solutions [121]. In this chapter, we focus on one popular example of an ill-posed imaging problem, compressed sensing (CS) MRI. CS techniques enable accurate MRI reconstruction from undersampled k-space (i.e., Fourier domain) measurements, by utilizing image sparsity. Popular CS MRI methods exploit either sparsity, or non-local self-similarity of the image. Here we survey several popular algorithms that are related to our proposed STROLLR-MRI. Comprehensive reviews can be found in [121–123].

To exploit image sparsity, Lustig et al. [72] proposed the Sparse MRI method, which uses wavelets and total variation regularization. Compared to such analytical transforms, adaptively learned transforms or dictionaries have proved to be more effective for image modeling [5, 44, 114]. Ravishankar and Bresler [44, 73] utilized dictionary learning (DL) and transform learning (TL) for MR image reconstruction achieving superior results. In other work, the PBDWS algorithm [82] used partially adaptive wavelets to form an MR image regularizer that exploited the patch-based geometric directions. More recently, the FRIST-MRI method [43] proposed to learn a sparsifying transform that is invariant to image patch orientations. On the other hand, non-local methods exploit the image self-similarity for high-quality MRI reconstruction. The PANO algorithm [83] used BM to group similar image patches, and applied the 3D Haar wavelet transform to model each group. Furthermore, Yoon et al. [51] proposed to approximate group-matched patches as low-rank. More recently, Yang et al. proposed ADMM-Net [124] to unroll the well-known alternating direction method of multipliers (ADMM) algorithm [125] applied to a standard variational formulation with p-norm sparsity regularization, into a feed forward neural network. ADMM-Net uses end-to-end training of linear operators that were fixed in the original variational formulation and ADMM algorithm. ADMM-Net achieved state-of-the-art performance in CS MRI reconstruction. This approach requires supervised training, in which the training corpus and the sampling patterns need to have distributions similar to those of the latent MRI measurements to be reconstructed. Table 5.2 summarizes the major attributes of the aforementioned CS MRI algorithms, as well as our proposed method.

5.2 STROLLR Model and Image Recovery

We propose a general image recovery framework based on the STROLLR model for image regularization. The goal is to recover an image (in vectorized form) $x \in \mathbb{C}^p$ from its degraded measurement $y \in \mathbb{C}^q$ using the classical variational formulation

$$\text{(P1)} \quad \hat{x} = \underset{x}{\operatorname{argmin}}\, \gamma^F \left\| Ax - y \right\|_2^2 + \mathfrak{R}_{strollr}(x)\,,$$

where $\gamma^F \left\| Ax - y \right\|_2^2$ is the image fidelity term with y being the measurement under the sensing operator $A \in \mathbb{C}^{q \times p}$, and γ^F being its weight. The structure of A varies in different image restoration problems. Here $\mathfrak{R}_{strollr}(x)$ is the STOLLR regularizer which jointly imposes sparsity in a certain transform domain and group low-rankness of the data. The proposed $\mathfrak{R}_{strollr}(x)$ is a weighted combination of non-local group low-rankness and sparsity penalties as follows:

$$\mathfrak{R}_{strollr}(x,\, \Theta) = \gamma^{LR}\, \mathfrak{R}_{LR}(x) \,+\, \gamma^S\, \mathfrak{R}_S(x,\, \Theta)\,, \tag{5.1}$$

where γ^{LR} and γ^S are the corresponding weights, and only the sparsity regularizer involves trainable parameters.

The term $\mathfrak{R}_{LR}(x)$ of the STROLLR regularizer imposes a low-rank prior on groups of similar patches via a matrix rank penalty,

$$\mathfrak{R}_{LR}(x) = \min_{\{D_i\}} \sum_{i=1}^{N} \left\{ \left\| V_i\, x - D_i \right\|_F^2 + \theta^2 \operatorname{rank}(D_i) \right\}\,, \tag{5.2}$$

where $V_i : x \mapsto V_i x \in \mathbb{C}^{n \times M}$ is a block matching (BM) operator. It takes $R_i\, x$ to be the reference patch, where $R_i \in \mathbb{C}^{n \times p}$ extracts the i-th n-pixel overlapping patch of x. The means of all overlapping patches are removed, and V_i selects M patches $\{u_j\}^i$ that are closest to $R_i x$ in Euclidean distance $\left\| u_j - R_i\, x \right\|_2$. There are N patches in total extracted from the image x. The selected patches $\{u_j\}^i$ are inserted into the columns of matrix $V_i x$ in ascending order of their Euclidean distance to $R_i\, x$. The removed means are added back once the patches are denoised via low-rank approximation. Computing the Euclidean distance between each patch pair, and sorting them, can be very expensive for large images. In practice, we set a square $\sqrt{Q} \times \sqrt{Q}$ pixel search window, which is centered at the reference patch. Only the overlapping patches within the search window are evaluated by the BM operator, assuming the neighborhood patches usually have higher spatial similarities. The optimal $\hat{D}_i$ is called the low-rank approximation of the matched block $V_i x$. The low-rank prior has been widely used to model spatially similar patch groups [48, 51, 99, 126]. Applying rank penalty leads to a simple low-rank approximation algorithm, which can be computed using singular value decomposition (SVD) and hard thresholding (see Section 5.3 for details).

The sparsity regularizer $\mathfrak{R}_S(x)$ assumes that a vectorized signal $u_i \in \mathbb{C}^n$ is approximately sparsifiable by some transform W that is adapted to the data x. One way to construct the sparsifiable signals is by using the vectorized 2D image patches [39], i.e., $u_i \triangleq R_i x$. Therefore, for a given transform $W \in \mathbb{C}^{m\times n}$, the sparsity regularizer on 2D image patches is formulated as

$$\mathfrak{R}_S^{2D}(x, W) = \min_{\{\alpha_i\}} \sum_{i=1}^{N} \left\{ \|W R_i x - \alpha_i\|_2^2 + \lambda^2 \|\alpha_i\|_0 \right\} , \tag{5.3}$$

where the ℓ_0 "norm" counts the number of nonzeros in each sparse vector α_i. Given the transform W, the optimal $\hat{\alpha}_i$ is called the sparse code of u_i, which can be calculated easily by hard thresholding (see Section 5.3).

We further extend the sparsity regularizer to impose sparsity over 3D patches. Instead of using $u_i \triangleq R_i x$, we construct the signals as $u_i \triangleq C_i x \in \mathbb{C}^{nl}$. The operator C_i first maps the BM matrix $V_i x$ (with first column $R_i x$) to the sub-matrix formed by its first l columns, and then vectorizes the sub-matrix (in column lexicographical order). Therefore, for a given transform $W \in \mathbb{C}^{m\times nl}$, the new sparsity regularizer $\mathfrak{R}_S(x, W)$ is formulated as

$$\mathfrak{R}_S(x, W) = \min_{\{\alpha_i\}} \sum_{i=1}^{N} \left\{ \|W\, C_i x - \alpha_i\|_2^2 + \lambda^2 \|\alpha_i\|_0 \right\} , \tag{5.4}$$

where each sparse code $\alpha_i \in \mathbb{C}^m$. Instead of using analytical transforms, an adaptively learned W [5, 38] provides superior sparsity, which serves as a better regularizer [6,27,44,45,127]. In the sparsity regularizer, the sparsifying transform is trainable, which is obtained by transform learning. Generally, the sparsifying transform W can be overcomplete [5] or square [38], with different types of regularizers or constraints [38]. In this work, we restrict ourselves to learning a square (i.e., $m = nl$) and unitary transform (i.e., $W^H W = I_{nl}$, where $I_{nl} \in \mathbb{C}^{nl\times nl}$ is the identity matrix) [38]. The sparsity regularization term in (5.1) is thus obtained as

$$\mathfrak{R}_S(x) = \min_{W \in \mathbb{C}^{nl \times nl}} \mathfrak{R}_S(x, W) \;\; s.t.\; W^H W = I_{nl} \,. \tag{5.5}$$

This optimization problem has a closed form solution requiring only the computation of the SVD of an $nl \times nl$ matrix, leading to highly efficient learning and image restoration algorithms [27, 42, 127].

In order to recover the underlying image x, we use the STROLLR regularizer for image recovery. We combine (P1) with (5.1), (5.2), (5.4) and (5.5), and pull the minimizations to the front. Therefore, the STROLLR learning based image recovery problem is formulated as follows:

$$\begin{aligned}
\text{(P2)} \quad \min_{\{x, W, \{\alpha_i, D_i\}\}} \quad & \gamma^F \left\| Ax - y \right\|_2^2 \\
& + \gamma^S \sum_{i=1}^{N} \left\{ \left\| W\, C_i x - \alpha_i \right\|_2^2 + \lambda^2 \left\| \alpha_i \right\|_0 \right\} \\
& + \gamma^{LR} \sum_{i=1}^{N} \left\{ \left\| V_i\, x - D_i \right\|_F^2 + \theta^2 \operatorname{rank}(D_i) \right\} \\
& \quad s.t. \;\; W^H W = I_{nl} \;.
\end{aligned}$$

5.3 Algorithm

We propose a simple block coordinate descent algorithm framework to solve (P2). The framework for the algorithm is given in Figure 5.1. Each iteration involves four steps: (*i*) low-rank approximation, (*ii*) sparse coding, (*iii*) transform update, and (*iv*) image reconstruction. For all applications under the general STROLLR image reconstruction framework (5.1), they follow the same STROLLR learning steps (*i*) - (*iii*). The image initialization, and the image reconstruction step (*iv*) may vary in specific applications with different sensing operator A's.

```
Algorithm 4.2: STROLLR-based Image Reconstruction
Input: The measurement $y$.
Initialize: $\hat{W}_0 = W_0$ (e.g., 2D DCT), and the image $\hat{x}_0$: For
$t = 1, 2, ..., T$ Repeat
    1. Low-rank Approximation for all $i = 1, ...N$:
        (a) Form $\{V_i \, \hat{x}_{t-1}\}$ using BM.
        (b) Compute the full SVD $\Gamma \, \mathrm{diag}(\omega) \, \Upsilon^H \leftarrow V_i \, \hat{x}_{t-1}$.
        (c) Update $\hat{D}_i = \Gamma \, \mathrm{diag}(H_\theta(\omega)) \, \Upsilon^H$.
    2. Sparse Coding: $\hat{\alpha}_i = H_\lambda(\hat{W}_{t-1} \, R_i \, \hat{x}_{t-1})$.
    3. Transform Update: Compute the full SVD $S \, \Sigma \, G^H \leftarrow$
       $\mathrm{SVD}(\sum_{i=0}^{N} (R_i \, \hat{x}_{t-1}) \hat{\alpha}_i)$, then update $\hat{W}_t = G \, S^H$.
    4. Image Reconstruction: Update $\hat{x}_t$ by solving the problem
       (5.9), with the specific $A$.
End
Output: The reconstructed image $\hat{x}_T$.
```

Figure 5.1: The STROLLR image recovery algorithm framework.

Low-rank Approximation

For fixed x, Problem (P2) separates into subproblems that we solve for each low-rank approximant D_i as:

$$\hat{D}_i = \underset{D_i}{\operatorname{argmin}} \; \|V_i\, x - D_i\|_F^2 + \theta^2 \operatorname{rank}(D_i)\,. \tag{5.6}$$

We form matrix $V_i\, x \in \mathbb{C}^{n\times M}$ using BM within the $\sqrt{Q}\times\sqrt{Q}$ search window, which is centered at the i-th patch u_i. Note that the locations (i.e., indices) of the patches used to form $V_i x$ and $C_i x$ in each iteration of the algorithm are updated and stored, to be used for the image reconstruction step.

Let $\Gamma\operatorname{diag}(\omega)\,\Upsilon^H = V_i\, x$ be the full SVD, where the diagonal vector ω contains the singular values. Then the low-rank approximation $\hat{D}_i = \Gamma\operatorname{diag}(H_\theta(\omega))\,\Upsilon^H$ is the exact solution. Here the hard thresholding operator $H_v(\cdot)$ is defined as

$$\left(H_v(\beta)\right)_r = \begin{cases} 0 & , \; |\beta_r| < v \\ \beta_j & , \; |\beta_r| \geq v \end{cases},$$

where $\beta \in \mathbb{C}^n$ is the input vector, v is the threshold value, and the subscript r indexes the vector entries.

Sparse Coding

Given the initialization, or the update of image x and transform W, we solve Problem (P2) for the sparse codes,

$$\hat{\alpha}_i = \underset{\alpha_i}{\operatorname{argmin}} \; \|W\, C_i\, x - \alpha_i\|_2^2 + \lambda^2\, \|\alpha_i\|_0 \quad \forall i\,, \tag{5.7}$$

which is the standard transform-model sparse coding problem. The optimal $\hat{\alpha}_i$ can be obtained using cheap hard thresholding, $\hat{\alpha}_i = H_\lambda(W\, C_i\, x)$.

Transform Update

For fixed x and $\{\alpha_i\}$, we solve for unitary W in (P2), which is equivalent to the following:

$$\hat{W} = \underset{W}{\operatorname{argmin}} \sum_{i=1}^{N} \|W\, C_i x - \alpha_i\|_2^2 \quad s.t.\; W^H\, W = I_n\,. \tag{5.8}$$

With the unitary constraint, the optimal $\hat{W}$ has a simple and exact solution [38]: denoting the full singular value decomposition (SVD) of $K \triangleq \sum_{i=1}^{N}(C_i\, x)\, \alpha_i^H$ as $S\, \Sigma\, G^H$, the transform update is $\hat{W} = G\, S^H$.

Image Reconstruction

With updated W, $\{D_i\}$, and $\{\alpha_i\}$, we reconstruct the underlying image x by solving the following problem:

$$\begin{aligned} \hat{x} \;=\; & \underset{x}{\operatorname{argmin}}\, \gamma^F\, \|A\, x - y\|_2^2 \\ & + \sum_{i=1}^{N} \left\{ \gamma^S\, \|C_i x - \hat{u}_i\|_2^2 + \gamma^{LR} \sum_{i=1}^{N} \|V_i\, x - D_i\|_F^2 \right\}\,. \end{aligned} \tag{5.9}$$

Here $\hat{u}_i \triangleq W^H \hat{\alpha}_i$ denotes the reconstructed patches via the transform-model sparse approximation. Since the unitary W preserves the norm, we have $\|C_i x - \hat{u}_i\|_2^2 = \left\|C_i x - W^H \alpha_i\right\|_2^2 = \|W C_i x - \alpha_i\|_2^2$.

The image reconstruction problem (5.9) is a least squares problem with solution given by the solution to the normal equation

$$B\, \hat{x} \;=\; z\,, \tag{5.10}$$

where the left and right sides of (5.10) are defined as

$$B \triangleq A^H A + \gamma^S \sum_{i=1}^{N} C_i^* C_i + \gamma^{LR} \sum_{i=1}^{N} V_i^* V_i \tag{5.11}$$

$$z \triangleq y + \gamma^S \sum_{i=1}^{N} C_i^* \hat{u}_i + \gamma^{LR} \sum_{i=1}^{N} V_i^* D_i\,. \tag{5.12}$$

Here $V_i^* : \mathbb{C}^{n \times M} \to \mathbb{C}^p$ and $C_i^* : \mathbb{C}^{nl} \to \mathbb{C}^p$ denote the adjoint operators of V_i and C_i, respectively, which correspond to patch deposit operators. In particular, V_i^* takes an $n \times M$ matrix of M patches, and "deposits" the patches in their respective locations in a (vectorized) image. Overlapping patches are added up where they overlap. A similar operation is performed by C_i^* on a length-nl vector, extracting l length-n consecutive subvectors and depositing them as patches in their respective locations in a (vectorized) image. In (5.11), both $\sum_{i=1}^{N} C_i^* C_i$ and $\sum_{i=1}^{N} V_i^* V_i$ are $p \times p$ diagonal matrices with (j,j) elements equal to the total number of the patches in all the $C_i x$'s and $V_i x$'s that contain the j-th pixel, respectively. In (5.12), the image-size vector $z \in \mathbb{C}^p$ is a weighted combination of noisy measurements and the images formed by the sparse and low-rank approximations of patches.

There are different sensing operators A associated with various inverse problems, leading to different forms of $B \in \mathbb{C}^{p \times p}$, and to variations in the solutions to step (iv). Direct inversion of B is typically expensive, but there exist efficient inverses of B for some inverse problems. Several exemplary applications will be discussed in Section 5.5. The general image recovery algorithm using STROLLR learning is summarized as Algorithm 4.2.

Computational Cost

In Algorithm 4.2, the computational cost for the STROLLR-based image recovery, excluding the image reconstruction step, is $O(NnQ + \min(NMn^2, NnM^2) + Nn^2l^2 + (Nn^2l^2 + n^3l^3))$ per iteration, corresponding to the steps of BM, low-rank approximation, sparse coding, and transform update, respectively, where the Nn^2l^2 term in the transform update step is the cost of forming matrix K, and the n^3l^3 corresponds to the cost of its SVD. Here the number of patches N scales similar to the image size $p \gg n$. Typically, the search window size needs to be sufficiently large, i.e., $Q \gg M, n$. The 3D patches are only formed by a small number of l highly correlated 2D patches, i.e., $l^2 < M$. Furthermore, the BM matrix size $M \gg 1$ and scales similar to n. Thus, the cost of the STROLLR-based image recovery is dominated by the cost of BM and low-rank approximation steps, and scales as $O(NnQ + \min(NMn^2, NnM^2))$, i.e., as $O\left(n(Q + M\min(n, M))\right)$ per image pixel per iteration.

Note that this cost analysis is based on full SVD, and naive BM algorithm. Further cost reductions are possible by randomized SVD to obtain an approximate truncated SVD [128, 129], and by using fast data structures and algorithms for k-NN (with k=M nearest neighbors) to perform the block matching [130].

5.4 Image Recovery Applications

The STROLLR model is particularly appealing in recovery of natural images, as well as biomedical images. In this section, we consider three such applications, namely image denoising, inpainting, and CS-based MRI. Each corresponds to a specific sensing operator A in (P2), thus leading to a different image reconstruction step in (5.9).

5.4.1 Image Denoising

When $A = I_p$, we are solving the image denoising problem, which is one of the most fundamental inverse problems in image processing. The goal is to recover the image x from its noisy measurement $y = x + e$, which is corrupted by noise vector e. In the image denoising algorithm based on STROLLR learning, we initialize $\hat{x}_0 = y$. The matrix B in (5.11) thus becomes

$$B = \operatorname{diag}(b) \triangleq I_p + \gamma^S \sum_{i=1}^{N} C_i^* C_i + \gamma^{LR} \sum_{i=1}^{N} V_i^* V_i \,, \tag{5.13}$$

where $B \in \mathbb{C}^{p \times p}$ is a diagonal matrix with positive diagonal elements $b^j > 0 \; \forall j$. Thus, with z given by (5.12), the denoised image has the closed form:

$$\hat{x} = B^{-1} z \,. \tag{5.14}$$

The inversion of the diagonal matrix B is simple, and the solution reduces to the pixel-wise division $\hat{x}_j = z_j / b^j \; \forall j = 1, ..., p$, which can be thought of as normalization to remove redundancy in z. The computational cost of this step is $O(p)$ per iteration, which is negligible compared to the cost of the other steps $(i) - (iii)$. Thus, the computational cost of the STROLLR based image denoising algorithm is on par with various state-of-the-art methods such as BM3D [32], SAIST [48], etc.

Though the proposed algorithm is designed to recover gray-scale images, it has a simple extension to color image denoising that exploits the correlation across the color channels. There are algorithm modifications in both transform learning and low-rank approximation. The red (R), green (G), and blue (B) channels[1] of each color patch are vectorized to form one training sample in transform learning. In the low-rank approximation step, the BM operator V_i selects the M patches that have the minimum Euclidean distance to $R_i x$, summed over the three color channels. The block matched matrix $V_i x$ is formed by matched patches, in which the R, G, and B channels of each selected patch are in the adjacent columns.

5.4.2 Image Inpainting

The goal of image inpainting is to estimate the missing pixel in an image. When $A = \Phi \in \mathbb{C}^{p \times p}$, the given image measurement is denoted as $y = \Phi x + e$, where the Φ is a diagonal binary matrix with zeros at the locations corresponding to missing pixels in y. The vector e denotes the additive noise on the available pixels. Similar to image denoising, we initialize $\hat{x}_0 = y$ in the STROLLR-based inpainting algorithm.

Similar to denoising, the inpainted image has a simple closed-form solution $x = B^{-1} z$, where z is again given by (5.12) and the normalization matrix is the diagonal matrix with positive diagonal elements:

$$B \triangleq \Phi + \gamma^S \sum_{i=1}^{N} C_i^* C_i + \gamma^{LR} \sum_{i=1}^{N} V_i^* V_i \,. \tag{5.15}$$

The matrix inversion is again cheap pixel-wise division.

In the ideal case when the noise e is absent, i.e., $\sigma = 0$, we replace the fidelity term $\|\Phi\, x - y\|_2^2$ with the hard constraint $\Phi\, x \;=\; y$. The image reconstruction step becomes a constrained optimization problem,

$$\min_{x} \sum_{i=1}^{N} \left\{ \gamma^S \, \|C_i x - \hat{u}_i\|_2^2 + \gamma^{LR} \textstyle\sum_{i=1}^{N} \|V_i \, x - D_i\|_F^2 \right\} \quad s.t. \;\; \Phi\, x \;=\; y \,. \tag{5.16}$$

[1]The proposed denoising algorithm can be also applied to data in a different color space possibly weighting more a highly informative channel such as luminance.

Let Ω denote the subset of image pixels that are sampled by y, i.e., $\text{diag}(A)_\Omega \neq 0$. With the hard constraint on the pixels in Ω, the closed-form solution $\hat{x}$ to (5.16) becomes

$$\hat{x}_j = \begin{cases} z_j/b_j & , \ j \notin \Omega \\ y_j & , \ j \in \Omega \end{cases} \tag{5.17}$$

where the vectors z and b are defined as

$$\text{diag}(b) \triangleq \gamma^S \sum_{i=1}^{N} C_i^* C_i + \gamma^{LR} \sum_{i=1}^{N} V_i^* V_i \tag{5.18}$$

$$z \triangleq \gamma^S \sum_{i=1}^{N} C_i^* \hat{u}_i + \gamma^{LR} \sum_{i=1}^{N} V_i^* D_i. \tag{5.19}$$

Similar to image denoising, the computational costs of the image reconstruction step in inpainting is also $O(p)$, i.e., $O(1)$ per pixel, per iteration, which is negligible relative to the costs of the other steps. The computational costs of both the noisy and ideal STROLLR based image inpainting algorithms are on par with popular competing methods, such as GSR [49].

5.4.3 Magnetic Resonance Imaging

We propose an MR image reconstruction scheme based on STROLLR learning, dubbed STROLLR-MRI. The sensing operator $A = F_{\boldsymbol{g}} \in \mathbb{C}^{q \times p}$ in (5.9) is the undersampled Fourier encoding matrix, composed of the q rows of the unitary $p \times p$ 2D DFT matrix F corresponding to the sampled locations in k space. The selected rows are indicated by the positions of ones in the binary vector $\boldsymbol{g} \in \{0,1\}^p$. The k-space measurement $y \in \mathbb{C}^q$ has lower dimension, i.e., $q \ll p$; thus, the MRI reconstruction is an ill-posed problem that requires an effective image regularizer. We can directly formulate STROLLR-MRI based on (P2), but this introduces several complications. First, as the number of pixel references generated by BM is not homogeneous across the image, the B matrix in (5.11) cannot be diagonalized by the DFT [43,44,73], making direct inversion of B impractical for MRI, requiring that the image update step (9) be performed by by iterative optimization methods, such as conjugate gradients at considerably higher computation cost. Second, as image content and therefore the matches by BM are updated from one iteration to the next, the structure of the cost function will usually vary and affect the algorithm convergence [51].

Instead, we propose to normalize the weights of patches that appear in multiple BM groups $V_i\,x$ or $C_i\,x$ by the number of their appearances, so that all patches exert similar influence on the regularizer. We initialize the image by the so-called zero-filled DFT inverse, $\hat{x}_0 = F_{\boldsymbol{g}}^H y$ in the STROLLR-based MRI algorithm. The STROLLR-MRI image reconstruction step (5.9) is replaced by

$$\begin{aligned}\hat{x} \; &= \underset{x}{\operatorname{argmin}} \; \|F_u\,x - y\|_2^2 \\ &+\gamma^S \sum_{i=1}^{N}\sum_{j=1}^{l} \frac{1}{P_{i,j}} \,\|(C_i\,x)_j - \hat{u}_{i,j}\|_2^2 \\ &+\gamma^{LR} \sum_{i=1}^{N}\sum_{j=1}^{M} \frac{1}{L_{i,j}} \,\|\,(\,V_i\,x\,)_j - D_{i,j}\,\|_2^2 \quad . \end{aligned} \tag{5.20}$$

Here $(\,V_i\,x\,)_j$ and $D_{i,j}$ denote the j-th column of the matrices $V_i\,x$ and D_i, respectively. The weight $L_{i,j}$ equals the number of times that the j-th column of $V_i\,x$ appears in all $\Big\{V_i\,x\Big\}_{i=1}^{N}$. Similarly, we use $(C_ix)_j \in \mathbb{C}^n$ and $\hat{u}_{i,j} \in \mathbb{C}^n$ to denote the j-th block of C_ix and $\hat{u}_i \in \mathbb{C}^n$, respectively. The weight $P_{i,j}$ equals the number of times that the j-th block of $C_i\,x$ appears in all $\Big\{C_i\,x\Big\}$. Define the sets $\Delta_k = \Big\{(i,j) \mid (\,C_i\,x\,)_j = R_k\,x\Big\}$, and $\Gamma_k = \Big\{(i,j) \mid (\,V_i\,x\,)_j = R_k\,x\Big\}$, which indicate the indices (i,j) where patch $R_k\,x$ appears in $C_i\,x$ and $V_i\,x$, respectively. As all wrap-around patches are used as reference patches, each R_kx appears at least once in $\Big\{V_ix\Big\}$ and $\Big\{C_ix\Big\}$, i.e., $\Big|\Delta_k\Big| \geq 1$ and $\Big|\Gamma_k\Big| \geq 1$, respectively. Thus, each $R_k\,x$ can be represented as

$$R_k\,x \; = \frac{1}{\Big|\Delta_k\Big|} \sum_{(i,j)\in\Delta_k} (C_i\,x)_j = \frac{1}{\Big|\Gamma_k\Big|} \sum_{(i,j)\in\Gamma_k} (V_i\,x)_j\,. \tag{5.21}$$

Using (5.21) and the fact that $P_{i,j} = \Big|\Delta_k\Big|$ for $(i,j) \in \Delta_k$, and $L_{i,j} = \Big|\Gamma_k\Big|$ for $(i,j) \in \Gamma_k$, defining $\tilde{u}_k \triangleq \frac{1}{\big|\Delta_k\big|} \sum_{(i,j)\in\Delta_k} \hat{u}_{i,j}$ and $d_k \triangleq \frac{1}{\big|\Gamma_k\big|} \sum_{(i,j)\in\Gamma_k} D_{i,j}$, and dropping terms independent of x, (20) simplifies to

$$
\begin{aligned}
\hat{x} \; = \underset{x}{\operatorname{argmin}} \; & \|F_u\, x - y\|_2^2 \; + \\
& \sum_{k=1}^{N} \left\{ \gamma^S \left\| \, R_k x - \tilde{u}_k \right\|_2^2 + \gamma^{LR} \left\| \, R_k x - d_k \right\|_2^2 \right\} .
\end{aligned} \tag{5.22}
$$

The normal equation of (5.22) for STROLLR-MRI in k-space is simplified to

$$
\begin{aligned}
& \left[F F_u^H F_u F^H + (\gamma^S + \gamma^{LR}) F \textstyle\sum_{i=1}^N R_i^* R_i F^H \right] \, F\hat{x} \\
& \qquad = F F_u^H y + F \sum_{i=1}^{N} R_i^* (\gamma^S \tilde{u}_i + \gamma^{LR} v_i) \, .
\end{aligned} \tag{5.23}
$$

On the left-hand side of (5.23), matrix $F F_u^H F_u F^H = \operatorname{diag} \boldsymbol{g}$ is diagonal with binary entries equal to one at the sampled locations in k space, and $F \sum_{i=1}^N R_i^* R_i F^H = nF\, I_p F^H = n\, I_p$ is a scaled identity. Therefore, a simple solution to (5.23) is

$$
\hat{x} = F^H B^{-1} \, z \, , \tag{5.24}
$$

where $B \triangleq \operatorname{diag} \boldsymbol{g} + n(\gamma^S + \gamma^{LR}) I_p$ is a diagonal matrix whose inversion is cheap, and $z \triangleq \boldsymbol{G} y + F \sum_{i=1}^N R_i^H (\gamma^S \tilde{u}_i + \gamma^{LR} v_i)$, where $\boldsymbol{G} = F F_g^H \in \mathbb{C}^{p \times q}$ is a binary "upsampling" matrix, which places the entries of y in their corresponding k-space locations indicated by g into a length-p vector.

5.5 Experiments

In this section, we demonstrate the promise of the STROLLR based image restoration framework by testing our gray-scale / color image denoising, image inpainting, and CS MRI algorithms on publicly available images or datasets [43, 44, 49, 75, 124, 131, 132]. To evaluate the performance of the image recovery algorithms, we measure the peak signal-to-noise ratio (PSNR) in decibel (dB), which is computed between the ground truth image and the recovered image.

All of the proposed STROLLR-based image recovery algorithms are unsupervised algorithms. There are several hyperparameters used in the algorithm, among which we set the spatial search window $\sqrt{Q} \times \sqrt{Q} = 30 \times 30$. We initialize the unitary sparsifying transform W_0 to be the 3D DCT (of size $\sqrt{n} \times \sqrt{n} \times l$). These settings are fixed in all experiments.

5.5.1 Image Denoising

We present image denoising results using our proposed algorithms in Sec.5.4.1. For gray-scale image denoising experiment, we first convert the images in the Kodak [131] and SIPI Misc [75] datasets to gray-scale, and simulate i.i.d. Gaussian noise at 5 different noise levels ($\sigma = 5, 10, 15, 20$ and 50). For the color image denoising experiment, we use all 24 color images from the *Kodak* dataset, and simulate equal-intensity i.i.d. Gaussian noise in each of the RGB channels at 4 different noise levels ($\sigma = 15, 25, 35$ and 50).

Implementation Details and Parameters

We set the regularizer weights $\gamma^S = \gamma^{LR} = 1$, and the fidelity weight $\gamma^F = 0.1/\sigma^2$, where σ is the noise standard deviation of the noisy image y. We directly use the noisy image as the initial estimate $\hat{x}_0$. Only for denoising, we modify the image update step (5.14), at iteration $t = 1, 2, ...T - 1$ (except for the last iteration), as follows:

$$\hat{x}_t = \delta B^{-1} z_t + (1 - \delta) y \,, \tag{5.25}$$

where we set $\delta = 0.1$. The modified $\hat{x}_t$ is the convex combination of the denoised estimate and the original noisy image. This method, widely known as iterative regularization for non-convex inverse problems [50,133], has been applied in various popular image restoration algorithms [50,105,134,135]. Let $\tilde{x}_t \triangleq \hat{x}_t - x$ denote the noise remaining in $\hat{x}_t$. We re-estimate the variance of $\tilde{x}_t$ using $\sigma_t^2 = \psi(\sigma^2 - (1/N)\|y - \hat{x}_t\|^2)$ [50, 135]. Here $(1/N)\|\hat{x}_t - y\|^2$ is an estimate of the variance of the noise removed throughout the t iterations. Assuming the removed noise to be white and uncorrelated with $\tilde{x}_t$, the estimated variance of $\tilde{x}_t$ is $\sigma^2 - (1/N)\|\hat{x}_t - y\|^2$. However, because in practice the removed and the remaining noises are positively correlated, σ_t^2 tends to be over-estimated using the ideal formula. To better approximate the actual σ_t^2, we compensate by the factor $\psi = 0.36$ [50,105,134,135]. At the tth iteration,

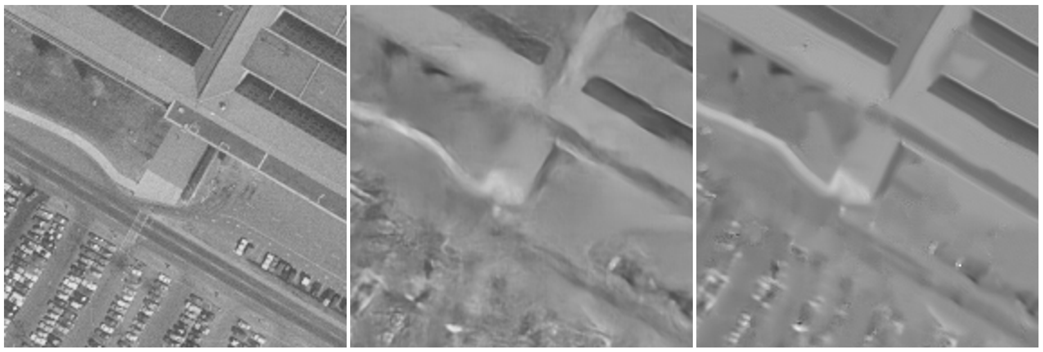


Figure 5.2: Denoising result of the image *Pentagon*: the zoom-in regions of (a) the ground truth, (b) the denoised image by DnCNN (PSNR = 25.87 dB), and (c) the denoised image by STROLLR (PSNR = 26.05 dB).

we set the penalty parameters $\lambda = 1.2\sigma_{t-1}$ and $\theta = 0.8\sigma_{t-1}(\sqrt{n}+\sqrt{M})$ [127], using the re-estimated σ_{t-1} (or the noise level of y, $\sigma_0 = \sigma$ at the first iteration). The remaining hyper parameters are the patch size n, data matrix sizes M and l, and the number of iterations T. We set them to be $\Big\{n, M, l, T\Big\} = \Big\{6^2, 70, 8, 8\Big\}$ and $\Big\{7^2, 80, 7, 10\Big\}$, for low-noise case (i.e., $0 \leq \sigma \leq 30$), and high-noise case (i.e. $\sigma > 30$), respectively.

Gray-Scale Image Denoising

We compare our proposed STROLLR based image denoising algorithm to various well-known alternatives, including denoising algorithms using over-complete DCT (ODCT) dictionary, KSVD [36], GHP [136], Shrinkage Fields (SF) [107], EPLL [104], NLM [46], OCTOBOS [5], BM3D [32], NCSR [137], PGPD [105], and SAIST [48]. We use their publicly available codes for implementation. Among these methods, ODCT, KSVD and OCTOBOS exploit sparsity of image patches. EPLL and GHP make use of image pixel statistics. SF uses an unrolled neural network based on an analysis sparse model. NLM, BM3D, SAIT, NCSR, and PGPD are all non-local methods that use collaborative filtering, low-rank, or sparse approximation. Additionally, to better understand the benefit of each of the regularizers used in STROLLR model, we evaluate the denoising results using only the transform learning (TL), and the low-rank approximation (LR).

Table 5.3 lists the denoised PSNRs obtained using the aforementioned

Table 5.3: Comparison of gray-scale image denoising PSNR values (in dB), averaged over the Kodak and USC-SIPI Misc datasets, using the proposed STROLLR image denoising method, versus other competing algorithms. For each dataset and noise level, the best denoising PSNR is marked in bold. For each method, Δ PSNR denotes the PSNR loss relative to the proposed STROLLR algorithm (highlighted in bold) averaged over the five different noise levels.

	Kodak Dataset (24 images)					ΔPSNR
σ	5	10	15	20	50	
SF	37.60	33.51	31.40	29.79	23.84	-1.53
NLM	36.85	32.91	30.93	29.62	25.55	-1.59
GHP	37.90	34.16	31.93	30.86	26.20	-0.55
ODCT	37.55	33.51	31.32	29.84	25.61	-1.20
KSVD	37.60	33.70	31.60	30.18	25.93	-0.96
EPLL	38.15	34.29	32.22	30.82	26.74	-0.32
OCTOBOS	38.27	34.24	32.16	30.70	26.48	-0.39
PGPD	38.21	34.37	32.32	30.92	27.08	-0.18
BM3D	38.30	34.39	32.30	30.92	26.98	-0.18
NCSR	38.35	34.48	32.36	30.92	26.84	-0.17
SAIST	38.39	34.51	32.39	30.98	26.95	-0.12
STROLLR	38.46	34.61	32.50	31.06	27.18	0.00

	USC-SIPI Misc Dataset (44 images)					ΔPSNR
σ	5	10	15	20	50	
SF	36.93	33.27	31.40	30.01	24.08	-2.66
NLM	37.14	33.38	31.52	30.29	26.12	-2.10
GHP	36.32	33.36	31.55	30.59	26.42	-2.14
ODCT	38.12	34.27	32.16	30.71	26.35	-1.47
KSVD	38.33	34.67	32.69	31.35	26.95	-1.00
EPLL	38.41	34.67	32.67	31.32	27.13	-0.95
OCTOBOS	38.91	35.06	33.12	31.70	27.35	-0.56
PGPD	38.28	34.94	33.09	31.86	27.88	-0.58
BM3D	39.04	35.31	33.36	32.05	27.85	-0.27
NCSR	39.13	35.38	33.39	32.03	27.90	-0.23
SAIST	39.13	35.39	33.39	32.06	27.87	-0.22
STROLLR	39.26	35.60	33.63	32.29	28.18	0.00

methods, with the best result for each noise level and testing dataset (i.e., each column) marked in bold. The proposed STROLLR image denoising

Table 5.4: PSNRs of gray-scale image denoising, using STROLLR and its variants, averaged over the Kodak image dataset. For each noise level, the best denoising PSNR is marked in bold. For each variant, ΔPSNR denotes the PSNR loss relative to the full STROLLR denoiser, averaged over the four noise levels.

	Kodak Dataset (24 images)				Δ
σ	5	10	20	50	PSNR
STROLLR-S w/o TL	38.15	34.09	30.33	25.82	-0.73
STROLLR-S w/o LR	38.09	34.02	30.36	25.96	-0.72
STROLLR-S	38.31	34.43	30.96	26.87	-0.19
STROLLR	38.46	34.61	31.06	27.18	0

method provides average PSNR improvements of 0.2dB, 0.2dB, 0.3dB, 0.4dB, 0.5dB, 0.7dB, 1.0dB, 1.4dB, 1.6dB, 2.0dB, and 2.2dB, respectively, over the SAIST, NCSR, BM3D, PGPD, OCTOBOS, EPLL, KSVD, ODCT, GHP, NLM and SF denoising methods. By imposing both sparsity and non-local (i.e., group low-rankness) regularizers, for all noise σ's and testing datasets, STROLLR performs consistently the best. Thus our proposed method demonstrates robust and promising performance in image denoising compared to popular competing methods.

To further analyze the effectiveness of imposing both the sparsity and the low-rankness regularizers, as well as applying the iterative regularization method, we conduct an "ablation" study by disabling in turn each of these three components in the STROLLR image denoising algorithm. We run STROLLR denoising with single pass, i.e., without iterative regularization, which is denoted as STROLLR-S. On top of STROLLR-S, we further disable the low-rank regularizer $\mathfrak{R}_{LR}$, and the sparsity regularizer $\mathfrak{R}_S$, which are referred as STROLLR-S w/o LR, and STROLLR-S w/o TL, respectively. Table 5.4 lists the denoised PSNRs obtained using STROLLR, and its variants. It is clear that STROLLR denoising outperforms its variants, and all of the three components contribute significantly to the success of the proposed STROLLR algorithm.

Besides the conventional approaches, popular deep learning techniques

[106–108, 138, 139] have been shown useful in image denoising. The recently proposed DnCNN [109] demonstrated image denoising results superior to those of the standard natural image datasets. The deep learning based methods typically require a large training corpus containing images that have similar distribution to the image to be recovered. However, such a training corpus may not always be easy to obtain in applications such as remote sensing, biomedical imaging, etc. Figure 5.2 shows an example denoising result of the image *Pentagon* from the SIPI aerial dataset [75]. Here we applied the DnCNN algorithm using the publicly available implementation and the trained models (using 400 images from BSDS500 dataset [103]) from the authors' project website. Comparing to the denoised result using the proposed STROLLR-based algorithm, DnCNN generates various artifacts and distortions in the denoised image.

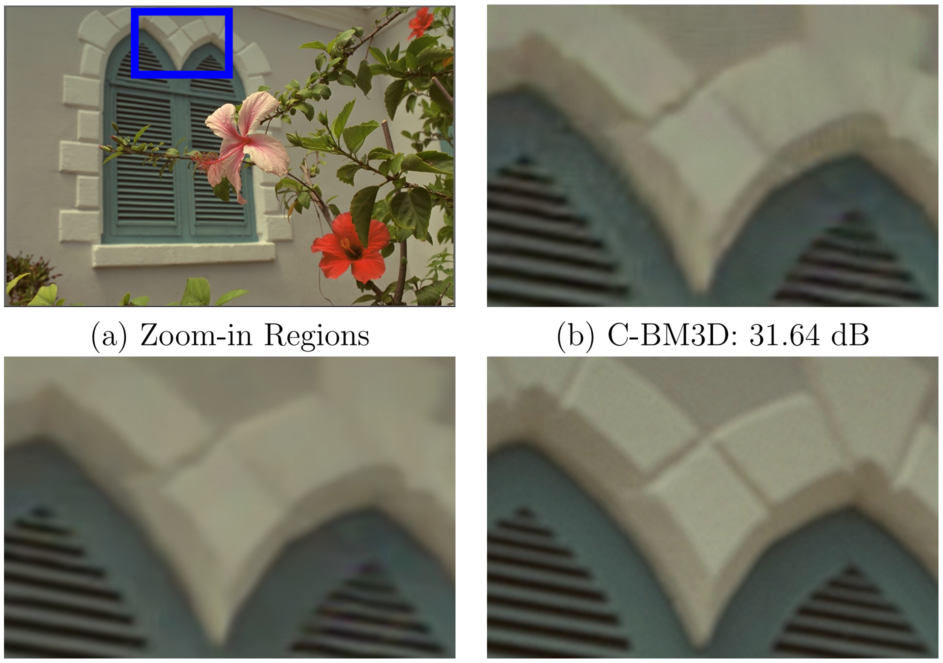
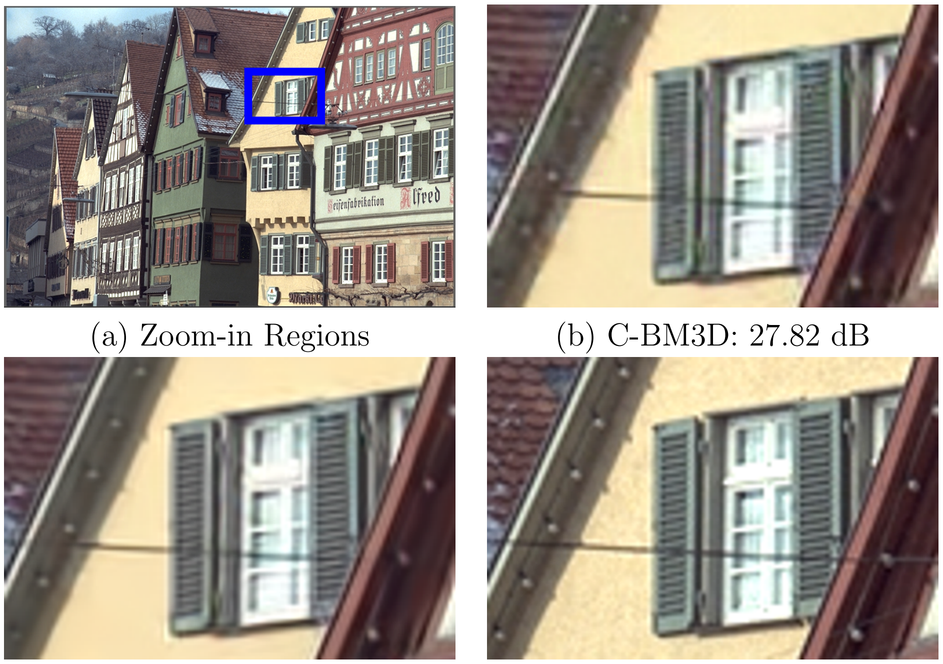

(a) Zoom-in Regions
(b) C-BM3D: 31.64 dB
(c) C-STROLLR: 32.08 dB
(d) Ground Truth

Figure 5.3: Denoising results of (a) the example color images *Kodim07* at $\sigma = 35$, with the blue rectangles highlighting the zoom-in regions of (b) the images denoised by C-BM3D (PSNR = 31.64 dB), and (c) the images denoised by C-STROLLR (PSNR = 32.08 dB), and (d) the ground truth.



(a) Zoom-in Regions

(b) C-BM3D: 27.82 dB

(c) C-STROLLR: 28.45 dB

(d) Ground Truth

Figure 5.4: Denoising results of (a) the example color images *Kodim08* at $\sigma = 35$, with the blue rectangles highlighting the zoom-in regions of (b) the images denoised by C-BM3D (PSNR = 27.82 dB), and (c) the images denoised by C-STROLLR (PSNR = 28.45 dB), and (d) the ground truth.

Color Image Denoising

The C-STROLLR algorithm extends STROLLR to color image denoising as described before. We compare to popular denoising methods including WNNM [50], TNRD [108], MC-WNNM [134], and C-BM3D [140]. Among the selected competing methods, TNRD is a deep learning based method, while WNNM, MC-WNNM, and C-BM3D are all internal methods using non-local image structures. WNNM is based on low-rank approximation, which is a gray-scale image denoising algorithm. Thus, we apply WNNM to each of the RGB channels of color images. MC-WNNM is the color image denoising extension of WNNM algorithm, which further exploits the cross-channel correlation.

Table 5.5 lists the average denoising PSNRs over the 24 color images from Kodak database, with the best result for each noise level marked in bold. It

Table 5.5: PSNR values of color image denoising, averaged over the Kodak color image dataset, using the TNRD, MC-WNNM, C-BM3D, and the proposed C-STROLLR image denoiser. For each noise level, the best denoising PSNR is marked in bold. For each method, Δ PSNR denotes the PSNR loss relative to the proposed STROLLR algorithm (highlighted in bold), averaged over the four noise levels.

	Kodak Dataset (24 color images)				ΔPSNR
σ	15	25	35	50	
WNNM	32.49	29.68	28.49	26.93	-1.98
TNRD	N/A	30.08	N/A	27.17	-1.76
MC-WNNM	33.94	31.35	29.70	28.02	-0.63
C-BM3D	34.41	31.81	30.04	28.62	-0.16
C-STROLLR	34.57	31.94	30.25	28.78	0

is clear that the proposed C-STROLLR performs consistently the best for all noise levels. The MC-WNNM algorithm generates significantly better results comparing to those using the channel-wise WNNM denoising, which demonstrates the importance of exploiting image color correlation in restoration. Figures 5.3 and 5.4 compare the denoising results of example images *Kodim07* and *Kodim08*, respectively, at $\sigma = 35$, using the best competitor C-BM3D, and the proposed C-STROLLR algorithms. The denoised image by C-STROLLR preserves clearer details thanks to the sparsity regularization, while C-BM3D generates undesired artifacts, e.g., the zoomed-in region in the blue boxes. We observe similar, or more severe, artifacts in the denoising results using other competing methods.

5.5.2 Image Inpainting

We present the image inpainting results using our STROLLR based algorithm. The testing gray-scale images *Barbara* and *House* which were used in the GSR inpainting method [49] are selected to evaluate our proposed STROLLR method, as well as competing methods including Bicubic interpolation, SKR [141], Smooth [81], and GSR [49]. We work with 6×6 patches,

Table 5.6: PSNR values of image inpainting, using bicubic interpolation, SKR, GSR, and the proposed STROLLR image inpainting method. For each image and available pixel percentage, the best inpainting PSNR is marked in bold.

Image	*Barbara*			*House*		
Available pixels	10%	20%	50%	10%	20%	50%
Bicubic	22.65	23.65	25.94	29.82	31.62	33.18
SKR	21.92	22.45	22.81	30.18	31.05	31.94
Smooth	28.32	30.90	35.94	33.67	36.62	39.98
GSR	31.32	34.42	39.12	35.61	37.65	41.61
STROLLR	31.51	34.56	39.33	35.72	37.75	41.70

and set $M = 80$, $l = 8$, the number of iterations $T = 150$, and $\gamma^S = \gamma^{LR} = 1$. We set the rank penalty weight to be $\theta = \lambda(\sqrt{n} + \sqrt{M})$. We randomly remove image pixels, and keep only 20%, 30% and 50% pixels of the entire image, and set the sparsity penalty weight $\lambda = 20, 12$ and 5 respectively.

Table 5.6 lists the inpainting PSNRs, over all available pixel percentages and testing images, obtained using the aforementioned methods. The best result for each testing case is marked in bold. The proposed STROLLR inpainting algorithm produces better results than the popular competitors.

5.5.3 MRI reconstruction

We present the MRI reconstruction results using the proposed STROLLR-MRI algorithm. The 4 testing MR images (3 anatomical and one physical phantom), i.e., the images **a** - **c** used in our experiments shown in Fig. 5.5(a)-(c) and the image **d** shown in Fig. 5.6, are all publicly available [43, 44, 132]. We simulated complex MR data obtained by taking the discrete Fourier transform (DFT) of the magnitude of the complex images,[2] with various un-

[2]The STROLLR-MRI, as well as the competing methods except for ADMM-Net [124], can handle complex MR data.

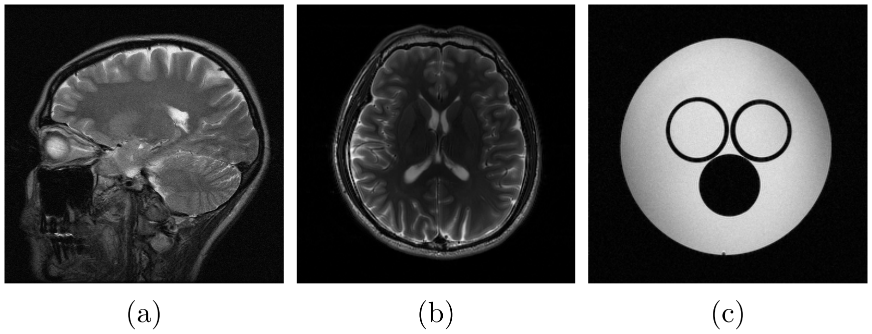
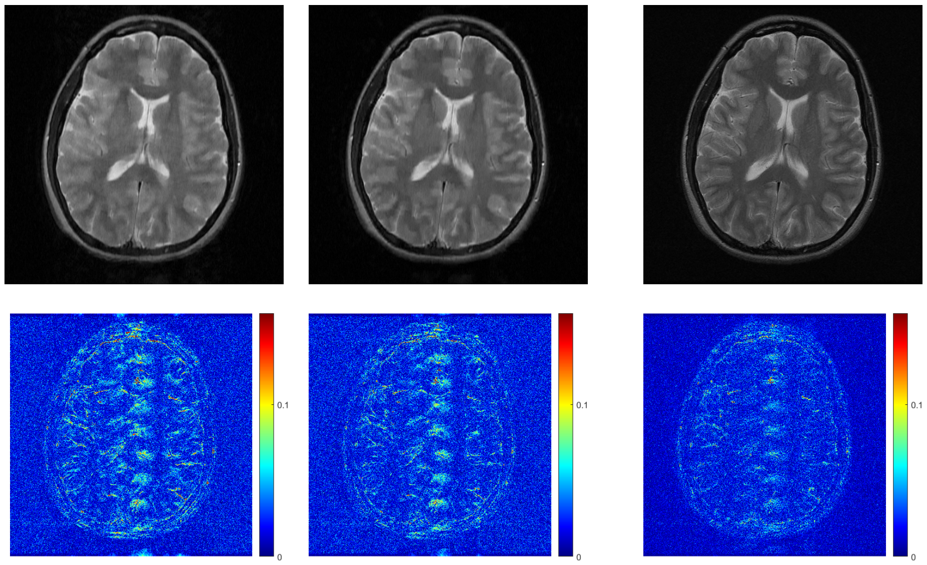


Figure 5.5: Testing anatomical and physical phantom MR images: (a) is used in TL-MRI and FRIST-MRI, and (b) and (c) are from a publicly available dataset.



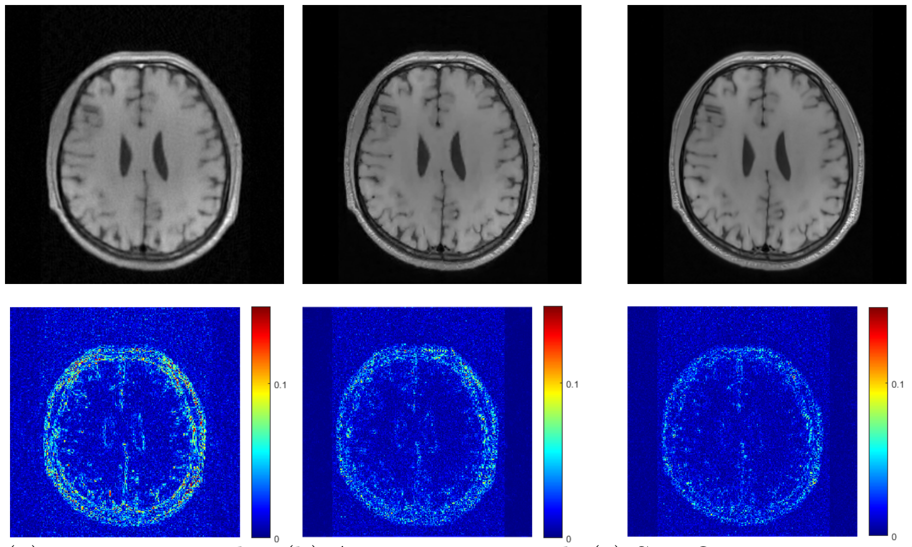
(a) DL-MRI, 31.7dB (b) FRIST-MRI, 32.1dB (c) STROLLR-MRI 33.3dB

Figure 5.6: Reconstruction of MR image **d** using (a) DL-MRI, (b) FRIST-MRI, and (c) the proposed STROLLR-MRI. Top row: reconstructions; bottom row: magnitude of the reconstruction error.

dersampling masks in k-space, using either Cartesian or 2D random sampling patterns [44,71], at undersampling ratios ranging from $2.5\times$ to $7\times$. The proposed STROLL-MRI scheme is applied to reconstruct the MR images. We set the weights of the STROLLR regularizers to $\gamma^S = \gamma^{LR} = 10^{-6}$. The sparsity and low-rankness penalty coefficients $\theta = 2\lambda = \theta_0$, where θ_0 depends on undersampling ratio of the k-space measurement, as well as the image distribution. For the three anatomical images **a**, **b** and **d**, we set $\theta_0 = 0.02$ when the undersampling ratio is smaller than or equal to $5\times$, and $\theta_0 = 0.05$ when undersampling ratio is higher than $5\times$. For the physical phantom image **c**, which has large piece-wise smooth regions, we set $\theta_0 = 0.05$ for reconstructing $2.5\times$ undersampled k-space measurement. We run STROLLR-MRI for 100 iterations, and have observed the empirical convergence of the objective functions.

We first compare our STROLLR-MRI reconstruction results to those obtained using popular *internal* methods, including naive Zero-filling, Sparse MRI [72], PBDWS [82], PANO [83], DL-MRI [73], TL-MRI [44], and FRIST-MRI [43]. To evaluate the performance of the MRI reconstruction schemes, we measure the reconstruction PSNRs (computed for image magnitudes) for various approaches, which are listed in Table 5.7. The proposed STROLLR-MRI algorithm provides PSNR improvements (averaged over all 7 cases) of 2.3dB, 2.5dB, 3.1dB, 2.9dB, 3.0dB, 7.7dB, over the FRIST-MRI, TL-MRI, PANO, PBDWS, DL-MRI, and Sparse MRI, respectively. Even when compared to the recently proposed TL-MRI and FRIST-MRI, our STROLLR-MRI provides much better reconstruction results. The quality improvement obtained by STROLLR-MRI demonstrates the effectiveness of the learned STROLLR as the regularizer, as it utilizes both sparsity and non-local similarity. Figure 5.6 compares the reconstructed MR images, and the magnitudes of their reconstruction error (i.e., difference between the magnitudes of the reconstructed images and the reference images) using DL-MRI, FRIST-MRI, and STROLLR-MRI. The reconstruction result using STROLLR-MRI contains fewer artifacts and less noise, compared to competing methods.

Finally, we compare the proposed STROLLR-MRI scheme to the recent ADMM-Net [124], which is an *external* method using deep learning. We use the publicly available implementation with the trained 15-stage model by Yang et al. [124]. Because the released ADMM-Net model requires the MR image to have fixed size (i.e., 256×256) with a specific sampling mask



(a) TL-MRI, 32.4dB (b) ADMM-Net, 35.9dB(c) STROLLR-MRI 37.3dB

Figure 5.7: Example of CS MRI of the *Brain Data* 1 using the 5× undersampled pseudo radial mask in K-space. Reconstructions using (a) TL-MRI, (b) ADMM-Net, and (c) the proposed STROLLR-MRI (top row), and the magnitudes of the corresponding reconstruction error (bottom row).

(i.e., pseudo radial sampling pattern with 5× undersampling ratio) [124], the testing images **a-d** are not applicable for direct comparison. Instead, we reconstruct the two example MR images that were used in the ADMM-Net demonstration [124]. The proposed STROLLR-MRI algorithm provides PSNR improvements of 1.4dB and 1.3dB over ADMM-Net, for reconstructing the testing images *Brain data* 1 and *Brain data* 2 , respectively. Figure 5.7 compares the reconstructed MR images with the magnitudes of their corresponding reconstruction errors using TL-MRI [44], ADMM-Net [124], and STROLLR-MRI. It is clear that the reconstruction result using STROLLR-MRI provides less noise and fewer artifacts than ADMM-Net. In contrast to deep learning algorithms, the proposed STROLLR-MRI does not require re-training the model using different sampling masks, over an image corpus with similar distribution, when reconstructing a class of MR images.

Table 5.7: PSNRs, corresponding to the Zero-filling, Sparse MRI, DL-MRI, and the proposed STROLLR-MRI reconstructions for various images, sampling schemes, and undersampling factors. The best PSNR for each case is marked in bold.

Image	Sampling Scheme	Under-sampl.	Zero-filling	Sparse MRI	DL-MRI	PBDWS
a	2D Random	5×	26.9	27.9	30.5	30.3
b	Cartesian	2.5×	28.1	31.7	37.5	42.5
c	Cartesian	2.5×	24.9	29.9	36.6	35.8
d	Cartesian	4×	28.9	29.7	32.7	31.7
	Cartesian	7×	27.9	28.6	30.9	31.1
	2D Random	4×	25.2	26.1	33.0	32.8
	2D Random	7×	25.3	26.4	31.7	30.9

Image	Sampling Scheme	Under-sampl.	PANO	TL-MRI	FRIST-MRI	STROLLR-MRI
a	2D Random	5×	30.4	30.6	30.7	**32.4**
b	Cartesian	2.5×	40.0	40.7	40.9	**44.0**
c	Cartesian	2.5×	34.8	36.3	36.7	**40.9**
d	Cartesian	4×	32.7	32.8	33.0	**35.2**
	Cartesian	7×	31.1	31.2	31.4	**32.8**
	2D Random	4×	32.8	33.1	33.1	**35.6**
	2D Random	7×	30.9	31.9	32.1	**33.3**

CHAPTER 6

ONLINE TENSOR RECONSTRUCTION FOR VIDEO DENOISING

6.1 Related Methods

Natural images and videos have local structures that are sparse or compressible in some transform domain or over certain dictionaries [86, 142, 143], 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 [85, 86]. 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 [32, 37, 47, 49, 56, 59, 87, 144–147].

The well-known BM3D image denoising algorithm [32] has been extended to both volumetric data [56] and video data [59]. 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 [59]. Buades et al. proposed a similar video denoising algorithm that differs in both the patch grouping and denoising strategy [144]. 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 [145]. Rather than transform domain threshold-

Table 6.1: Comparison of the key attributes between the proposed SALT denoising, its variations, and the competing methods.

Methods	**Local Sparse Model**			BM	**Non-Local Method**
	Fixed	Adaptive	Online		
fBM3D [32]	✓			✓	✓
sKSVD [31]		✓			
VIDOSAT [6]		✓	✓		
VBM3D [87] / VBM4D [59]	✓			✓	✓
BM-DCT	✓			✓	
BM-TL		✓	✓	✓	
BM-LR				✓	✓
SALT		✓	✓	✓	✓

ing, they denoise the resulting tensor using a low-rank approximation. A recent approach splits videos into sparse and low-rank "layers" before denoising [148].

Table 6.1 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.

6.2 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 [6] 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

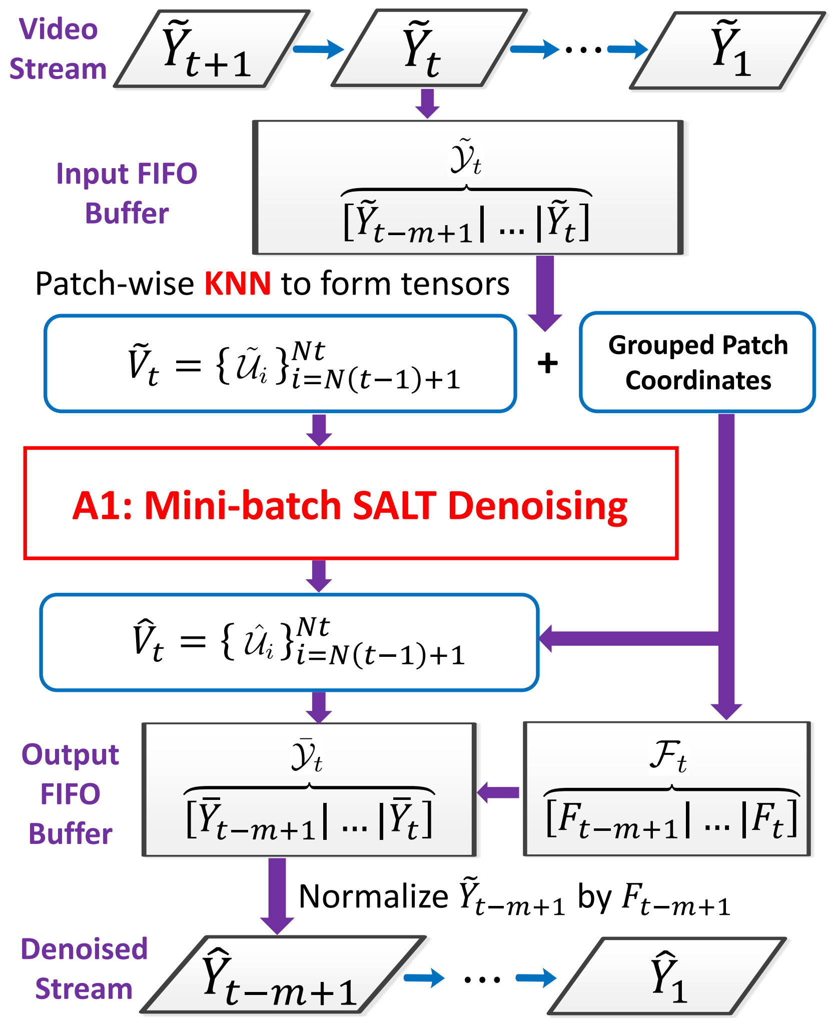

Video Stream
$\tilde{Y}_{t+1}$
$\tilde{Y}_t$
$\tilde{Y}_1$
Input FIFO Buffer
$\overbrace{[\tilde{Y}_{t-m+1} | \dots | \tilde{Y}_t]}^{\tilde{\mathcal{Y}}_t}$
Patch-wise KNN to form tensors
$\tilde{V}_t = \{\tilde{\mathcal{U}}_i\}_{i=N(t-1)+1}^{Nt}$
+
Grouped Patch Coordinates
A1: Mini-batch SALT Denoising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

Figure 6.1: A diagram for SALT based video denoising

video typically involves various types of motion, patch grouping methods are widely used to generate high-dimensional data with better correlation and redundancy [59, 145]. 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 (see Section 6.4 for more details). The reconstructed tensors are aggregated to output denoised frame estimates.

Figure 6.1 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 $\tilde{\mathcal{Y}}_t = \left[\tilde{Y}_{t-m+1} \mid \tilde{Y}_{t-m+2} \mid \; ... \; \mid \tilde{Y}_t\right]$, 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 outputs of the mini-batch denoising algorithm $\hat{V}_t = \{\hat{\mathcal{U}}_i\}$ are accumulated to the fixed-size output buffer $\bar{\mathcal{Y}}_t = \left[\bar{Y}_{t-m+1} \mid \; ... \; \mid \bar{Y}_t\right] \in \mathbb{R}^{a\times b\times m}$, 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 $\mathcal{F}_t = \left[F_{t-m+1} \mid \; ... \; \mid F_t\right] \in \mathbb{R}^{a\times b\times m}$. 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 $\hat{Y}_\tau$.

6.3 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 Fig. 6.1.

6.3.1 Online Sparsifying Transform Learning with a 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 $t = 1, 2, ...,$ 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 $1 \le \tau \le t-1$ [6,28].

6.3.2 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

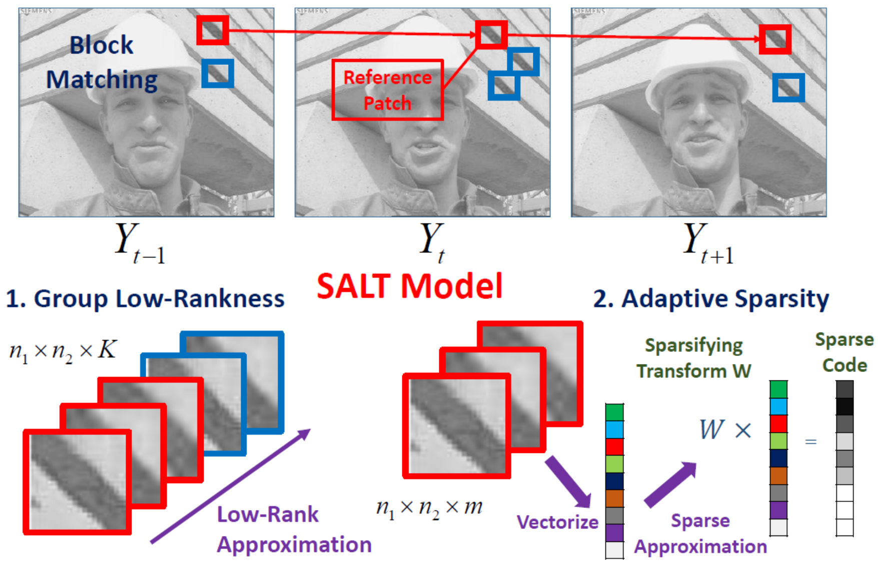


Figure 6.2: A simple illustration of the SALT model for video

some reshaping operations on tensors. We use $\text{mat}(\cdot) : \mathbb{R}^{n_1 \times n_2 \times K} \to \mathbb{R}^{n_s \times K}$ 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 $\text{vec}(\cdot) : \mathbb{R}^{n_s \times K} \to \mathbb{R}^{n}$ 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 $U = \text{mat}(\mathcal{U})$, and its vectorization $u = \text{vec}(U)$ can be summarized by the following diagram:

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

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 6.2 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 $t = 1, 2, 3, ...$):

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

where 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 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 6.3.1.

6.3.3 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 6.3.2, and used in the proposed video denoising framework. The modified mini-batch SALT denoising problem is the following:

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

where $S_\tau = \{N(\tau-1)+1, \dots, N\tau\}$ 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 [6,28]. This is motivated by two factors: 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 [6]. 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 $n = n_s \times m$.

6.4 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 $\left\{\alpha_i\right\}_{i \in S_t}$, W_t,

$\left\{D_i\right\}_{i \in S_t}$, and $\left\{U_i\right\}_{i \in S_t}$, 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 $\text{mat}(\tilde{\mathcal{U}}_i)$ 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 $\hat{\mathcal{U}}_i = \text{mat}^{-1}(\hat{U}_i)$, 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 $u_i^m = \text{vec}(C_{1:m}U_i)$. We solve the Problem (P3) for the optimal sparse code $\forall i \in S_t$,

$$\hat{\alpha}_i = \underset{\alpha_i}{\text{argmin}} \; \left\|\hat{W}_{t-1}u_i^m - \alpha_i\right\|_2^2 + \rho^2 \left\|\alpha_i\right\|_0 \,, \tag{6.1}$$

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 [63], where the hard thresholding operator $H_\rho(\cdot)$ is defined as

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

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.

(ii) **Mini-batch transform update**. Fixing $\left\{u_i^m\right\}_{i=1}^{Nt}$ 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:

$$\begin{aligned} \hat{W}_t = \underset{W}{\text{argmin}} \; & \frac{1}{tN}\sum_{\tau=1}^{t} \varrho^{t-\tau} \sum_{i \in S_\tau} \left\|W u_i^m - \hat{\alpha}_i\right\|_2^2 \\ s.t. \quad & W^T W = I_n \,. \end{aligned} \tag{6.2}$$

Prior work on batch unitary transform learning introduced closed-form transform update [63]. Similarly, the optimal solution $\hat{W}_t$ to problem (6.2) 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 (6.2) 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

$$\hat{D}_i = \underset{D_i}{\operatorname{argmin}}\ \|U_i - D_i\|_F^2 + \theta^2 \operatorname{rank}(D_i)\,. \tag{6.3}$$

Suppose the economy-size SVD of U_i is $\Lambda_i \Omega_i \Delta_i^T = \mathrm{SVD}(U_i)$. Then (6.3) 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:

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

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

$$\hat{U}_{i,1} = \frac{\gamma_s \mathrm{vec}^{-1}(W_t^T\, \hat{\alpha}_i) + C_{1\,:\,m}(\gamma_l \hat{D}_i + \gamma_f \mathrm{mat}(\tilde{\mathcal{U}}_i))}{\gamma_s + \gamma_l + \gamma_f} \tag{6.4}$$

$$\hat{U}_{i,2} = \frac{C_{m+1\,:\,K}(\gamma_l\, \hat{D}_i + \gamma_f\, \mathrm{mat}(\tilde{\mathcal{U}}_i))}{\gamma_l + \gamma_f}\,. \tag{6.5}$$

When the iteration completes at time t, each denoised $\hat{U}_i$ is tensorized to be $\hat{\mathcal{U}}_i = \mathrm{mat}^{-1}(\hat{U}_i)$ as output. Algorithm 6.1 summarizes the SALT mini-batch denoising algorithm.

Algorithm Complexity. The computational cost of the 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.

6.5 Experiment

6.5.1 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 [91] [1] and Tampere University of Technology (TUT) dataset [59, 87]. 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 $\sigma = 5$, 10, 15, 20, and 50) for each video.

Implementation details. We explain several implementation details and minor modifications. First, at each time instant t, instead of grouping the noisy patches directly by KNN, we pre-clean the input buffer sequentially, 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 [149]. The reconstructions 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 un-

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

supervised 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 chapter. 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 Algorithm 6.1, Step 4(b)), where s_i is the sparsity of $\hat{\alpha}_i$ (see Algorithm 6.1, 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 [6], which are tuned empirically for each σ. We initialize the sparsifying transform with the 3D DCT W_0.

6.5.2 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) [32], sparse KSVD image sequence denoising (sKSVD) [31], VIDOSAT [6], VBM3D [87], and VBM4D [59]. 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 as 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, we fix the sparsifying transform in BM-TL as 3D DCT, and denote such a method as BM-DCT. Table 6.1 summarizes the key attributes of the SALT denoising, as well as other competing methods.

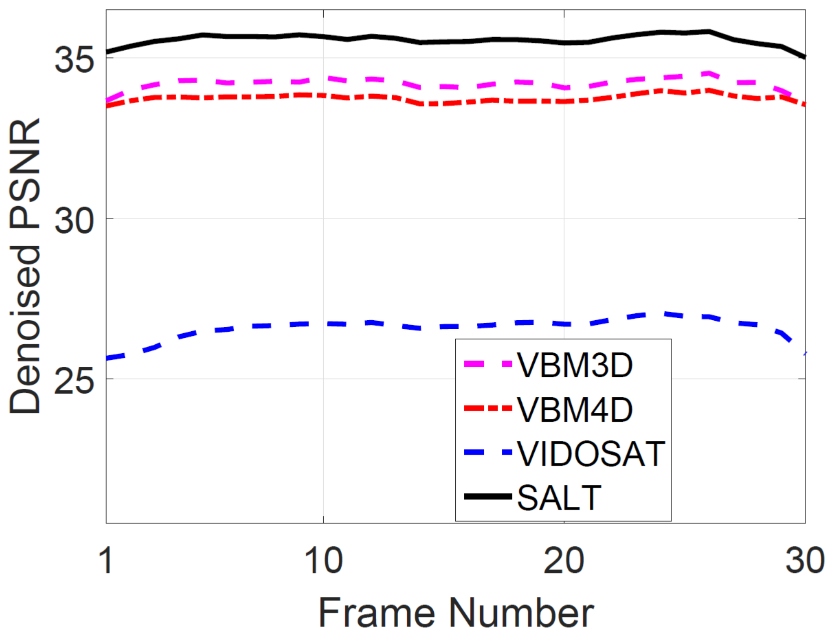
Denoising results. To evaluate the performance of the denoising schemes, we measure the peak signal-to-noise-ratio (PSNR) in decibel (dB), which is computed between the noiseless reference and the denoised video. Table 6.2 and 6.3 list the average denoised PSNRs over videos from TUT and ASU (excluding 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 6.3 and 6.5 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 6.4 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 Figs. 6.4(a) and (b)), obtained by SALT (see Fig. 6.4(c)) and VIDOSAT (see Fig. 6.4(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 Figs. 6.4(e) and (f)). Similarly in Fig. 6.6, we observe better denoised result by SALT compared to VBM4D.

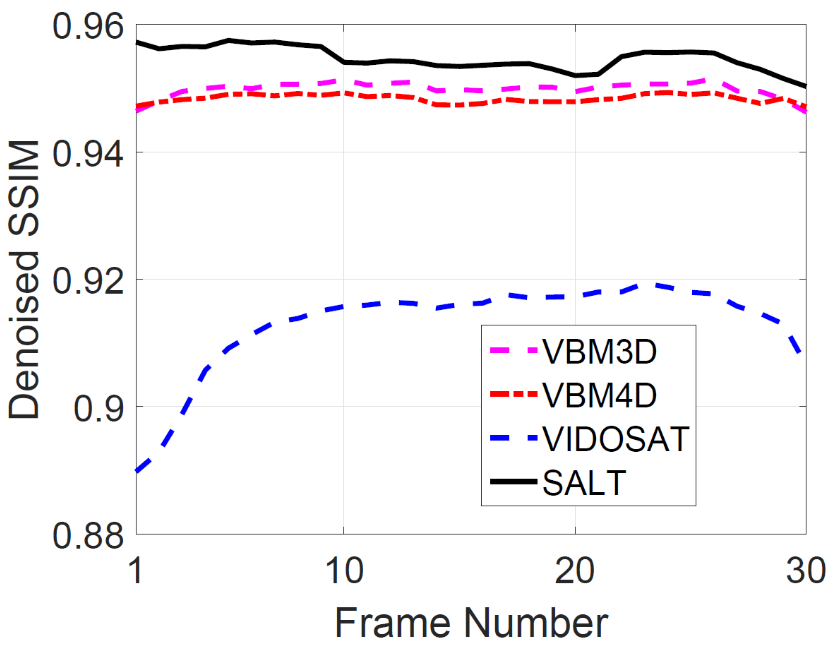
6.6 Conclusion

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.

```
Algorithm 6.1: 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: $S_\tau = \{N(\tau-1)+1, \ldots, N\tau\}$.

    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 (6.4).
        (c) Reconstruct last $K-m$ columns of $\hat{U}_i$ by (6.5).
        (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 $\hat{W}_t$.
```

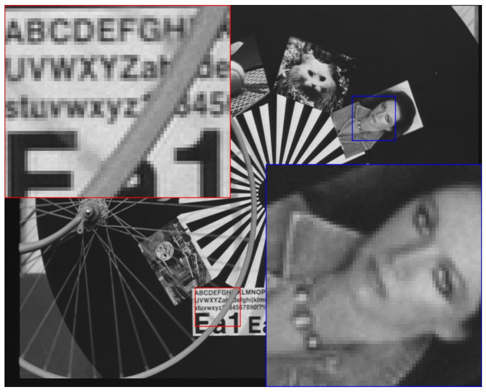


(a)

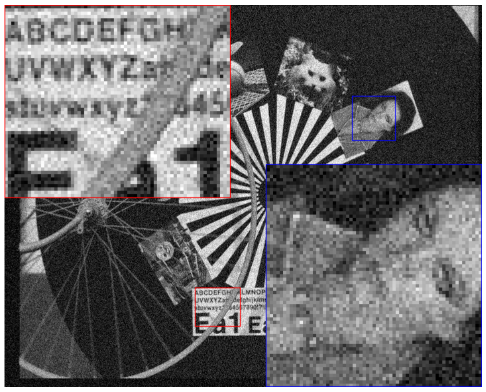


(b)

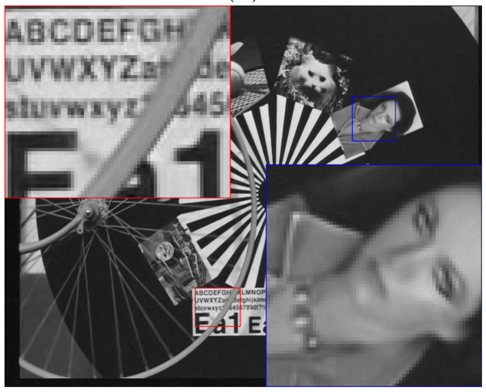
Figure 6.3: Frame-by-frame (a) PSNR(dB) and (b) SSIM of the video *Gbicycle* with $\sigma = 20$, denoised by the proposed SALT denoising scheme, VIDOLSAT, VBM3D and VBM4D.



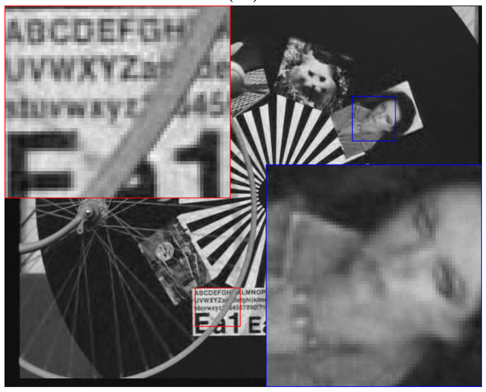
(a)



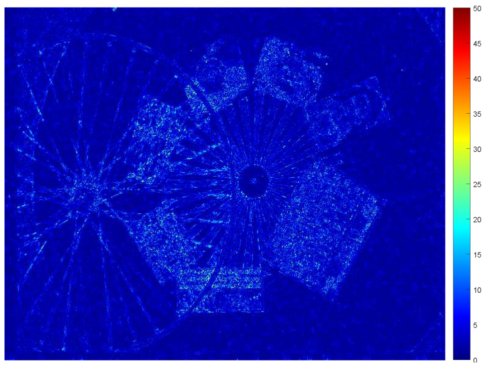
(b)



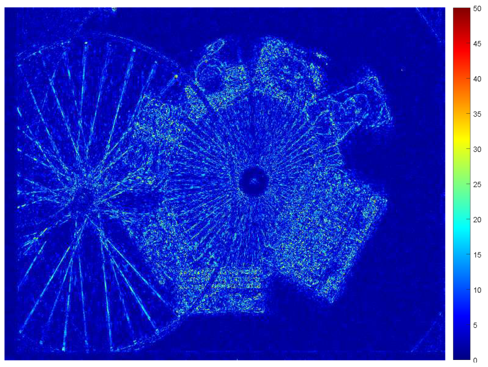
(c)



(d)

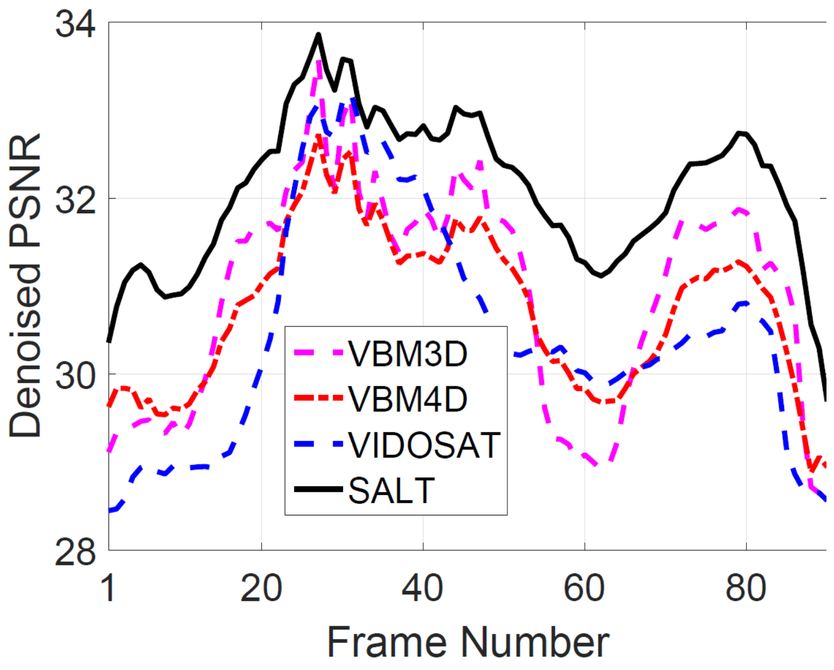


(e)

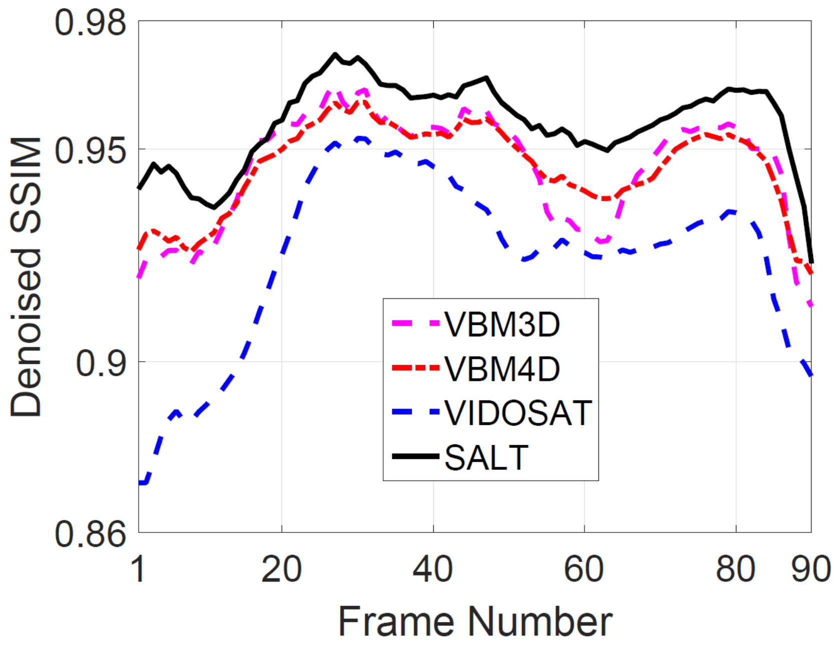


(f)

Figure 6.4: Denoising result: (a) One frame of the clean video *Gbicycle*. (b) Frame corrupted with noise at $\sigma = 20$ (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).



(a)



(b)

Figure 6.5: 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.

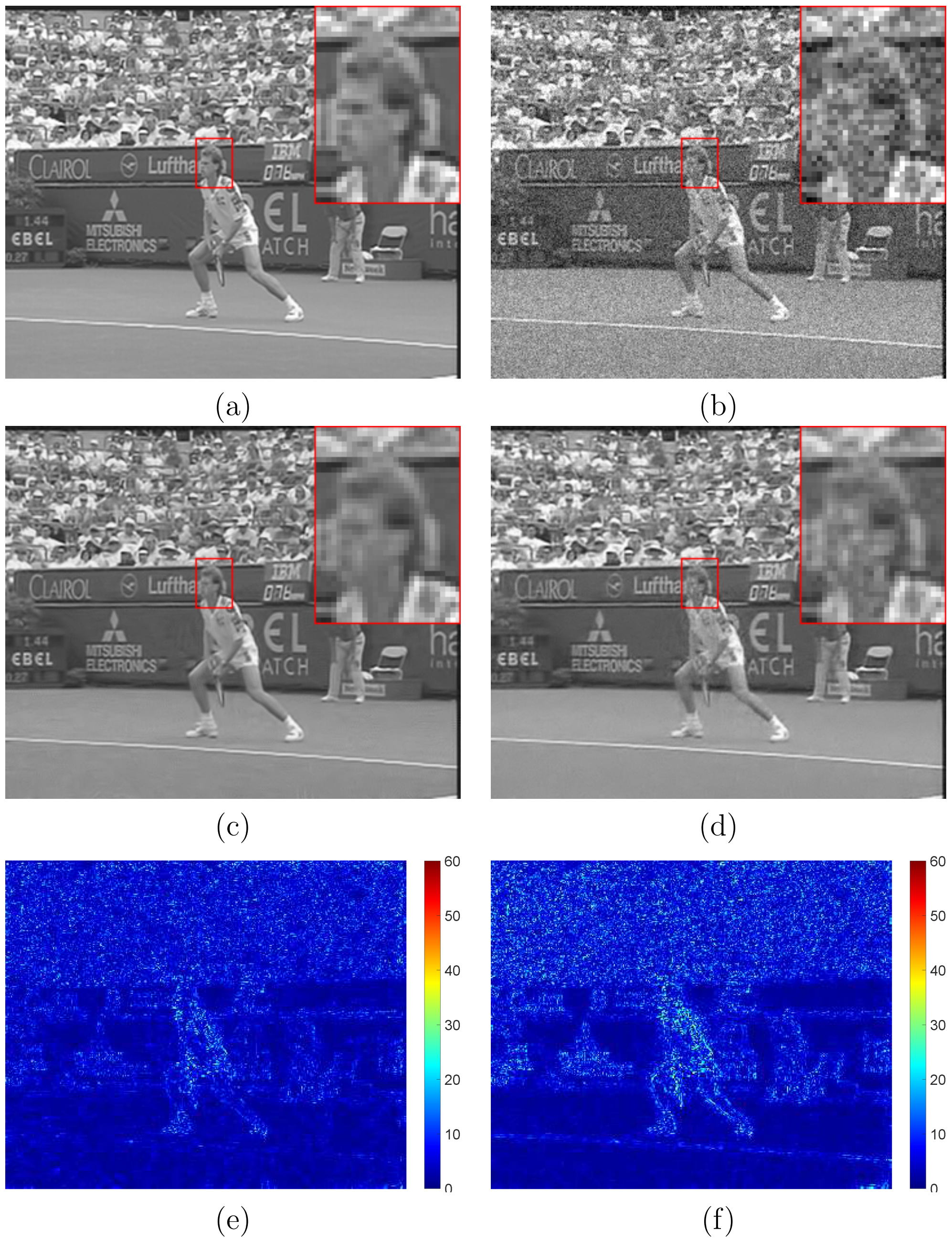


Figure 6.6: Denoising result: (a) One frame of the clean video *Stefan*. (b) Frame corrupted with noise at $\sigma = 20$ (PSNR = 22.11 dB). (c) Denoised frame using the proposed SALT denoising (PSNR = 29.69 dB). (d) Denoised frame using VBM4D (PSNR = 28.56 dB). (e) Magnitude of error in (c). (f) Magnitude of error in (d).

Table 6.2: 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	**TUT** Dataset (8 videos)					Δ P
σ	5	10	15	20	50	(std.)
fBM3D [32]	38.05	34.06	31.89	30.42	25.88	-2.86 (0.78)
sKSVD [31]	38.87	34.95	32.80	31.33	26.89	-1.95 (1.02)
VIDO-SAT [6]	39.56	35.75	33.54	31.98	27.29	-1.30 (0.92)
VBM3D [87]	39.20	35.75	33.87	32.49	26.51	-1.36 (0.57)
VBM4D [59]	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 6.3: 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.

Data	**ASU** Dataset (18 videos)					Δ P
σ	5	10	15	20	50	(std.)
fBM3D [32]	39.44	35.47	33.26	31.73	27.00	-3.90 (0.95)
sKSVD [31]	41.83	38.09	35.96	34.46	29.80	-1.45 (0.77)
VIDO-SAT [6]	42.49	38.63	36.36	34.79	29.78	-0.87 (0.47)
VBM3D [87]	41.66	38.55	36.32	34.70	29.72	-1.09 (0.82)
VBM4D [59]	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

REFERENCES

[1] A. M. Bruckstein, D. L. Donoho, and M. Elad, "From sparse solutions of systems of equations to sparse modeling of signals and images," *SIAM Review*, vol. 51, no. 1, pp. 34–81, 2009.

[2] M. Elad, P. Milanfar, and R. Rubinstein, "Analysis versus synthesis in signal priors," *Inverse Problems*, vol. 23, no. 3, p. 947, 2007.

[3] W. K. Pratt, J. Kane, and H. C. Andrews, "Hadamard transform image coding," *Proceedings of the IEEE*, vol. 57, no. 1, pp. 58–68, 1969.

[4] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transactions on Image Processing*, vol. 15, no. 12, pp. 3736–45, Dec. 2006.

[5] B. Wen, S. Ravishankar, and Y. Bresler, "Structured overcomplete sparsifying transform learning with convergence guarantees and applications," *International Journal of Computer Vision*, vol. 114, no. 2-3, pp. 137–167, 2015.

[6] B. Wen, S. Ravishankar, and Y. Bresler, "Video denoising by online 3D sparsifying transform learning," in *IEEE International Conference on Image Processing (ICIP)*. IEEE, 2015, pp. 118–122.

[7] K. Engan, S. O. Aase, and J. H. Husoy, "Method of optimal directions for frame design," in *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*, vol. 5. IEEE, 1999, pp. 2443–2446.

[8] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD : An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[9] G. Davis, S. Mallat, and M. Avellaneda, "Adaptive greedy approximations," *Constructive Approximation*, vol. 13, no. 1, pp. 57–98, 1997.

[10] Y. C. Pati, R. Rezaiifar, and P. Krishnaprasad, “Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition,” in *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on.* IEEE, 1993, pp. 40–44.

[11] S. G. Mallat and Z. Zhang, “Matching pursuits with time-frequency dictionaries,” *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[12] S. S. Chen, D. L. Donoho, and M. A. Saunders, “Atomic decomposition by basis pursuit,” *SIAM Review*, vol. 43, no. 1, pp. 129–159, 2001.

[13] M. Yaghoobi, T. Blumensath, and M. E. Davies, “Dictionary learning for sparse approximations with the majorization method,” *IEEE Transactions on Signal Processing*, vol. 57, no. 6, pp. 2178–2191, 2009.

[14] K. Skretting and K. Engan, “Recursive least squares dictionary learning algorithm,” *IEEE Transactions on Signal Processing*, vol. 58, no. 4, pp. 2121–2130, 2010.

[15] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, “Online learning for matrix factorization and sparse coding,” *Journal of Machine Learning Research*, vol. 11, no. Jan, pp. 19–60, 2010.

[16] J. Mairal, M. Elad, and G. Sapiro, “Sparse representation for color image restoration,” *IEEE Transactions on Image Processing*, vol. 17, no. 1, pp. 53–69, 2008.

[17] M. Aharon and M. Elad, “Sparse and redundant modeling of image content using an image-signature-dictionary,” *SIAM Journal on Imaging Sciences*, vol. 1, no. 3, pp. 228–247, 2008.

[18] R. Rubinstein, A. M. Bruckstein, and M. Elad, “Dictionaries for sparse representation modeling,” *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1045–1057, 2010.

[19] S. Mallat, *A Wavelet Tour of Signal Processing.* Academic Press, 1999.

[20] S. Ravishankar and Y. Bresler, “Learning sparsifying transforms,” *IEEE Transactions on Signal Processing*, vol. 61, no. 5, pp. 1072–1086, 2013.

[21] L. Pfister and Y. Bresler, “Learning sparsifying filter banks,” in *Proc. SPIE Wavelets & Sparsity XVI*, vol. 9597. SPIE, Aug. 2015.

[22] L. Pfister and Y. Bresler, "Model-based iterative tomographic reconstruction with adaptive sparsifying transforms," in *Proc. SPIE Computational Imaging XII*, C. A. Bouman and K. D. Sauer, Eds. SPIE, Mar. 2014, pp. 90 200H–90 200H–11.

[23] S. Ravishankar and Y. Bresler, "Sparsifying transform learning with efficient optimal updates and convergence guarantees," *IEEE Transactions on Signal Processing*, vol. 63, no. 9, pp. 2389–2404, 2015.

[24] B. Wen, S. Ravishankar, and Y. Bresler, "Learning overcomplete sparsifying transforms with block cosparsity," in *IEEE International Conference on Image Processing (ICIP).* IEEE, 2014, pp. 803–807.

[25] H. Wersing, J. Eggert, and E. Körner, "Sparse coding with invariance constraints," in *Artificial Neural Networks and Neural Information Processing—ICANN/ICONIP 2003.* Springer, 2003, pp. 385–392.

[26] Z. Zhan, J.-F. Cai, D. Guo, Y. Liu, Z. Chen, and X. Qu, "Fast multiclass dictionaries learning with geometrical directions in MRI reconstruction," *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 9, pp. 1850–1861, 2016.

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

[28] S. Ravishankar, B. Wen, and Y. Bresler, "Online sparsifying transform learning—part i: Algorithms," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 4, pp. 625–636, 2015.

[29] S. Ravishankar, B. Wen, and Y. Bresler, "Online sparsifying transform learning for signal processing," in *Signal and Information Processing (GlobalSIP), 2014 IEEE Global Conference on.* IEEE, 2014, pp. 364–368.

[30] M. Protter and M. Elad, "Image sequence denoising via sparse and redundant representations," *IEEE Transactions on Image Processing*, vol. 18, no. 1, pp. 27–35, 2009.

[31] R. Rubinstein, M. Zibulevsky, and M. Elad, "Double sparsity: Learning sparse dictionaries for sparse signal approximation," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1553–1564, 2010.

[32] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *IEEE Transactions on Image Processing*, vol. 16, no. 8, pp. 2080–2095, 2007.

[33] G. Yu and G. Sapiro, "Dct image denoising: a simple and effective image denoising algorithm," *Image Processing On Line*, vol. 1, pp. 292–296, 2011.

[34] S. G. Chang, B. Yu, and M. Vetterli, "Adaptive wavelet thresholding for image denoising and compression," *IEEE Transactions on Image Processing*, vol. 9, no. 9, pp. 1532–1546, 2000.

[35] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[36] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. Image Process.*, vol. 15, no. 12, pp. 3736–3745, 2006.

[37] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *IEEE International Conference on Computer Vision (ICCV)*, Sept 2009, pp. 2272–2279.

[38] S. Ravishankar and Y. Bresler, "ℓ_0 sparsifying transform learning with efficient optimal updates and convergence guarantees," *IEEE Transactions on Signal Processing*, vol. 63, no. 9, pp. 2389–2404, 2014.

[39] B. Wen, Y. Li, and Y. Bresler, "When sparsity meets low-rankness: Transform learning with non-local low-rank constraint for image restoration," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 2297–2301.

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

[41] B. Wen, S. Ravishankar, and Y. Bresler, "Vidosat: High-dimensional sparsifying transform learning for online video denoising," *IEEE Transactions on Image Processing*, 2018.

[42] S. Ravishankar and Y. Bresler, "Data-driven learning of a union of sparsifying transforms model for blind compressed sensing," *IEEE Transactions on Computational Imaging*, vol. 2, no. 3, pp. 294–309, 2016.

[43] B. Wen, S. Ravishankar, and Y. Bresler, "FRIST- flipping and rotation invariant sparsifying transform learning and applications," *Inverse Problems*, vol. 33, no. 7, p. 074007, 2017.

[44] S. Ravishankar and Y. Bresler, “Efficient blind compressed sensing using sparsifying transforms with convergence guarantees and application to magnetic resonance imaging,” *SIAM Journal on Imaging Sciences*, vol. 8, no. 4, pp. 2519–2557, 2015.

[45] L. Pfister and Y. Bresler, “Tomographic reconstruction with adaptive sparsifying transforms,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 6914–6918.

[46] A. Buades, B. Coll, and J.-M. Morel, “A non-local algorithm for image denoising,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, June 2005, pp. 60–65 vol. 2.

[47] W. Dong, X. Li, L. Zhang, and G. Shi, “Sparsity-based image denoising via dictionary learning and structural clustering,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2011, pp. 457–464.

[48] W. Dong, G. Shi, and X. Li, “Nonlocal image restoration with bilateral variance estimation: a low-rank approach,” *IEEE Transactions on Image Processing*, vol. 22, no. 2, pp. 700–711, 2013.

[49] J. Zhang, D. Zhao, and W. Gao, “Group-based sparse representation for image restoration,” *IEEE Transactions on Image Processing*, vol. 23, no. 8, pp. 3336–3351, 2014.

[50] S. Gu, Q. Xie, D. Meng, W. Zuo, X. Feng, and L. Zhang, “Weighted nuclear norm minimization and its applications to low level vision,” *International Journal of Computer Vision*, vol. 121, no. 2, pp. 183–208, 2017.

[51] H. Yoon, K. S. Kim, D. Kim, Y. Bresler, and J. C. Ye, “Motion adaptive patch-based low-rank approach for compressed sensing cardiac cine MRI,” *IEEE Transactions on Medical Imaging*, vol. 33, no. 11, pp. 2069–2085, 2014.

[52] Z. Zha, X. Liu, Z. Zhou, X. Huang, J. Shi, Z. Shang, L. Tang, Y. Bai, Q. Wang, and X. Zhang, “Image denoising via group sparsity residual constraint,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 1787–1791.

[53] B. Wen, Y. Li, and Y. Bresler, “The power of complementary regularizers: Image recovery via transform learning and low-rank modeling,” arXiv preprint arXiv:1808.01316, 2018.

[54] D. Liu, B. Wen, X. Liu, and T. S. Huang, “When image denoising meets high-level vision tasks: A deep learning approach,” in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2018, pp. 842–848.

[55] D. Liu, B. Wen, J. Jiao, X. Liu, Z. Wang, and T. S. Huang, "Connecting image denoising and high-level vision tasks via deep learning," arXiv preprint arXiv:1809.01826, 2018.

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

[57] Y. Peng, D. Meng, Z. Xu, C. Gao, Y. Yang, and B. Zhang, "Decomposable nonlocal tensor dictionary learning for multispectral image denoising," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6 2014, pp. 2949–2956.

[58] B. Wen, U. S. Kamilov, D. Liu, H. Mansour, and P. T. Boufounos, "Deepcasd: An end-to-end approach for multi-spectral image super-resolution," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 6503–6507.

[59] 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*, vol. 21, no. 9, pp. 3952–3966, 2012.

[60] I. Tosic and P. Frossard, "Dictionary learning," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 27–38, Mar 2011.

[61] B. Natarajan, "Sparse approximate solutions to linear systems," *SIAM Journal on Computing*, vol. 24, no. 2, pp. 227–234, 1995.

[62] D. Needell and J. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," *Appl. Comput. Harmon. Anal.*, vol. 26, no. 3, pp. 301–321, May 2009.

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

[64] S. Ravishankar and Y. Bresler, "Online sparsifying transform learning—part ii: Convergence analysis," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 4, pp. 637–646, 2015.

[65] D. G. Lowe, "Object recognition from local scale-invariant features," in *IEEE International Conference on Computer Vision (ICCV)*, vol. 2, 1999, pp. 1150–1157.

[66] Y. Ke and R. Sukthankar, "PCA-SIFT: A more distinctive representation for local image descriptors," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2. IEEE, 2004, pp. II–II.

[67] E. Le Pennec and S. Mallat, "Bandelet image approximation and compression," *Multiscale Modeling & Simulation*, vol. 4, no. 3, pp. 992–1039, 2005.

[68] X. Qu, D. Guo, B. Ning, Y. Hou, Y. Lin, S. Cai, and Z. Chen, "Undersampled MRI reconstruction with patch-based directional wavelets," *Magnetic Resonance Imaging*, vol. 30, no. 7, pp. 964–977, 2012.

[69] S. Ravishankar and Y. Bresler, "Learning doubly sparse transforms for images," *IEEE Transactions on Image Processing*, vol. 22, no. 12, pp. 4598–4612, 2013.

[70] L. Zelnik-Manor, K. Rosenblum, and Y. C. Eldar, "Dictionary optimization for block-sparse representations," *IEEE Transactions on Signal Processing*, vol. 60, no. 5, pp. 2386–2395, 2012.

[71] J. Trzasko and A. Manduca, "Highly undersampled magnetic resonance image reconstruction via homotopic *ell*_{0}-minimization," *IEEE Transactions on Medical Imaging*, vol. 28, no. 1, pp. 106–121, 2009.

[72] M. Lustig, D. Donoho, and J. M. Pauly, "Sparse MRI: The application of compressed sensing for rapid MR imaging," *Magnetic Resonance in Medicine*, vol. 58, no. 6, pp. 1182–1195, 2007.

[73] S. Ravishankar and Y. Bresler, "MR image reconstruction from highly undersampled k-space data by dictionary learning," *IEEE Transactions on Medical Imaging*, vol. 30, no. 5, pp. 1028–1041, 2011.

[74] S. Gleichman and Y. C. Eldar, "Blind compressed sensing," *IEEE Transactions on Information Theory*, vol. 57, no. 10, pp. 6958–6975, 2011.

[75] "The USC-SIPI Database," online: http://sipi.usc.edu/database/database.php?volume=misc.

[76] O. Bryt and M. Elad, "Compression of facial images using the k-svd algorithm," *Journal of Visual Communication and Image Representation*, vol. 19, no. 4, pp. 270–282, 2008.

[77] "Michael Elad's Homepage," online: http://www.cs.technion.ac.il/ẽlad.

[78] "Transform Learning Website," online: http://transformlearning.csl.illinois.edu/.

[79] T. Yang, *Finite Element Structural Analysis.* Prentice Hall, 1986.

[80] D. Watson, *Contouring: A Guide to the Analysis and Display of Spatial Data.* Elsevier, 2013, vol. 10.

[81] I. Ram, M. Elad, and I. Cohen, "Image processing using smooth ordering of its patches," *IEEE Transactions on Image Processing*, vol. 22, no. 7, pp. 2764–2774, 2013.

[82] B. Ning, X. Qu, D. Guo, C. Hu, and Z. Chen, "Magnetic resonance image reconstruction using trained geometric directions in 2d redundant wavelets domain and non-convex optimization," *Magnetic Resonance Imaging*, vol. 31, no. 9, pp. 1611–1622, 2013.

[83] X. Qu, Y. Hou, F. Lam, D. Guo, J. Zhong, and Z. Chen, "Magnetic resonance image reconstruction from undersampled measurements using a patch-based nonlocal operator," *Medical Image Analysis*, vol. 18, no. 6, pp. 843–856, 2014.

[84] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

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

[86] N. Rajpoot, Z. Yao, and R. Wilson, "Adaptive wavelet restoration of noisy video sequences," in *IEEE International Conference on Image Processing (ICIP)*, vol. 2, 2004, pp. 957–960.

[87] K. Dabov, A. Foi, and K. Egiazarian, "Video denoising by sparse 3D transform-domain collaborative filtering," in *Signal Processing Conference, 2007 15th European*, Sept 2007, pp. 145–149.

[88] S. Dev, B. Wen, Y. H. Lee, and S. Winkler, "Ground-based image analysis: A tutorial on machine-learning techniques and applications," *IEEE Geoscience and Remote Sensing Magazine*, vol. 4, no. 2, pp. 79–93, 2016.

[89] T. Ouni and M. Abid, "Scan methods and their application in image compression," *International Journal of Signal Processing, Image Processing and Pattern Recognition*, vol. 5, no. 3, pp. 49–64, 2012.

[90] D. Le Gall, "MPEG: A video compression standard for multimedia applications," *Communications of the ACM*, vol. 34, no. 4, pp. 46–58, 1991.

[91] 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*, vol. 2014, 2014.

[92] H. C. Burger, C. Schuler, and S. Harmeling, "Learning how to combine internal and external denoising methods," in *German Conference on Pattern Recognition.* Springer, 2013, pp. 121–130.

[93] M. Zontak and M. Irani, "Internal statistics of a single natural image," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* IEEE, 2011, pp. 977–984.

[94] F. Chen, L. Zhang, and H. Yu, "External patch prior guided internal clustering for image denoising," in *IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 603–611.

[95] Z. Wang, Y. Yang, Z. Wang, S. Chang, J. Yang, and T. S. Huang, "Learning super-resolution jointly from external and internal examples," *IEEE Transactions on Image Processing*, vol. 24, no. 11, pp. 4359–4371, 2015.

[96] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1-4, pp. 259–268, 1992.

[97] A. Beck and M. Teboulle, "Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems," *IEEE Transactions on Image Processing*, vol. 18, no. 11, pp. 2419–2434, 2009.

[98] W. Dong, G. Shi, Y. Ma, and X. Li, "Image restoration via simultaneous sparse coding: Where structured sparsity meets gaussian scale mixture," *International Journal of Computer Vision*, vol. 114, no. 2-3, pp. 217–232, 2015.

[99] W. Dong, G. Shi, X. Li, Y. Ma, and F. Huang, "Compressive sensing via nonlocal low-rank regularization," *IEEE Transactions on Image Processing*, vol. 23, no. 8, pp. 3618–3632, 2014.

[100] J. Li, X. Chen, D. Zou, B. Gao, and W. Teng, "Conformal and low-rank sparse representation for image restoration," in *IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, pp. 235–243.

[101] R. Yin, T. Gao, Y. M. Lu, and I. Daubechies, "A tale of two bases: Local-nonlocal regularization on image patches with convolution framelets," *SIAM Journal on Imaging Sciences*, vol. 10, no. 2, pp. 711–750, 2017.

[102] S. Osher, Z. Shi, and W. Zhu, "Low dimensional manifold model for image processing," *SIAM Journal on Imaging Sciences*, vol. 10, no. 4, pp. 1669–1690, 2017.

[103] S. Roth and M. J. Black, "Fields of experts," *International Journal of Computer Vision*, vol. 82, no. 2, p. 205, 2009.

[104] D. Zoran and Y. Weiss, "From learning models of natural image patches to whole image restoration," in *IEEE International Conference on Computer Vision (ICCV).* IEEE, 2011, pp. 479–486.

[105] J. Xu, L. Zhang, W. Zuo, D. Zhang, and X. Feng, "Patch group based nonlocal self-similarity prior learning for image denoising," in *IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 244–252.

[106] H. C. Burger, C. J. Schuler, and S. Harmeling, "Image denoising: Can plain neural networks compete with bm3d?" in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* IEEE, 2012, pp. 2392–2399.

[107] U. Schmidt and S. Roth, "Shrinkage fields for effective image restoration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2774–2781.

[108] Y. Chen and T. Pock, "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1256–1272, 2017.

[109] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142–3155, 2017.

[110] D. Liu, B. Wen, Y. Fan, C. C. Loy, and T. S. Huang, "Non-local recurrent network for image restoration," arXiv preprint arXiv:1806.02919, 2018.

[111] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," in *Annual Conference on Computer Graphics and Interactive Techniques.* ACM Press/Addison-Wesley Publishing Co., 2000, pp. 417–424.

[112] C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, and J. Verdera, "Filling-in by joint interpolation of vector fields and gray levels," *IEEE Transactions on Image Processing*, vol. 10, no. 8, pp. 1200–1211, 2001.

[113] A. Levin, A. Zomet, and Y. Weiss, “Learning how to inpaint from global image statistics,” in *IEEE International Conference on Computer Vision (ICCV)*, 2003, p. 305.

[114] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing.* New York: Springer, 2010.

[115] O. G. Guleryuz, “Nonlinear approximation based image recovery using adaptive sparse reconstructions and iterated denoising-part ii: adaptive algorithms,” *IEEE Transactions on Image Processing*, vol. 15, no. 3, pp. 555–571, 2006.

[116] Z. Xu and J. Sun, “Image inpainting by patch propagation using patch sparsity,” *IEEE Transactions on Image Processing*, vol. 19, no. 5, pp. 1153–1165, 2010.

[117] X. Li, “Image recovery via hybrid sparse representations: A deterministic annealing approach,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 5, pp. 953–962, 2011.

[118] K. H. Jin and J. C. Ye, “Annihilating filter-based low-rank hankel matrix approach for image inpainting,” *IEEE Trans. Image Proc.*, vol. 24, no. 11, pp. 3498–3511, 2015.

[119] Z. Zha, X. Yuan, B. Wen, J. Zhou, and C. Zhu, “Joint patch-group based sparse representation for image inpainting,” in *Asian Conference on Machine Learning*, 2018, pp. 145–160.

[120] C. Guillemot and O. Le Meur, “Image inpainting: Overview and recent advances,” *IEEE signal processing magazine*, vol. 31, no. 1, pp. 127–144, 2014.

[121] S. Geethanath, R. Reddy, A. S. Konar, S. Imam, R. Sundaresan, and R. Venkatesan, “Compressed sensing MRI: a review,” *Critical Reviews^{TM} in Biomedical Engineering*, vol. 41, no. 3, 2013.

[122] M. Lustig, D. L. Donoho, J. M. Santos, and J. M. Pauly, “Compressed sensing MRI,” *IEEE signal processing magazine*, vol. 25, no. 2, pp. 72–82, 2008.

[123] O. N. Jaspan, R. Fleysher, and M. L. Lipton, “Compressed sensing MRI: a review of the clinical literature,” *The British journal of radiology*, vol. 88, no. 1056, p. 20150487, 2015.

[124] Y. Yang, J. Sun, H. Li, and Z. Xu, “Deep ADMM-Net for compressive sensing MRI,” in *Advances in Neural Information Processing Systems*, 2016, pp. 10–18.

[125] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein et al., "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[126] H. Ji, C. Liu, Z. Shen, and Y. Xu, "Robust video denoising using low rank matrix completion," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2010, pp. 1791–1798.

[127] B. Wen, Y. Li, L. Pfister, and Y. Bresler, "Joint adaptive sparsity and low-rankness on the fly: an online tensor reconstruction scheme for video denoising," in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 241–250.

[128] T. Zhou and D. Tao, "Godec: Randomized low-rank & sparse matrix decomposition in noisy case," in *International Conference on Machine Learning (ICML)*. Omnipress, 2011.

[129] E. Liberty, F. Woolfe, P.-G. Martinsson, V. Rokhlin, and M. Tygert, "Randomized algorithms for the low-rank approximation of matrices," *Proceedings of the National Academy of Sciences*, vol. 104, no. 51, pp. 20 167–20 172, 2007.

[130] Y.-S. Chen, Y.-P. Hung, and C.-S. Fuh, "Fast block matching algorithm based on the winner-update strategy," *IEEE Transactions on Image Processing*, vol. 10, no. 8, pp. 1212–1222, 2001.

[131] Kodak, "The Kodak lossless true color image suite," available at: http://r0k.us/graphics/kodak/.

[132] X. Qu, "PBDWS code," available at: http://www.quxiaobo.org/.

[133] B. Kaltenbacher, A. Neubauer, and O. Scherzer, *Iterative Regularization Methods for Nonlinear Ill-posed Problems.* Walter de Gruyter, 2008, vol. 6.

[134] J. Xu, L. Zhang, D. Zhang, and X. Feng, "Multi-channel weighted nuclear norm minimization for real color image denoising," in *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[135] Y. Romano and M. Elad, "Boosting of image denoising algorithms," *SIAM Journal on Imaging Sciences*, vol. 8, no. 2, pp. 1187–1219, 2015.

[136] W. Zuo, L. Zhang, C. Song, and D. Zhang, "Texture enhanced image denoising via gradient histogram preservation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. ieee, 2013, pp. 1203–1210.

[137] W. Dong, L. Zhang, G. Shi, and X. Li, “Nonlocally centralized sparse representation for image restoration,” *IEEE Transactions on Image Processing*, vol. 22, no. 4, pp. 1620–1630, 2013.

[138] J. C. Ye, Y. Han, and E. Cha, “Deep convolutional framelets: A general deep learning framework for inverse problems,” *SIAM Journal on Imaging Sciences*, vol. 11, no. 2, pp. 991–1048, 2018.

[139] W. Bae, J. Yoo, and J. C. Ye, “Beyond deep residual learning for image restoration: Persistent homology-guided manifold simplification,” in *IEEE Conf. Comp. Vision and Pattern Recog (CVPR) Workshops*, 2017, pp. 145–153.

[140] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, “Color image denoising via sparse 3d collaborative filtering with grouping constraint in luminance-chrominance space,” in *IEEE International Conference on Image Processing (ICIP)*, vol. 1. IEEE, 2007, pp. I–313.

[141] H. Takeda, S. Farsiu, and P. Milanfar, “Kernel regression for image processing and reconstruction,” *IEEE Transactions on Image Processing*, vol. 16, no. 2, pp. 349–366, 2007.

[142] K. Lee, Y. Li, M. Junge, and Y. Bresler, “Blind recovery of sparse signals from subsampled convolution,” *IEEE Transactions on Information Theory*, vol. 63, no. 2, pp. 802 – 821, 2017.

[143] Y. Li, K. Lee, and B. Yoram, “Blind gain and phase calibration for low-dimensional or sparse signal sensing via power iteration,” in *Int. Conf. Sampling Theory and Applications (SampTA)*, 2017.

[144] A. Buades, J.-L. Lisani, and M. Miladinovic, “Patch-based video denoising with optical flow estimation,” *IEEE Transactions on Image Processing*, vol. 25, no. 6, pp. 2573–2586, 2016.

[145] 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 *IEEE International Conference on Computer Vision (ICCV)*, 12 2015, pp. 442–449.

[146] Z. Zha, X. Liu, X. Huang, H. Shi, Y. Xu, Q. Wang, L. Tang, and X. Zhang, “Analyzing the group sparsity based on the rank minimization methods,” in *IEEE International Conference on Multimedia and Expo (ICME)*, 2017, pp. 883–888.

[147] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, “Bm3d image denoising with shape-adaptive principal component analysis,” in *Signal Processing with Adaptive Sparse Structured Representations (SPARS)*, 2009.

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

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