Learning Filter Bank Sparsifying Transforms

Luke Pfister, *Student Member, IEEE,* Yoram Bresler, *Fellow, IEEE*

***Abstract*—Data is said to follow the transform (or analysis) sparsity model if it becomes sparse when acted on by a linear operator called a sparsifying transform. Several algorithms have been designed to learn such a transform directly from data, and data-adaptive sparsifying transforms have demonstrated excellent performance in signal restoration tasks. Sparsifying transforms are typically learned using small sub-regions of data called patches, but these algorithms often ignore redundant information shared between neighboring patches.**

We show that many existing transform and analysis sparse representations can be viewed as filter banks, thus linking the local properties of patch-based model to the global properties of a convolutional model. We propose a new transform learning framework where the sparsifying transform is an undecimated perfect reconstruction filter bank. Unlike previous transform learning algorithms, the filter length can be chosen independently of the number of filter bank channels. Numerical results indicate filter bank sparsifying transforms outperform existing patch-based transform learning for image denoising while benefiting from additional flexibility in the design process.

***Index Terms*—sparsifying transform, analysis model, analysis operator learning, sparse representations, perfect reconstruction, filter bank, convolutional analysis operators.**

I. Introduction

Countless problems, from statistical inference to geological exploration, can be stated as the recovery of high-quality data from incomplete and/or corrupted linear measurements. Often, recovery is possible only if a model of the desired signal is used to regularize the recovery problem.

A powerful example of such a signal model is the *sparse representation*, wherein the signal of interest admits a representation with few nonzero coefficients. Sparse representations have traditionally been hand-designed for optimal properties on a mathematical signal class, such as the coefficient decay properties of a cartoon-like signal under a curvelet representation [1]. Unfortunately, these signal classes do not include the complicated and textured signals common in applications; further, it is difficult to design optimal representations for high dimensional data. In light of these challenges, methods to *learn* a sparse representation, either from representative training data or directly from corrupted data, have become attractive.

We focus on a particular type of sparse representation, called *transform sparsity*, in which the signal $x \in \mathbb{R}^N$ satisfies $Wx = z+\eta$. The matrix $W \in \mathbb{R}^{K\times N}$ is called a *sparsifying transform* and earns its name as $z \in \mathbb{R}^K$ is sparse and $\|\eta\|_2$ is small [2]. Of course, a W that is uniformly zero satisfies this definition but provides no insight into the transformed signal. Several algorithms have been proposed to learn a sparsifying transform from data, and each must contend with this type of degenerate solution. The most common approach is to ensure that W is left invertible, so that Wx is uniformly zero if and only if x is uniformly zero. Such a matrix is a *frame* for $\mathbb{R}^n$.

In principle, we can learn a sparse representation for any data represented as a vector, including data from genomic experiments or text documents, yet most research has focused on learning models for spatio-temporal data such as images. With these signals it is common to learn a model for smaller, possibly overlapping, blocks of the data called *patches*. We refer to this type of model as a *patch-based* model, while we call a model learned directly at the image level an *image-based* model. Patch-based models tend to have fewer parameters than an unstructured image-based model, leading to lower computational cost and reduced risk of overfitting. In addition, an image contains many overlapping patches, and thus a model can be learned from a single noisy image [3].

Patch-based models are not without drawbacks. Any patch-based W learned using the usual frame constraints must have at least as many rows as there are elements in a patch, *i.e.* W must be square or tall. This limits practical patch sizes as W must be small to benefit from a patch-based model.

If our ultimate goal is image reconstruction, we must be mindful of the connection between extracted patches and the original image. Requiring W to be a frame for patches ignores this relationship and instead requires that each patch can be independently recovered. Yet, neighboring patches can be highly correlated- leading us to wonder if the patch-based frame condition is too strict. This leads to the question at the heart of this paper: *Can we learn a sparsifying transform that forms a frame for images, but not for patches, while retaining the computational efficiency of a patch-based model?*

In this paper, we show that existing sparsifying transform learning algorithms can be viewed as learning perfect reconstruction filter banks. This perspective leads to a new approach to learn a sparsifying transform that forms a frame over the space of images, and is structured as an undecimated, multidimensional filter bank. We call this structure a *filter bank sparsifying transform*. We keep the efficiency of a patch-based model by parameterizing the filter bank in terms of a small matrix W. In contrast to existing transform learning algorithms, our approach can learn a transform corresponding to a tall, square, or fat W. Our learned model outperforms earlier transform learning algorithms while maintaining low cost of learning the filter bank and the processing of data by it. Although we restrict our attention to 2D images, our technique is applicable to any data amenable to patch-based methods, such as 3D imaging data.

The rest of the paper is organized as follows. In Section II we review previous work on transform learning, analysis learning, and the relationship between patch-based and image-based models. In Section III we develop the connection between perfect reconstruction filter banks and patch-based transform

This work was supported in part by the National Science Foundation (NSF) under Grants CCF 1018660 and CCF-1320953.

learning algorithms. We propose our filter bank learning algorithm in Section IV, describe denoising algorithms in Section V, and present numerical results in Section VI. In Section VII we compare our learning framework to the current crop of deep learning inspired approaches, and conclude in Section VIII.

II. Preliminaries

A. Notation

Matrices are written as capital letters, while general linear operators are denoted by script capital letters such as $\mathcal{A}$. Column vectors are written as lower case letters. The i-th component of a vector x is x_i. The i,j-th element of a matrix A is A_{ij}. We write the j-th column of A as $A_{:,j}$, and $A_{i,:}$ is the column vector corresponding to the transpose of the i-th row. The transpose and Hermitian transpose are A^T and A^*, respectively. Similarly, $\mathcal{A}^*$ is the adjoint of the linear operator $\mathcal{A}$. The $n \times n$ identity matrix is I_n. The n-dimensional vectors of ones and zeros are written as $\mathbf{1}_n$ and $\mathbf{0}_n$, respectively. For $x \in \mathbb{R}^N$, the diagonal matrix $\operatorname{ddiag}(x) \in \mathbb{R}^{N \times N}$ has the entries of x along its diagonal. The convolution of signals x and y is written $x * y$. For vectors $x, y \in \mathbb{R}^N$, the Euclidean inner product is $\langle x, y \rangle = \sum_{i=1}^N x_i y_i$ and the Euclidean norm is written $\|x\|_2$. The vector space of matrices $\mathbb{R}^{M \times N}$ is equipped with the inner product $\langle X, Y \rangle = \operatorname{trace}\left(X^T Y\right)$; the Frobenius norm is written $\|X\|_F$. When necessary, we indicate the vector space on which the inner product is defined; *e.g.* $\langle x, y \rangle_{\mathbb{R}^N}$.

B. Transform Sparsity

Recall that a signal $x \in \mathbb{R}^N$ satisfies the transform sparsity model if there is a matrix $W \in \mathbb{R}^{K \times N}$ such that $Wx = z + \eta$, where z is sparse and $\|\eta\|_2$ is small. The matrix W is called a *sparsifying transform* and the vector z is a *transform sparse code*. Given a signal x and sparsifying transform W, the *transform sparse coding* problem is

$$\arg\min_z \frac{1}{2}\|Wx - z\|_2^2 + \nu\psi(z) \tag{1}$$

for a sparsity promoting functional ψ. Exact s-sparsity can be enforced by selecting ψ to be the indicator function over the set of s-sparse vectors. We recognize (1) as the evaluation of the proximal operator of ψ, defined as

$$\operatorname{prox}_{\psi\nu}(t) = \arg\min_z \frac{1}{2}\|t - z\|_2^2 + \nu\psi(z), \tag{2}$$

at the point $t = Wx$. Transform sparse coding is often cheap as the proximal mapping of many sparsity penalties can be computed cheaply and in closed form. For instance, when $\psi(z) = \|z\|_0$, then $z = \operatorname{prox}_{\psi\nu}(Wx)$ is computed by setting $z_i = [Wx]_i$ whenever $|[Wx]_i|^2 > \nu^2$, and setting $z_i = 0$ otherwise. This operation is called *hard thresholding*.

Several methods have been proposed to learn a sparsifying transform from data, including algorithms to learn square transforms [2], orthonormal transforms [4], structured transforms [5], and overcomplete transforms consisting of a stack of square transforms [6], [7]. Degenerate solutions are prevented by requiring the rows of the learned transform to constitute a well-conditioned frame. In the square case, the transform learning problem can be written

$$\min_{W,Z} \frac{1}{2}\|WX - Z\|_F^2 + \psi(Z) + \frac{1}{2}\|W\|_F^2 - \mu \log|\det W| \tag{3}$$

where X is a matrix whose columns contain training signals and ψ is a sparsity-promoting functional. The first term ensures that the transformed data, WX, is close to the matrix Z, while the second term ensures that Z is sparse. The remaining terms ensure that W is full rank and well-conditioned [2]. Square sparsifying transforms have demonstrated excellent performance in image denoising, magnetic resonance imaging, and computed tomographic reconstruction [8]–[11].

C. Analysis sparsity

Closely related to transform sparsity is the *analysis model*. A signal $x \in \mathbb{R}^N$ satisfies the analysis model if there is a matrix $\Omega \in \mathbb{R}^{K \times N}$, called an analysis operator, such that $\Omega x = z$ is sparse. The analysis model follows by restricting $\eta = \mathbf{0}_K$ in the transform sparsity model.

A typical analysis operator learning algorithm is of the form

$$\min_{\Omega} \psi(\Omega X) + J(\Omega) \tag{4}$$

where X are training signals, ψ is a sparsity promoting functional, and J is a regularizer to ensure the learned Ω is informative. In the Analysis K-SVD algorithm, the rows of Ω are constrained to have unit norm, but frame constraints are the most common [12]. Yaghoobi *et al.* observed that learning an analysis operator with $q > K$ rows while using a tight frame constraint resulted in operators consisting of a full rank matrix appended with $q - K$ uniformly zero rows. They instead proposed a uniformly-normalized tight frame (UNTF) constraint, wherein the rows of Ω have equal ℓ_2 norm and constitute a tight frame [13]–[15].

Hawe *et al.* utilized a similar set of constraints in their GeOmetric Analysis operator Learning (GOAL) framework [16]. They constrained the learned Ω to the set of full column rank matrices with unit-norm rows and solved the optimization problem using a manifold descent algorithm.

Transform and analysis sparsity are closely linked. Indeed, using a variable splitting approach (*e.g.* $Z = \Omega X$) to solve (4) leads to algorithms that are similar to transform learning algorithms [13]–[15]. The relationships between the transform model, analysis model, and noisy variations of the analysis model have been explored [2]. We focus on the transform model because the proximal interpretation of sparse coding fits nicely within a filter bank interpretation (see Section III-F).

D. From patch-based to image-based models

A link between patch-based and image-based models can be made using the *Field of Experts* (FoE) model proposed by Roth and Black [17]. They modeled the prior probability of an image as a Markov Random Field (MRF) with overlapping "cliques" of pixels that serve as image patches. Using the so-called Product of Experts framework, a model for the prior probability of an image patch is expressed as a sparsity-inducing potential function applied to the inner products

between multiple 'filters' and the image patch. A prior for the entire image is formed by taking the product of the prior for each patch and normalizing.

Continuing in this direction, Chen *et. al* proposed a method to learn an image-based analysis operator using the FoE framework using a bi-level optimization formulation [18]. This approach was recently extended into an iterated filter bank structure called a *Trainable Nonlinear Reaction Diffusion* (TNRD) network [19]. Each stage of the TNRD network consists of a set of analysis filters, a channelwise nonlinearity, the adjoint filters, and a direct feed-forward path. The filters, nonlinearity, and feed-forward mixing weights are trained in a supervised fashion. The TNRD approach has demonstrated state of the art performance on image denoising tasks.

The TNRD and FoE algorithms are supervised and use the filter bank structure only as a computational tool. In contrast, our approach is unsupervised and uses the theory of perfect reconstruction filter banks to regularize the learning problem.

Cai *et al*. developed an analysis operator learning method based on a filter bank interpretation of the operator [20]. The operator can be thought to act on images, rather than patches. Their approach is fundamentally the same as learning a square, orthonormal, patch-based sparsifying transform [4]. In contrast, our approach does not have these restrictions: we learn a filter bank that is a frame for images, and corresponds to a tall, fat, or square patch-based transform.

These methods fall under the analysis paradigm. In Section III we show that patch-based analysis models naturally induce a image-based model. In contrast, synthesis patch-based models do not directly lead to an image based model. Figueiredo studied this dichotomy between patch-based synthesis and analysis priors and proposed a method for image-based denoising using patch-based synthesis methods [21].

Image-based modeling using synthesis sparsity can be implemented in an entirely different manner by imposing shift-invariance properties on the synthesis dictionary [22]–[25]. Briefly, the goal of *convolutional dictionary learning (CDL)* is to find a set of filters, $\{d_i\}_{i=1}^{N_c}$, such that the training signals can be modeled as $y = \sum_{i=1}^{N_c} d_i * a_i$, where the a_i are sparse. Here, y is an image, not a patch. The filters d_i are required to have compact support so as to limit the number of free parameters in the learning problem. The desired convolutional structure can be imposed by writing the convolution in the frequency domain, but care must be taken to ensure that the d_i remain compactly supported. For further details, see the recent reviews [24], [25].

Finally, Muramatsu *et al*. proposed an approach for the design of multidimensional, multirate, nonseparable, overlapped linear phase perfect reconstruction synthesis filter banks [26]–[28]. We will refer to a dictionary designed in this manner as a (synthesis) Non-Separable Oversampled Lapped Transforms, or NSOLT. Despite using the synthesis sparsity model, the NSOLT design problem shares more in common with our proposed filter bank sparsifying transform learning than the usual CDL problem. We further discuss the NSOLT structure in Section III-D after the language of polyphase matrices has been established. Differences between NSOLT and our proposed method are discussed in Section IV-D.

In the next section, we show that patch-based analysis and transform models, in contrast to synthesis models, are naturally endowed with a convolutional structure.

III. From Patch-based Transforms to Filter Banks

In this section, we illustrate the connections between patch-based sparsifying transforms and multirate finite impulse response (FIR) filter banks. The link between patch-based analysis methods and convolution has been previously established, but used only as a computational tool [17], [18], [20], [29], [30]. Our goal is to illustrate how and when the boundary conditions, patch stride, and a patch-based sparsifying transform combine to form a frame over the space of images.

A. Frames, Patches, and Images

A set of vectors $\{\omega_i\}_{i=1}^{M}$ in $\mathbb{R}^m$ is a *frame* for $\mathbb{R}^m$ if there exists $0 < A \leq B < \infty$ such that

$$A\|y\|_2^2 \leq \sum_{j=1}^{M} |\langle y, \omega_i \rangle|^2 \leq B\|y\|_2^2 \tag{5}$$

for all $y \in \mathbb{R}^m$ [31]. Equivalently, the matrix $\Omega \in \mathbb{R}^{M\times m}$, with i-th row given by ω_i, is left invertible. The frame bounds A and B correspond to the smallest and largest eigenvalues of $\Omega^T\Omega$, respectively. The frame is *tight* if $A = B$, and in this case $\Omega^T\Omega = AI_n$. The condition number of the frame is the ratio of the frame bounds, B/A. The ω_i are called *frame vectors*, and the matrix Ω implements a *frame expansion*.

Consider a patch-based model using $K \times K$ (vectorized) patches from an $N \times N$ image. We call $\mathbb{R}^{K^2}$ the *space of patches* and $\mathbb{R}^{N\times N}$ the *space of images*. In this setting, transform learning algorithms find a $W \in \mathbb{R}^{N_c\times K^2}$ with rows that form a frame for the space of patches [2], [4]–[7].

We can extend this W to a frame over the space of images as follows. Suppose the rows of W form a frame with frame bounds $0 < A \leq B$. Let $\mathcal{R}_j : \mathbb{R}^{N\times N} \to \mathbb{R}^{K^2}$ be the linear operator that extracts and vectorizes the j-th patch from the image, and suppose there are M such patches. So long as each pixel in the image is contained in at least one patch, we have

$$\|x\|_F^2 \leq \sum_{j=1}^{M} \|\mathcal{R}_j x\|_2^2 \leq M\|x\|_F^2 \tag{6}$$

for all $x \in \mathbb{R}^{N\times N}$. Letting $w^i = W_{i,:}$ denote the i-th row of W, we have for all $x \in \mathbb{R}^{N\times N}$

$$\sum_{j=1}^{M}\sum_{i=1}^{N_c} \left|\langle w^i, \mathcal{R}_j x\rangle\right|^2 \geq \sum_{j=1}^{M} A\|\mathcal{R}_j x\|_2^2 \geq A\|x\|_F^2, \tag{7}$$

$$\sum_{j=1}^{M}\sum_{i=1}^{N_c} \left|\langle w^i, \mathcal{R}_j x\rangle\right|^2 \leq B\sum_{j=1}^{M} \|\mathcal{R}_j x\|_2^2 \leq MB\|x\|_F^2. \tag{8}$$

Because $\langle w^i, \mathcal{R}_j x\rangle_{\mathbb{R}^{K^2}} = \langle \mathcal{R}_j^* w^i, x\rangle_{\mathbb{R}^{N\times N}}$, it follows that the collection $\left\{\mathcal{R}_j^* w^i\right\}_{i=1,j=1}^{N_c,M}$ forms a frame for the space of images with bounds $0 < A \leq MB$. Thus, every frame over the spaces of patches corresponds to a frame over the space of images. Next, our goal is to determine when the patch extraction operators and the transform, W, form a frame for the space of images but *not* the space of patches.

B. Patch-based Sparsifying Transforms as Filter Banks

We consider two ways to represent applying the transform $W \in \mathbb{R}^{N_c \times K^2}$ to the image $x \in \mathbb{R}^{N \times N}$. The usual approach is to form the *patch matrix* $X \in \mathbb{R}^{K^2 \times M^2}$ with j-th column $\mathcal{R}_j x$, as illustrated in Fig. 1a. We call the spacing between adjacent extracted patches the *stride* and denote it by s. The extracted patches overlap when $s < K$ and are disjoint otherwise. We assume the stride is the same in both horizontal and vertical directions and evenly divides N. The number of patches, M^2, depends on the boundary conditions and patch stride; *e.g.* $M^2 = N^2/s^2$ if periodic boundary conditions are used. The patch matrix for the transformed image is $WX \in \mathbb{R}^{N_c \times M^2}$.

Our second approach eliminates patches and their vectorization by viewing WX as the output of a multirate filter bank with 2D FIR filters and input x. Let

$$\mathcal{H} : \mathbb{R}^{N \times N} \to \mathbb{R}^{N_c} \otimes \mathbb{R}^{M \times M} \tag{9}$$

be this filter bank operator, which transforms an $N \times N$ image into a three-dimensional array formed as a stack of N_c output images, each of size $M \times M$.

We build $\mathcal{H}$ from a collection of downsampled convolution operators. For $i = 1, 2, \dots N_c$, we define the i-th channel operator $\mathcal{H}_i : \mathbb{R}^{N \times N} \to \mathbb{R}^{M \times M}$ such that $[\mathcal{H}_i x]_{a,b} = [h_i * x]_{sa,sb}$. The stride s dictates the downsampling level, and the patch extraction boundary conditions determine the convolution boundary conditions; in particular, if periodic boundary conditions are used, then $\mathcal{H}_i$ implements cyclic convolution. The impulse response h_i is obtained from the i-th row of W as $\mathcal{R}_1^* w^i$. This matrix consists of a $K \times K$ submatrix embedded into the upper-left corner of an $N \times N$ matrix of zeros as illustrated in Fig. 1b. [1]

Finally, we construct $\mathcal{H}$ by "stacking" the channel operators: $\mathcal{H} = \sum_{i=1}^{N_c} e_i \otimes \mathcal{H}_i$, where e_i is the i-th standard basis vector in $\mathbb{R}^{N_c}$ and $\otimes$ denotes the Kronecker (or tensor) product. With this definition, $y = \mathcal{H}x = \sum e_i \otimes \mathcal{H}_i x = \sum e_i \otimes y_i$. The filter bank structure is illustrated in Fig. 2. We refer to $\mathcal{H}$ constructed in this form as a *filter bank sparsifying transform*. The following proposition links WX and $\mathcal{H}x$:

Proposition 1. *Let $X \in \mathbb{R}^{K^2 \times M^2}$ be a patch matrix for image X, and let $W \in \mathbb{R}^{N_c \times K^2}$. The rows of WX can obtained by passing x through the N_c channel, 2D FIR multirate analysis filter bank $\mathcal{H}$ and vectorizing the channel outputs.*

A proof for 1D signals is given in Appendix A. The proof for 2D is similar, using vector indices.

Proposition 1 connects the local, patch-extraction process and the matrix W to a filter bank operator that acts on images. Unlike convolutional synthesis models, patch-based analysis operators naturally have a convolutional structure.

Next, we investigate connections between the frame properties of $\mathcal{H}$ and the combination of W and the patch extraction scheme. Our primary tool is the polyphase representation of filter banks [32], [33]. Consider the image x as a 2D

[1] In the case of cyclic convolution, $\mathcal{R}_1^* w^i$ is exactly the impulse response of the i-th channel, but only the nonzero portion of $\mathcal{R}_1^* w^i$ is the impulse response when using linear convolution. In a slight abuse of terminology, we call $\mathcal{R}_1^* w^i$ the impulse response in both instances.

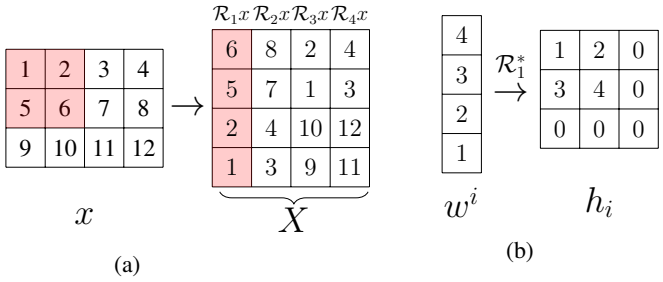


Fig. 1: (a) Construction of the patch matrix $X \in \mathbb{R}^{4\times 4}$ from 2×2 patches of $x \in \mathbb{R}^{3\times 4}$ using periodic boundary conditions and a stride of 2. Note that the vectorized patch is "flipped" from the natural ordering; *i.e.*, the top-left pixel in the patch is the final element of the vector. (b) Obtaining the impulse response h_i from w^i.

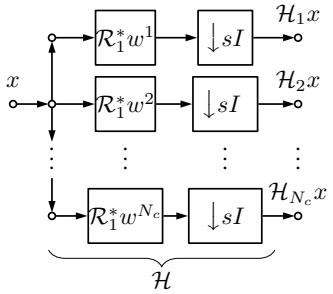


Fig. 2: Analysis filter bank $\mathcal{H}$ generated by a sparsifying transform W and stride length s.

sequence $x[n_1, n_2]$ for $0 \le n_1, n_2 \le N-1$. The z-transform of the (a,b)-th polyphase component of x is z-transform $\hat{X}_{a,b}(\mathbf{z}) = \sum_{n_1,n_2} x[n_1 \cdot s + a, n_2 \cdot s + b] z_1^{-n_1} z_2^{-n_2}$ of the shifted and downsampled sequence, where $\mathbf{z} = [z_1, z_2] \in \mathbb{C}^2$ and $0 \le a, b \le s-1$. The polyphase representation for the sequence x is formed by stacking the polyphase components in lexicographical order into a single $\hat{X}(\mathbf{z}) = [X_{0,0}(\mathbf{z}), \dots, X_{s-1,s-1}(\mathbf{z})]^T \in \mathbb{C}^{s^2}$.

The filter bank $\mathcal{H}$ has a polyphase matrix $\hat{H}(\mathbf{z}) \in \mathbb{C}^{Nc \times s^2}$ formed by stacking the polyphase representations of each channel into a row, and stacking the N_c rows. Explicitly,

$$\hat{H}(\mathbf{z}) = \begin{bmatrix} \hat{H}^0_{0,0}(\mathbf{z}) & \hat{H}^0_{0,1}(\mathbf{z}) & \dots & \hat{H}^0_{s,s}(\mathbf{z}) \\ \hat{H}^1_{0,0}(\mathbf{z}) & \hat{H}^1_{0,1}(\mathbf{z}) & \dots & \hat{H}^1_{s,s}(\mathbf{z}) \\ \vdots & \vdots & \ddots & \vdots \\ \hat{H}^{N_c-1}_{0,0}(\mathbf{z}) & \hat{H}^{N_c-1}_{0,1}(\mathbf{z}) & \dots & \hat{H}^{N_c-1}_{s,s}(\mathbf{z}) \end{bmatrix} \tag{10}$$

where $\hat{H}^i_{a,b}(\mathbf{z})$ is the (a,b)-th polyphase component of the i-th filter in $\mathcal{H}$. The entries of $\hat{H}(\mathbf{z})$ are, in general, bi-variate polynomials in $\mathbf{z} = [z_1, z_2]$. The output of the filter bank, $y = \mathcal{H}x$, can be written in the polyphase domain as $\hat{Y}(\mathbf{z}) = \hat{H}(\mathbf{z})\hat{X}(\mathbf{z})$, where the i-th element of the vector $\hat{Y}(\mathbf{z})$ is the z-transform of the i-th output channel.

Many important properties of $\mathcal{H}$ are tied to its polyphase matrix. An analysis filter bank $\mathcal{H}$ is said to be *perfect reconstruction* (PR) if there is a (synthesis) filter bank $\mathcal{G}$ such that $\mathcal{G}\mathcal{H} = \mathcal{I}$, or in the polyphase domain $\hat{G}(\mathbf{z})\hat{H}(\mathbf{z}) = I$. A

filter bank is PR if and only if $\hat{H}(\mathbf{z})$ is full column rank on the unit circle [34]. A filter bank is said to be *orthonormal* if $\mathcal{H}^*\mathcal{H} = \mathcal{I}$, that is, the filter bank with analysis section $\mathcal{H}$ and synthesis section $\mathcal{H}^*$ is an identity mapping on $\mathbb{R}^{N\times N}$. In the polyphase domain, this corresponds to

$$\hat{H}^*(\mathbf{z}^{-1})\hat{H}(\mathbf{z}) = I, \tag{11}$$

where the star superscript denotes Hermitian transpose and $\mathbf{z}^{-1} = [z_1^{-1}, z_2^{-1}]$ [32], [35]. A matrix satisfying (11) is *paraunitary*, and $\hat{H}^*(\mathbf{z}^{-1})$ is the *paraconjugate* of $\hat{H}(\mathbf{z})$.

A PR filter bank implements a frame expansion over the space of images, and an orthonormal filter bank implements a tight frame expansion over the same space [36], [37]. The frame vectors are the collection of the shifts of the impulse responses of each channel, and are precisely the collection $\left\{\mathcal{R}_j^* w^i\right\}$ discussed in Section III-A. The link between patch-based transforms and filter banks does not directly lead to new transform learning algorithms, as the characterization and construction of multidimensional PR filter banks is hard due to the lack of a multidimensional spectral factorization theorem [33], [38]–[40].

Next, we study we illustrate the connections between patch-based sparsifying transforms and perfect reconstruction filter banks as a function of the stride length. We show that in certain cases the PR condition takes on a simple form.

C. Perfect Recovery: Non-overlapping patches

Consider $s = K$, so that the extracted patches do not overlap. Applying the sparsifying transform W to non-overlapping patches is an instance of a *block transform* [41]. Block transforms are found throughout in signal processing applications; for example, the JPEG compression algorithm. Block transforms are viewed as a decimated FIR filter bank with uniform downsampling by K in each dimension, consistent with Proposition 1.

It is informative to view patch-based transform learning algorithms through the lens of block transformations. Because we downsample by K in each dimension, and the filters are of size $K \times K$, the polyphase matrix $\hat{H}(\mathbf{z})$ is constant in (independent of) $\mathbf{z}$ and is equal to W. This gives a direct connection between the PR properties of $\mathcal{H}$, which acts on images, and W, which acts on patches. Patch-based transform learning algorithms enforce either invertibility of W (in the square case) or invertibility of W^TW (in the overcomplete case), and thus $\mathcal{H}$ is PR. If W is orthonormal, so too is $\mathcal{H}$.

D. Perfect Recovery: Partially overlapping patches

Next, consider patches extracted using a stride $1 < s < K$. While WX is no longer a block transformation, it is related to a *lapped transformation* [41]. Lapped transforms aim to reduce artifacts that arise from processing each block (patch) independently by allowing for overlap between neighboring blocks. Many lapped transforms, such as the Lapped Orthogonal Transform, the Extended Lapped Transform, and the Generalized Lapped Orthogonal Transform [42], enjoy both the PR property and efficient implementation.

Lapped transforms were designed for signal coding applications. The number of channels in the filter bank is decreased as the degree of overlap increases, so that the number of transform coefficients using a lapped transform is the same as using a non-lapped transform. While redundancy may be undesirable in certain coding applications, it aids the solution of inverse problems by allowing for richer and more robust signal models [43]. We allow the stride length to decrease while keeping the number of channels fixed, and interpret WX as a "generalized" lapped transform. When the stride is less than K, W no longer corresponds to the polyphase matrix of the filter bank $\mathcal{H}$; instead, the polyphase matrix $\hat{H}(\mathbf{z})$ will contain high-order, 2D polynomials. While the filter bank may still be PR, the PR property is not directly implied by invertibility of W.

We can learn a PR generalized lapped transform by enforcing the more restrictive PR conditions for non-overlapping patches, that is, invertibilty of W^TW. When $s = 1$, this technique is equivalent to cycle spinning, which was developed to add shift-invariance to decimated wavelet transforms [44]. When $1 < s < K$, we can interpret $\mathcal{H}x$ as cycle spinning without all possible shifts.

Muramatsu *et al*. proposed a different method to learn a PR *synthesis* non-separable oversampled lapped transform (NSOLT) [26]–[28]. The filter bank is designed such that each channel consists of linear phase filters; thus each channel consists of either symmetric or anti-symmetric filters. The filter bank is parameterized by a certain lattice structure that implicitly ensures the NSOLT implements a tight-frame expansion and is thus PR. This lattice structure leads to a particular factorization of the (paraunitary) polyphase matrix; in two dimensions, we have $\hat{\mathbf{H}}(\mathbf{z}) = \mathbf{G}_1(z_1)\mathbf{G}_2(z_2)\mathbf{H}_0$, where $\mathbf{G}_i(z_i)$ is a univariate polynomial matrix of specified order and $\mathbf{H}_0$ is constant in $\mathbf{z}$. Each of these matrices is further parameterized to lead to a tractable optimization problem; see [28] for details.

Patch-based sparsifying transforms and NSOLTs are both parameterized in terms of a polyphase matrix, and thus lead to filter banks with compactly supported filters. This is in stark contrast to the usual convolutional dictionary learning problems, where variable splitting methods are often used to obtain both a convolutional structure and compactly supported filters.

E. Perfect Recovery: Maximally overlapping patches

Finally, consider extracting maximally overlapping patches by setting $s = 1$. The resulting filter bank $\mathcal{H}$ is undecimated and the Gram operator $\mathcal{H}^*\mathcal{H}$ is shift invariant. As there is no downsampling, the polyphase representations of x and y are the z-transforms of the sequences x and y. The polyphase matrix of $\mathcal{H}$ is the column vector $\hat{H}(\mathbf{z}) = [\hat{H}_1(\mathbf{z}), \dots \hat{H}_{N_c}(\mathbf{z})]^T$ where $\hat{H}_i(\mathbf{z})$ is the z-transform of $h_i = \mathcal{R}_1^* w^i$.

An undecimated linear convolution filter bank is PR if and only if its filters have no common zeros on the unit circle; *i.e*., each frequency must pass through at least one channel of the filter bank [34]. When evaluated on the unit circle

the z-transform becomes the Discrete Time Fourier Transform (DTFT), defined for $h \in \mathbb{R}^{K\times K}$ as

$$\hat{H}(\boldsymbol{\omega}) = \sum_{n_1=0}^{K-1}\sum_{n_2=0}^{K-1} h[n_1,n_2]e^{-j\omega_1 n_1}e^{-j\omega_2 n_2} \tag{12}$$

where $\boldsymbol{\omega} = [\omega_1, \omega_2]$ with $\omega_1, \omega_2 \in [0, 2\pi)$. Now, the polyphase matrix is full rank on the unit circle if and only if

$$\varphi(\boldsymbol{\omega}) \triangleq \sum_{i=1}^{N_c} \left|\hat{H}_i(\boldsymbol{\omega})\right|^2 > 0 \quad \forall w_1, w_2 \in [0, 2\pi), \tag{13}$$

where $\varphi(\boldsymbol{\omega})$ is the DTFT of the impulse response of $\mathcal{H}^*\mathcal{H}$ and is an even, real, non-negative, 2D trigonometric polynomial with maximum component order $K-1$. Explicitly,

$$\varphi(\boldsymbol{\omega}) = \sum_{n_1=-K+1}^{K-1}\sum_{n_2=-K+1}^{K-1} \tilde{h}[n_1,n_2]\cos(\omega_1 n_1)\cos(\omega_2 n_2), \tag{14}$$

where the impulse response of $\tilde{h}$ is $\mathcal{H}^*\mathcal{H}$ is the sum of the channel-wise autocorrelations; that is,

$$\tilde{h}[n_1,n_2] = \sum_{i=1}^{N_c}\sum_{l_1=-\infty}^{\infty}\sum_{l_2=-\infty}^{\infty} h_i[l_1,l_2]h_i[l_1-n_1, l_2-n_2]. \tag{15}$$

Direct verification of the PR condition (13) is NP-Hard for $K \geq 2$, underlining the difficulty of multidimensional filter bank design [45], [46]. We sidestep the difficulty of working with (13) by developing the PR condition when image patches are extracted using periodic boundary conditions. The resulting filter bank implements *cyclic* convolution. Afterwards, we show that under certain conditions, the PR property of a cyclic convolution filter bank implies the PR property of a linear convolution filter bank constructed from the same filters.

1) Periodic Boundary Conditions / Cyclic Convolution: If image patches are extracted using periodic boundary conditions, the channel operators $\mathcal{H}_i : \mathbb{R}^{N\times N} \to \mathbb{R}^{N\times N}$ implement cyclic convolution and are diagonalized by the 2D Discrete Fourier Transform (DFT). Let $\mathcal{F}$ be the orthonormal 2D-DFT operator such that

$$(\mathcal{F}h_i)[\mathbf{k}] = N^{-1}\sum_{n_1=0}^{K-1}\sum_{n_2=0}^{K-1} h_i[n_1,n_2]e^{-j\frac{2\pi k_1 n_1}{N}}e^{-j\frac{2\pi k_2 n_2}{N}} \tag{16}$$

for $\mathbf{k} = [k_1, k_2]$ and $0 \leq k_1, k_2 < N$; that is, the length N 2D-DFT of the filter h_i padded with $N-K$ zeros in each dimension. Define $\mathcal{D}_i \in \mathbb{C}^{N\times N} \to \mathbb{C}^{N\times N}$ as the operator that multiplies pointwise by $\mathcal{F}h_i$: for $u \in \mathbb{C}^{N\times N}$, we have $(\mathcal{D}_i u)(\mathbf{k}) = (\mathcal{F}h_i)(\mathbf{k}) \cdot u(\mathbf{k})$. The cyclic convolution operator $\mathcal{H}_i$ has eigenvalue decomposition $\mathcal{F}^*\mathcal{D}_i\mathcal{F}$. We can use this channel-wise decomposition to find the spectrum of $\mathcal{H}^*\mathcal{H}$:

Lemma 1. *The N^2 eigenvalues of the undecimated cyclic analysis-synthesis filter bank $\mathcal{H}^*\mathcal{H}$ are given by $\sum_{i=1}^{N_c}|(\mathcal{F}h_i)[\mathbf{k}]|^2$ for $\mathbf{k} = [k_1, k_2]$ and $0 \leq k_1, k_2 < N$.*

Proof. We have

$$\mathcal{H}^*\mathcal{H} = \sum_{i=1}^{N_c}(e_i \otimes \mathcal{H}_i)^*(e_i \otimes \mathcal{H}_i) = \sum_{i=1}^{N_c}\mathcal{H}_i^*\mathcal{H}_i$$
$$= \mathcal{F}^*\left(\sum_{i=1}^{N_c}\mathcal{D}_i^*\mathcal{D}_i\right)\mathcal{F} = \mathcal{F}^*\mathcal{D}\mathcal{F}$$

where $(\mathcal{D}u)[\mathbf{k}] = \sum_{i=1}^{N_c}|(\mathcal{F}h_i)[\mathbf{k}]|^2 \cdot u[\mathbf{k}]$. □

The quantity $|(\mathcal{F}h_i)[\mathbf{k}]|^2$ is the squared magnitude response of the i-th filter evaluated at the DFT frequency $\mathbf{k}$, and the eigenvalues of $\mathcal{H}^*\mathcal{H}$ are the sum over the N_c channels of these squared magnitude responses. As the DFT consists of samples of the DTFT, by Lemma 1 and (13), the eigenvalues of $\mathcal{H}^*\mathcal{H}$ can be seen to be samples of the trigonometric polynomial $\varphi(\boldsymbol{\omega})$ over the set $\Theta_N = \left\{\left(\frac{2\pi k_1}{N}, \frac{2\pi k_2}{N}\right) : 0 \leq k_1, k_2 < N\right\}$.

Recall that $\mathcal{H}$ implements a frame expansion only if the smallest eigenvalue of $\mathcal{H}^*\mathcal{H}$ is strictly positive [31]. We have the following PR condition for cyclic convolution filter banks:

Corollary 1. *The undecimated cyclic filter bank $\mathcal{H}$ implements a frame expansion for $\mathbb{R}^{N\times N}$ if and only if $\sum_{i=1}^{N_c}|(\mathcal{F}h_i)[\mathbf{k}]|^2 > 0$ for $0 \leq k_1, k_2 < N$. If $\mathcal{H}$ implements a frame expansion, the upper and lower frame bounds are $\min_{\mathbf{k}}\sum_{i=1}^{N_c}|(\mathcal{F}h_i)[\mathbf{k}]|^2$ and $\max_{\mathbf{k}}\sum_{i=1}^{N_c}|(\mathcal{F}h_i)[\mathbf{k}]|^2$.*

Whereas the PR condition for a linear convolution filter bank must hold over the unit circle, the PR condition for cyclic convolution filter bank involves only the N^2 DFT frequencies.

The factorization $\mathcal{H}^*\mathcal{H} = \mathcal{F}^*\mathcal{D}\mathcal{F}$ also provides an easy way to compute the (minimum norm) synthesis filter bank $\mathcal{H}^\dagger$ that satisfies $\mathcal{H}^\dagger\mathcal{H} = \mathcal{I}$. We have $\mathcal{H}^\dagger = (\mathcal{H}^*\mathcal{H})^{-1}\mathcal{H}^*$, and the necessary inverse is given by $(\mathcal{H}^*\mathcal{H})^{-1} = \mathcal{F}^*\mathcal{D}^{-1}\mathcal{F}$.

2) Return to Linear Convolution: We now want to link the PR conditions for cyclic and linear convolution filter banks. Recently, we have shown [47], [48] that the minimum value of a real, multivariate trigonometric polynomial can be lower bounded given sufficiently many uniformly spaced samples of the polynomial, provided that the polynomial does not vary too much over the sampling points.

Theorem 1 (Corollary 3, [48])**.** *Let $\varphi(\boldsymbol{\omega})$ be a real, non-negative, two-dimensional trigonometric polynomial with maximum component order n. Define $\Theta_N = \left\{\left(\frac{2\pi k_1}{N}, \frac{2\pi k_2}{N}\right) : 0 \leq k_1, k_2 < N\right\}$ where $N \geq 2n+1$. If $\kappa_N \triangleq \frac{\max_{\boldsymbol{\omega}\in\Theta_N}\varphi(\boldsymbol{\omega})}{\min_{\boldsymbol{\omega}\in\Theta_N}\varphi(\boldsymbol{\omega})}$ satisfies*

$$\kappa_N \leq \frac{N}{n} - 1, \tag{17}$$

then $\varphi(\boldsymbol{\omega}) > 0$ for all $\omega_1, \omega_2 \in [0, 2\pi)$.

If $\varphi(\boldsymbol{\omega})$ is defined by (13), then κ_N is the frame condition number of a cyclic convolution filter bank operating on $N \times N$ images. Theorem 1 is the link between PR properties of cyclic and linear convolution filter banks we desired, and we have the following PR condition for linear convolution filter banks:

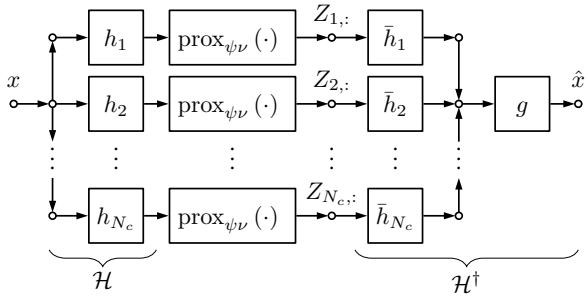


Fig. 3: Analysis-synthesis filter bank generated by sparsifying transform W and separable sparsity penalty ψ. Here, $h_i = \mathcal{R}_1^* w^i$, the impulse response $\bar{h}_i$ is the flipped version of h_i, and g is the impulse response for the filter $(\mathcal{H}^*\mathcal{H})^{-1}$.

Corollary 2. *Let $\mathcal{H}_C$ be an undecimated cyclic convolution filter bank with $K \times K$ filters that operates on $N \times N$ images, with frame condition number κ_N. Let $\mathcal{H}$ be a linear convolution filter bank constructed from the same filters as $\mathcal{H}_C$. Then $\mathcal{H}$ is PR if $\kappa_N \leq \frac{N}{K-1} - 1$.*

Proof. Take $n = K - 1$ in Theorem 1. □

Corollary 2 states that well-conditioned PR cyclic convolution filter banks, with filters that are short relative to image size N, are also PR *linear* convolution filter banks.

The PR conditions of Corollaries 1 and 2 are significantly more general than the patch-based PR conditions. For example, $W \in \mathbb{R}^{N_c \times K^2}$ can be left-invertible only if $N_c \geq K^2$. The PR conditions of Corollaries 1 and 2 have no such requirements; indeed, a single channel "filter bank" can be PR. Our PR conditions are easy to check, requiring only the 2D DFT of N_c small filters.

F. The role of sparsification

We have interpreted the transformed image patches WX as the output of a filter bank. The sparse matrix Z in (1) can be viewed as passing the filter bank output through a nonlinear function implementing $\mathrm{prox}_{\psi\nu}(WX)$. This interpretation is particularly appealing whenever ψ is coordinate-wise separable, meaning $\psi(z) = \sum_i \psi(z_i)$. Then the transform sparse code for the j-th channel depends only on the j-th filtered channel and is given by $\mathrm{prox}_{\psi\nu}(\mathcal{H}_j x)$. The resulting nonlinear analysis-synthesis filter bank is illustrated in Fig. 3. If the input signal x is indeed perfectly sparsifiable by the filter bank (*i.e.*, $\mathcal{H}x = \mathrm{prox}_{\psi\nu}(\mathcal{H}x)$), then the output of the analysis stage is invariant to the application of the nonlinearity and the entire system functions as an identity operator.

We can replace the usual soft or hard thresholding functions by exotic nonlinearities, such as the firm thresholding function [49] or generalized p-shrinkage functions [50]. These nonlinearities have led to marginally improved performance in image denoising [18] and compressed sensing [51]. Alternatively, we can abandon the interpretation of the nonlinearity as a proximal operator and instead learn a custom nonlinearity for each channel, either in a supervised setting [19], [52] or in an unsupervised setting with a Gaussian noise model [53].

Filter bank sparsifying transforms share many similarities with convolutional autoencoder (CAE) [54]. Both consist of a filter bank followed by a channelwise nonlinearity. However, in the case of a CAE, the "decoder" $\mathcal{H}^\dagger$ is typically discarded and the output of the "encoder", $\mathrm{prox}_{\psi\nu}(\mathcal{H}x)$, is passed into additional layers for further processing.

G. Principal Component Filter Banks

The previous sections have shown that transform learning can be viewed as adapting a filter bank to sparsify our data. A similar problem is the design of *principal component filter banks* (PCFB). Let $\mathscr{C}$ denote a set of orthonormal filter banks, such as N_c-channel filter banks with downsampling matrix M, and let x be a given input signal. A filter bank $\mathcal{H}^{\mathscr{P}}$ is said to be a PCFB for the class $\mathscr{C}$ and the input x if, for any $\mathcal{H} \in \mathscr{C}$ and all $m = 1, \ldots, N_c$,

$$\sum_{i=1}^{m} a_i^2 \geq \sum_{i=1}^{m} b_i^2 \tag{18}$$

where a_i and b_i are the ℓ_2 norms of the i-th channel of $\mathcal{H}^{\mathscr{P}}x$ and $\mathcal{H}x$, respectively [55]. A PCFB provides compaction of the output energy into lower-indexed channels, and thus minimal ℓ_2 error when reconstructing a signal from $m < M$ filtered components. The existence and design of PCFBs in 1D is well studied [56]. However, the design of multidimensional FIR PCFBs is again made difficult due to the lack of a multidimensional spectral factorization theorem, although suboptimal algorithms exist [57], [58].

There are superficial similarities between the design of PCFBs and transform learning, especially when W is restricted to be square and orthonormal. The sparsity of the transformed signal implies a form of energy compaction. However, we impose no constraints on location of non-zero coefficients and thus the learned transform is unlikely to satisfy the majorization property (18). Further, an orthogonal W matrix induces an orthogonal filter bank only if non-overlapping patches are used. The PCFB for such a block transformation is known to be the Karhunen-Loeve transformation of the data [59], from which the learned transform can differ substantially [2]. Conversely, the energy majorization property (18) does not imply sparsity of the channel outputs, and a PCFB will not, in general, be a filter bank sparsifying transform.

IV. Learning a Sparsifying Filter Bank

We briefly review methods to incorporate an adaptive sparsity model in the solution of inverse problems. We consider two paradigms: in the "universal" paradigm, our sparsifying transform $\mathcal{H}$ is learned off-line over a set of training data. In the "adaptive" paradigm, the transform is learned during the solution of the inverse problem, typically by alternating between a signal recovery phase and a transform update phase. For synthesis dictionary learning it has been reported that the adaptive method typically works better for lower noise levels while the universal method shines in high noise [3]. In both paradigms, we learn sparsifying transform by minimizing a function that is designed to encourage "good" transforms.

A. Problem Formulation

We now develop a method to learn an undecimated filter bank sparsifying transform that takes advantage of the flexibility granted by the PR conditions of Corollaries 1 and 2. Let x be a training signal, possibly drawn from a set of training signals. We wish to learn a filter bank sparsifying transform that satisfies four properties:

(D1) $\mathcal{H}x$ should be sparse;
(D2) $\mathcal{H}$ should be left invertible and well conditioned;
(D3) $\mathcal{H}$ should contain no duplicated or uniformly zero filters;
(D4) $\mathcal{H}$ should have few degrees of freedom.

Properties (D1) – (D3) ensure our transform is "good" in that it sparsifies the training data, is a well-behaved frame over the space of images, and is not overly redundant. Property (D4) ensures good generalization properties under the universal paradigm and prevents overfitting under the adaptive paradigm.

As with previous transform learning approaches, we satisfy (D1) by minimizing $\frac{1}{2}\|\mathcal{H}x - z\|_2^2 + \nu\psi(z)$ where ψ is a sparsity-promoting functional. The first of term is called the *sparsification error*, as it measures the distance between $\mathcal{H}x$ and its sparsified version, z.

We satisfy (D4) by writing the action of the sparsifying transform on the image as WX, where $W \in \mathbb{R}^{N_c \times K^2}$ and X is formed by extracting and vectorizing $K \times K$ patches with unit stride. This parameterization ensures that we have the desired filter bank structure, and that the learned filters are compactly supported and have only $N_c K^2$ free parameters.

This is a key difference between convolutional analysis-based methods, such as ours, and synthesis-based convolutional dictionary learning; learning a convolutional dictionary requires careful parameterization to get both the desired convolutional structure and filters of compact support.

We emphasize that WX and $\mathcal{H}x$ are equivalent modulo a reshaping operation. Both expressions should be thought of independently of the computational tool used to calculate the results; WX can be implemented using Fourier-based fast convolution algorithms, just as $\mathcal{H}x$ can be implemented by dense matrix-matrix multiplication. We further elaborate on this point in Section IV-C. We choose to write the filter bank application as WX so that we can express the sparsification error directly in terms of W; in particular, we have

$$f(W, Z, x) = \frac{1}{2}\|WX - Z\|_F^2, \tag{19}$$

where the j-th row of $Z \in \mathbb{R}^{N_c \times N^2}$ is the sparse code for the j-th channel output. We can learn a transform over several images by summing the sparsification error for each image.

We promote transforms that satisfy (D2) through the penalty $\frac{1}{2}\sum_{j=1}^{N_c}\|W_{j,:}\|_2^2 - \log\det\mathcal{H}^*\mathcal{H}$. The log determinant term ensures that no eigenvalues of $\mathcal{H}^*\mathcal{H}$ become zero, while the ℓ_2 norm term ensures that the filters do not grow too large. This penalty can be written as $\sum_{j=1}^{N^2}\lambda_i - \log\lambda_i$, where the λ_i are the eigenvalues of $\mathcal{H}^*\mathcal{H}$ as given by Lemma 1. Our proposed penalty serves the role of the final two terms of the patch-based objective (3). The key difference is that the patch-based regularizer acts on the singular values of W, while the proposed regularizer acts on the singular values of $\mathcal{H}$.

To satisfy (D4) we write the eigenvalues λ_i in terms of the matrix W. Let $F \in \mathbb{C}^{N^2 \times N^2}$ denote the matrix that computes the $N \times N$ orthonormal 2D-DFT for a vectorized signal, and let $\bar{F} \in \mathbb{C}^{N^2 \times K^2}$ represent the $N \times N$ 2D-DFT of a zero-padded and vectorized $K \times K$ signal. The i-th column of $\bar{F}W^T$ contains the (vectorized) 2D-DFT of the i-th filter. Then $\lambda_i = \sum_{j=1}^{N_c}\left|\bar{F}W^T\right|_{i,j}^2$, and

$$\log\det\mathcal{H}^*\mathcal{H} = \sum_{i=1}^{N_F^2}\log\left(\sum_{j=1}^{N_c}\left|\bar{F}W^T\right|_{i,j}\right), \tag{20}$$

where the absolute value and squaring operations are taken pointwise. We can reduce the computational and memory burden of the algorithm by using smaller $N_F \times N_F$ DFTs, provided that Corollary 2 implies the corresponding linear convolution filter bank is PR. We take $N_F = 4K$, which is suitable for filter banks with condition number less than 3.

Similar to earlier work on analysis operator learning [13]–[15], we found that our tight frame penalty often resulted in transforms with many uniformly zero filters. We prevent zero-norm filters by adding the log-barrier penalty $\sum_{j=1}^{N_c} -\log\left(\|W_{j,:}\|_2^2\right)$. The combined regularizer is written as

$$J_1(W) = \frac{1}{2}\sum_{i=1}^{N_c}\|W_{i,:}\|_2^2 - \sum_{i=1}^{N_F^2}\log\left(\sum_{j=1}^{N_c}\left|\bar{F}W^T\right|_{i,j}^2\right) - \sum_{i=1}^{N_c}\log\left(\|W_{i,:}\|_2^2\right). \tag{21}$$

The following proposition (proved in Appendix B) indicates that J_1 promotes filter bank transforms that satisfy (D2).

Proposition 2. *Let $W^\sharp$ be a minimizer of J_1, and let $\mathcal{H}$ be the undecimated cyclic convolution filter bank generated by the rows of $W^\sharp$. Then $\mathcal{H}$ implements a uniformly normalized tight frame expansion over the space of images, with filter squared norms equal to $2(1 + N_F^2/N_c)$ and frame constant $2(1 + N_c/N_F^2)$.*

Finally, we would like to discourage learning copies of the same filter. To that end, we define the coherence between rows i and j of W as

$$\Gamma_{i,j}(W) \triangleq \frac{\langle W_{i,:}, W_{j,:}\rangle}{\|W_{i,:}\|_2\|W_{j,:}\|_2}. \tag{22}$$

One option is to apply a log barrier to the squared coherence between each pair of filters [16]:

$$J_2(W) = \sum_{1 \le i < j \le N_c} -\log\left(1 - \left(\Gamma_{i,j}(W)\right)^2\right). \tag{23}$$

This penalty works well whenever the filters have small support ($K \le 8$). For larger filters, we observed the algorithm often learned filters with disjoint support that are shifted versions of one another. These filters do not cause a large value in (23), yet provide no advantage over a single filter. We modify our coherence penalty to discourage filters that differ by only a linear phase term by applying (23) to the squared magnitude responses of our filters. This coherence

penalty naturally promotes zero-mean filters, as the coherence between two non-zero-mean filters can be reduced simply by removing their mean.

Our learning problem is written as

$$\min_{W,Z} f(W,Z,x) + \mu J_1(W) + \lambda J_2(W) + \nu\psi(Z). \quad (24)$$

The scalar $\mu > 0$ controls the strength of the UNTF penalty and should be large enough that the learned filter bank is well conditioned, so that approximating the eigenvalues using $N_F \times N_F$ DFTs remains valid. The non-negative scalar parameters λ and ν control the emphasis given to the coherence and sparsity penalties, respectively.

B. Optimization Algorithm

We use an alternating minimization algorithm to solve (24). In the *sparse coding step*, we fix W and solve (24) for Z. In the second stage, called the *transform update step*, we update our transform W by minimizing (24) with fixed Z. We use superscripts to indicate iteration number, and we take $\mathcal{H}^{(k)}$ to mean the filter bank generated using filters contained in the rows of $W^{(k)}$.

The sparse coding step reduces to

$$Z^{(k+1)} = \arg\min_Z \frac{1}{2}\|W^{(k)}X - Z\|_F^2 + \nu\psi(Z) \quad (25)$$

with solution $Z^{(k+1)} = \text{prox}_{\psi\nu}\left(\mathcal{H}^{(k)}x\right)$. Next, with $Z^{(k+1)}$ fixed, we update W by solving

$$W^{(k+1)} = \arg\min_W f(W, Z^{(k+1)}, x) + \mu J_1(W) + \lambda J_2(W). \quad (26)$$

Unlike the square, patch-based case, we do not have a closed-form solution for (26) and we must resort to an iterative method. The limited-memory BFGS (L-BFGS) algorithm works well in practice. The necessary gradients are

$$\nabla_W f(W,Z,x) = 2WXX^T - 2XZ^T, \quad (27)$$

$$\nabla_W \log\det \mathcal{H}^*\mathcal{H} = 2W\bar{F}^* \,\text{ddiag}\left(\left|\bar{F}W^T\right|^2 \mathbf{1}_{N_c}\right)^{-1}\bar{F}, \quad (28)$$

$$\frac{\partial}{W_{r,s}} \sum_{i=1}^{N_c} \log\left(\|W_{i,:}\|_2^2\right) = \frac{2W_{r,s}}{\|W_{r,:}\|_2^2}, \quad (29)$$

$$\frac{\partial J_2(W)}{\partial W_{r,s}} = \sum_{i=1, i\neq r}^{N_c} \frac{W_{i,s}[WW^T]_{i,r} - W_{r,s}[WW^T]_{i,r}^2\|W_{r,:}\|_2^{-2}}{\|W_{i,:}\|_2^2 \cdot \|W_{r,:}\|_2^2 - [WW^T]_{i,r}^2}. \quad (30)$$

C. Computational Considerations

The primary bottleneck in using L-BFGS to solve (26) is the line search step, which requires multiple evaluations of the objective function (24) with fixed Z. The cost of this computation is dominated by evaluation of (19). With X and Z fixed, we precompute and store the small matrices $G = XX^T$ and $Y = XZ^T$. The sparsification residual is evaluated as

$$\text{trace}\left(W^TWG\right) - 2\cdot\text{trace}\left(WY\right) + \|Z\|_F^2 \quad (31)$$

and requires only small matrix-matrix products.

In the patch-based case, evaluating WX using dense matrix multiplication requires $\mathcal{O}(N_cK^2N^2)$ floating point operations (FLOPS). The filter bank structure of $\mathcal{H}$ naturally leads to efficient calculation of $\mathcal{H}x$ through the use of Fourier-based convolution methods. For simplicity, we will restrict our attention to radix-2 FFT algorithms and assume that both K and N are powers of 2.

The usual Fourier-based circular convolution methods require adding zeros until both signals are of the same size. In our case, we must extend each row of W to be of length N. Passing the input signal through a single filter will require $\mathcal{O}(N_cN^2\log N)$ FLOPS, representing a $K^2/\log(N)$ reduction over the patch-based case. For the typical sizes of $N = 512$ and $K = 8$, this is roughly a factor of 7. Importantly, using convolution to evaluate $\mathcal{H}x$ does not require explicitly forming or storing the matrix X. The number of multiplications needed to apply the analysis filter bank using patch-based and Fourier approaches is plotted in Fig. 4

We also use convolution to accelerate calculation of (27). The first term, XX^T, is just the circular correlation of x evaluated at the first K shifts in each direction and can be calculated using FFTs at a cost of $\mathcal{O}(N^2\log(N))$. In contrast, evaluating this gradient using a dense matrix multiply involving the image patch matrix scales as $\mathcal{O}(K^4N^2)$. Thus the filter bank interpretation yields a savings of $K^4/\log(N)$. For typical sizes of $N = 512$ and $K = 8$, this is a $450\times$ reduction in order. However, this term remains constant throughout the iterations and must be computed only once.

Computing the product XZ^T is more complicated. The matrix Z, while sparse, has no fixed sparsity pattern or common structure. We must compute the product with each row of Z independently using convolution. This requires a forward FFT of length N for x and for each of the N_c rows of Z, the necessary elementwise products, and finally the inverse FFT of these products. All told, this operation will scale like $\mathcal{O}(N_cN^2\log(N))$. In contrast, directly using dense matrix multiplication scales as $\mathcal{O}(N_cK^2N^2)$. Unlike XX^T, this term must be calculated each time z is updated.

The dominant computation in evaluating $J_1(W)$ is that of $\bar{F}W^T$. This requires N_c separate 2D-DFTs of size $N_F \times N_F$, at a cost of $\mathcal{O}(N_cN_F^2\log N_F)$.

Similarly, calculating the gradient of $J_1(W)$ is dominated by the cost of (28). We first compute $\bar{F}W^T$. Then, we require the multiplication of an $N_F^2 \times N_F^2$ diagonal matrix by the $N_F^2 \times K^2$ matrix $\bar{F}$ at a cost of $\mathcal{O}(N_F^2K^2)$. Next, we take K^2 separate $N_F \times N_F$ 2D inverse FFTs, followed by the product of $N_c \times K^2$ and $K^2 \times K^2$ matrices. Together, $\nabla_W J_1(W)$ scales as $\mathcal{O}(K^4N_c + (N_c + K^2)N_F^2\log(N_F))$. The cost of evaluating $J_2(W)$ and $\nabla J_2(W)$ is $\mathcal{O}(N_c^2K^2)$ and $\mathcal{O}(N_cK^4)$, respectively. These costs are summarized in Table I.

For many choices of ψ, the sparse coding step is cheap. For instance, when ψ is the ℓ_0 norm, we need only to pass over each element of $\mathcal{H}x$ and set to zero all entries that are less than the given threshold. This operation will cost $\mathcal{O}(N_cN^2)$.

The necessary function and gradient evaluations consist of basic linear algebra operations, such as matrix-matrix products, and elementwise function evaluations, such as $\log(\cdot)$ or $|\cdot|^2$. As such, our algorithm is easy to implement on a graphics processing unit (GPU).

As noted, we can implement the action of the filter bank, $\mathcal{H}x$, using Fourier convolution methods, direct convolution methods, or using the patch-based multiplication WX. The best choice depends on computational platform (CPU vs GPU), filter size, and dimensionality of training data. While Fourier methods likely win on a CPU, patch-based multiplication is well suited for GPU-based implementations. Finally, note that we can limit the amount of memory consumed by the algorithm by applying the filter bank in a channel-by-channel (or row-by-row) fashion. This is useful when learning a transform for higher dimensional data, as the matrix WX may not fit in memory- for d-dimensional data the matrix WX is of size $N_c N^d$, while a single row of WX is of size N^d.

D. Comparison with NSOLT

The closest analogue to our proposed filter bank design algorithm is the Non-Separable Oversampled Lapped Transform (NSOLT), as described in Section III-D. We briefly draw distinctions between our proposed filter bank learning algorithm and the design of multi-dimensional NSOLTs.

An immediate difference is that the NSOLT structure is proposed for use as a synthesis filter bank. This is not a meaningful distinction, though, as the designed NSOLT implements a tight frame expansion, and thus the adjoint of the NSOLT is itself a paraunitary analysis filter bank with compactly supported FIR filters [34].

The true differences between our approach and the design of NSOLTs lie in the structure of the filter bank. First, our algorithm is only applicable to undecimated filter banks, while NSOLTs can incorporate downsampling. Second, our approach can learn any undecimated PR filter bank with compactly supported FIR filters, whereas the NSOLT framework can learn only a subset of paraunitary filter banks. This difference manifests itself in both the structure and optimization of the filter banks. Our filter banks are unstructured, and use special regularizers to ensure the PR property holds. In our approach, the frame bounds are indirectly controlled through the penalty parameter μ. In contrast, the NSOLT uses a particular lattice form that implicitly guarantees the learned filter bank is paraunitary.

Further, NSOLTs are designed using a combination of symmetric and anti-symmetric impulse responses to ensure the filter bank is linear phase [26]–[28]. This constraint limits the ability of individual NSOLT channels to capture structures which are not strictly symmetric or anti-symmetric, such as edges that are not strictly horizontal, vertical, or at an angle of 45 degrees. Our undecimated filter banks do not have this restriction.

V. Application to Image Denoising

A preliminary version of our filter bank transform learning framework has been applied in an "adaptive" manner for magnetic resonance imaging [60]. Here, we restrict our attention to image denoising in the "universal" paradigm: we use a pre-trained sparsifying filter bank, $\mathcal{H}$ to recover a clean image x^* from a noisy copy, $y = x^* + e$, where $e \sim \mathcal{N}(0, \sigma^2 I_N)$. We consider two algorithms for image denoising.

Algorithm 1 Filter bank sparsifying transform learning

INPUT: Image x, Initial transform $W^{(0)}$

$Z^{(0)} \leftarrow \text{prox}_{\psi\nu}\left(\mathcal{H}^{(0)}x\right)$
$k \leftarrow 0$
repeat
  $W^{(k+1)} \leftarrow \arg\min_W f(W, x, Z^{(k)}) + \mu J_1(W) + \lambda J_2(W)$
  $Z^{(k+1)} \leftarrow \text{prox}_{\psi\nu}\left(\mathcal{H}^{(k+1)}x\right)$
  $k \leftarrow k + 1$
until Halting Condition

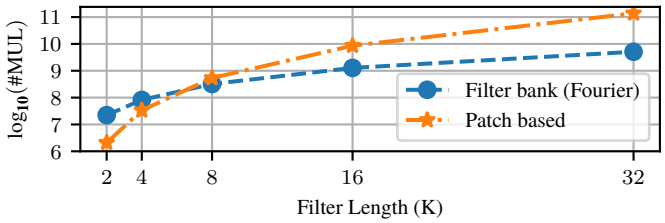


Fig. 4: Number of multiplications needed to apply a square filter bank ($N_c = K^2$) to a 512×512 image using Fourier and patch-based methods.

A. Iterative denoising

Our first denoising method is to solve a regularized inverse problem using a transform sparsity penalty, written as

$$\min_{x,z} \frac{\lambda_r}{2}\|y - x\|_2^2 + \frac{1}{2}\|\mathcal{H}x - z\|_2^2 + \nu\psi(z), \qquad (32)$$

where $\lambda_r > 0$ controls the regularization strength. We solve this problem by alternating minimization: we update z for fixed x, and then update x with z fixed. This procedure is summarized as Algorithm 2. The eigenvalue decomposition of Lemma 1 provides an easy way to compute the necessary matrix inverse for cyclic convolution filter banks. For linear convolution filter banks, we use Lemma 1 to implement a circulant preconditioner [61].

Algorithm 2 has three key parameters. The regularization parameter λ_r reflects the degree of confidence in the noisy observations y and should be chosen inversely proportional to the noise variance. The sparsity of the transform sparse code is controlled by ν. The value of ν when denoising an image need not be the same as ν during the learning procedure and should be proportional to σ. The choice of both ν and λ_r depends on the final parameter: the number of iterations used during denoising. Empirically, we've found that using $\text{ceil}\{\sigma \cdot 255/10\}$ iterations works well.

B. Denoising by Transform-domain Thresholding

We also consider a simpler algorithm, inspired by the transform domain denoising techniques of old. We can form a denoised estimate by passing y through the system in Fig. 3; that is, computing

$$\mathcal{H}^\dagger \text{prox}_{\psi\nu}\left(\mathcal{H}y\right). \qquad (33)$$

Penalty	Evaluation	Gradient
$f(W, Z, x)$	$\mathcal{O}(N_c N^2 \log(N))$	$\mathcal{O}(N_c K^4 + N_c N^2 \log(N))$
$J_1(W)$	$\mathcal{O}\left(N_c N_F^2 \log(N_F) + N_F^2 K^2 + N_c K^2\right)$	$\mathcal{O}(K^4 N_c + (N_c + K^2) N_F^2 \log(N_F)$
$J_2(W)$	$\mathcal{O}\left(N_c^2 K^2\right)$	$\mathcal{O}(N_c K^4)$

TABLE I: Computational cost for function and gradient computation.

Algorithm 2 Iterative denoising with filter bank transform

INPUT: Noisy signal y, Learned filter bank transform $\mathcal{H}$

1: $k \leftarrow 0$
2: **repeat**
3: $\quad z^{(k+1)} \leftarrow \text{prox}_{\psi\nu}\left(\mathcal{H}x^{(k)}\right)$
4: $\quad x^{(k+1)} \leftarrow (\mathcal{H}^*\mathcal{H} + \lambda_r I)^{-1}(\mathcal{H}^* z^{(k)} + \lambda_r y)$
5: $\quad k \leftarrow k + 1$
6: **until** Halting Condition

Fig. 5: Training images.

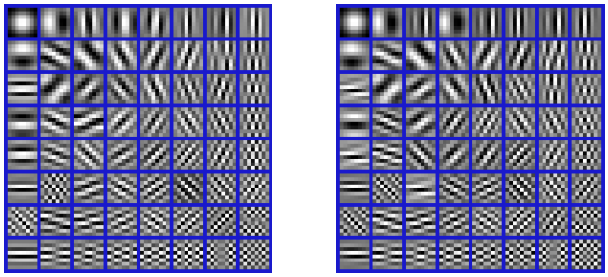


Fig. 6: Comparing initializations for 64 channel filter bank with 8×8 filters. (a) DCT Initialization; maximum coherence: 0.9 (b) Random initialization; maximum coherence: 0.85. The filters in (b) have been ordered to match those in (a).

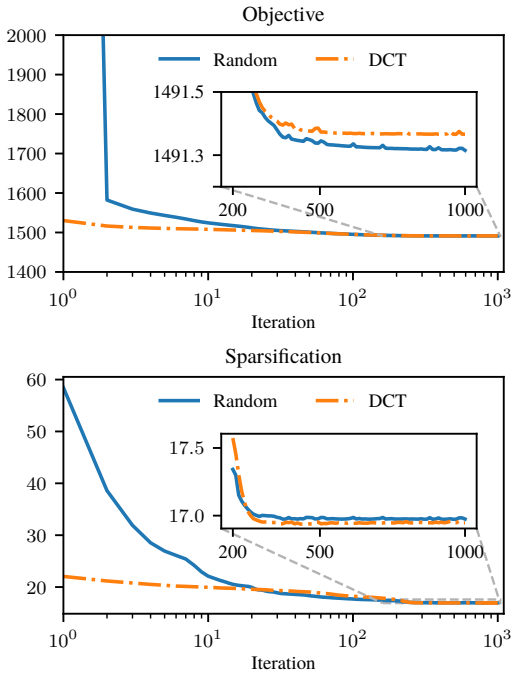


Fig. 7: Plots of objective function (24) and sparsification error $\frac{1}{2}\|\mathcal{H}x - z\|_2^2$ while training the filter banks shown in Fig. 6.

This approach simplifies denoising by eliminating two parameters from Algorithm 2: the number of iterations and λ_r.

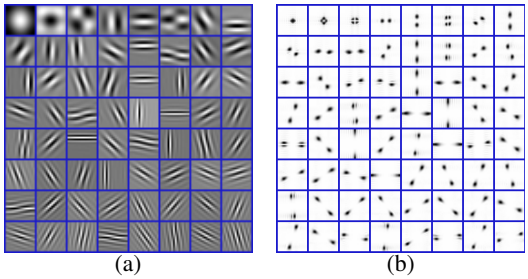
Denoising in this manner is sensible because of the properties we have imposed on $\mathcal{H}$. Noise in the signal will not be sparse in the transform domain and thus will be reduced by the nonlinearity. In contrast, the image will be sparse in the transform domain, and significant components will pass through the nonlinearity with little change. The left-inverse is guaranteed to exist, and as $\mathcal{H}$ must be well-conditioned, any noise remaining after the nonlinearity will not be strongly amplified by $\mathcal{H}^\dagger$. Finally, if $\mathcal{H}$ has low coherence, the transformed noise $\mathcal{H}e$ will not be correlated across channels, suggesting that a channelwise nonlinearity is sufficient. A multi-channel nonlinearity may be beneficial if the transform is coherent.

Unless $\mathcal{H}$ implements a tight frame expansion, the minimum norm synthesis filters comprising $\mathcal{H}^\dagger$ will not be compactly supported; indeed, if $\mathcal{H}$ is a linear convolution filter bank, then the minimum-norm synthesis filter bank will have infinite duration impulse response filters [34]. Fortunately, if $\mathcal{H}$ is well-conditioned, then the minimum-norm synthesis filters will have an exponentially decaying impulse response and can thus be well approximated by FIR filters [62]. Alternatively, one can search for a (non-minimum-norm) left inverse of $\mathcal{H}$ that consists of FIR filters [63].

VI. Experiments

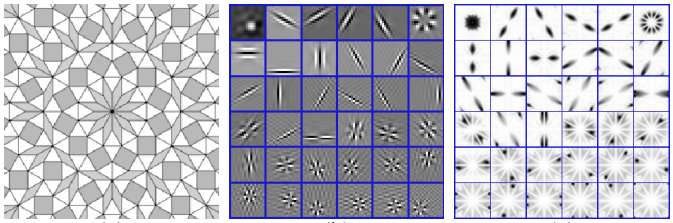
We implemented GPU versions of our algorithms using NumPy 1.11.3 and SciPy 0.18.1. Our code interfaces with Python through PyCUDA and scikits.cuda, and we conducted experiments on an NVidia Maxwell Titan X GPU.

We conducted training experiments using the five training images in Fig. 5. Each image, in testing and training, was normalized to have unit ℓ_2 norm. Unlike many patch-based methods, we do not subtract the DC (mean) value of the image prior to training. Unless otherwise specified, our transforms were learned using 1000 iterations of Algorithm 1 with parameters $\mu = 3.0$, $\lambda = 7 \times 10^{-4}$, and $\nu = 5.5 \times 10^{-3}$. Sparsity was promoted using an ℓ_0 penalty, for which the prox operator corresponds to hard thresholding. For each filter bank, we compute the coherence between each pair of squared magnitude filter responses and report the largest value; that is,



(a) (b)

Fig. 8: Examples of learned 16×16 filters. (a) Filter impulse responses; (b) Magnitude frequency responses. The zero frequency (DC) is located at the center of each small box. Maximum coherence of learned filter bank: 0.88.



(a) (b) (c)

Fig. 9: Adaptivity of filters. (a) Training image; (b) Learned filter impulse responses. (c) Magnitude frequency responses. Zero frequency (DC) is located at the center of each small box. Maximum coherence of learned filter bank: 0.75.

$\max_{1\le i<j<N_c} \Gamma_{i,j}(\left|\bar{F}W^T\right|^2)$.

The initial transform $\mathcal{H}^{(0)}$ must be feasible, *i.e.* left-invertible. Random Gaussian and DCT initializations work well in practice. We learned a 64 channel filter bank with 8×8 filters using these initializations. The learned filters are shown in Fig. 6. The evolution of the objective function and sparsification error are shown in Fig. 7. The learned filters appear similar, reach nearly the same objective value, and perform equally well in sparsifying the data set.

More examples of learned filters and their magnitude frequency responses are shown in Fig. 8. We show a subset of channels from a filter bank consisting of 16×16 filters and 128 channels. This transform is $2\times$ *under*-complete if viewed as a patch-based transform. The ability to choose longer filters without increasing the number of channels is a key advantage of our framework over patch-based transform learning.

A. Image denoising

We investigate the denoising performance of the filter bank sparsifying transforms as a function of number of channels, N_c, and filter size, K, using our two algorithms. We refer to Filter Bank Sparsifying Transform with N_c channels and $K \times K$ filters as FBST-N_c-K.

We evaluate our filter bank learning formulation using 64, 128, and 256 channels with 8×8 and 16×16 filters. During the denoising stage, we set $\nu = 10^{-4} \times 0.1\sigma$ and λ_r was adjusted for the particular noise level.

We also evaluate image denoising using filters learned with the square, patch-based transform learning algorithm [2]. We used 8×8 and 16×16 image patches to learn a patch-based transform W. We used the rows of W to generate an undecimated filter bank and used this filter bank to denoise using Algorithm 2 and (33). The filter bank implements cyclic convolution and image patches are extracted using periodic boundary conditions. Following the convention used to denote filter bank sparsifying transforms, we refer to a Patch-Based Sparsifying Transform with $K \times K$ patches as PBST-K^2-K. We keep the number of channels $N_c = K^2$ explicit to facilitate comparison with filter bank sparsifying transforms. Observe that for a given K, FBST-K^2-K and PBST-K^2-K differ only in the regularizer and learning algorithm used; in particular, both transforms have the same number of design parameters. All sparsifying transforms were trained under the "universal" paradigm using all possible (maximally overlapping) patches extracted from the set of images shown in Fig. 5.

We also include comparisons with 2D NSOLTs trained using the `SaivDr`[2] MATLAB package. The NSOLTs were trained using the images Fig. 5. We investigate denoising performance as a function of polyphase order, downsampling ratio, and number of channels. We refer to an NSOLT with downsampling matrix $2I$, 48 channels, and polyphase order 4 as NSOLT-D2-C48-O4. We consider only NSOLTs with an identical number of symmetric and antisymmetric channels, thus NSOLT-D2-C48-O4 has 24 symmetric and 24 antisymmetric channels.

Image denoising using NSOLTs is accomplished by solving $\arg\min_x \|y-x\|^2/2 + \lambda\|\mathcal{H}x\|_1$, where $\mathcal{H}$ denotes the NSOLT operator and λ is tuned for best denoising performance on an image-by-image basis. The optimization problem itself was solved using FISTA [64]. We also attempted denoising by using Algorithm 2 with an ℓ_0 penalty, but found that ℓ_1 + FISTA gave the best results. We used two tree levels for each NSOLT.

While our main interest is comparing the denoising performance of FBST versus PBST, we also compare against several competing image denoising methods. These are divided into two camps. The first group includes two methods based on non-local self-similarity: BM3D [65] and WNNM [66].

The second group includes a handful of MRF learning-based methods. These can be interpereted as either patch-based analysis or convolutional analysis models and are thus local in nature. We include EPLL-GMM [67], the Field of Experts (FoE) [17], [68], and the Cascade of Shrinkage Fields (CSF) [52]. While these methods have a convolutional structure– indeed, FoE and CSF are closely related to our nonlinear analysis-synthesis filter bank– they must be trained in a supervised in nature and do not impose any perfect reconstruction property on the resulting filters. We use the default parameters in the FoE and CSF packages. For FoE, we use 3×3 filters. For CSF we use 5×5 and 7×7 filters;

[2]Available: https://github.com/msiplab/SaivDr

in both cases, we use 5 stages with 25 channels. We trained the CSF to operate at our noise levels.

Finally, we include the recent STROLLR denoising algorithm, which uses both square patch-based transform learning and non-local self-similarity [69], [70]. STROLLR is an unsupervised method and the transform learning step uses the "adaptive" paradigm. We used 8×8 patches, resulting in a 64×64 sparsifying transform.

Our metric of interest is the peak signal-to-noise ratio (PSNR) between the reconstructed image x and the ground truth image x^*, defined in decibels as PSNR $= 20 \log_{10}(N^2/\|x - x^*\|_2)$. In the Supplemental Material, we also report the (mean) Structural Similarity (SSIM) index, a perceptual image metric that has been shown to be consistent with qualitative visual appearance [71]. The SSIM takes values between 0 and 1, with higher values indicating better quality. We evaluate the denoising performance of our algorithm on the grayscale `barbara`, `man`, `peppers`, `baboon` and `boat` images.

Table II collects the reconstruction PSNR for each test image, in addition to the mean PSNR for the entire test set. The SSIM is collected in the Supplemental Material. In both tables, the best value is written in bold with gray shading; the second best value is shaded gray with no bold. Here, FBST-64-8 indicates a 64 channel filter bank with 8×8 filters where we denoise using transform domain thresholding (33), while FBST-128-16-I indicates the use of a 128 channel filter bank with 16×16 filters and denoising using the iterative Algorithm 2.

When averaged over the entire test set, FBST-64-8 outperforms PBST-64-8 by between $0.2 - 0.3$ dB. As the only difference in these two transforms is the regularizer used during learning, we attribute this improvement to the change from a patch-based to an image-based point of view.

Using 1000 iterations to learn a 196 channel filter bank with 16×16 filters with Algorithm 1 took roughly 5 minutes on our GPU. In contrast, using the same GPU to learn a square 256×256 patch-based transform over the same data took less than one minute. This illustrates the efficiency of the closed-form transform update step in the patch-based case [4]. Our slower learning algorithm is offset by the ability to choose $N_c < K^2$, and this leads to faster application of the learned transform. Comparing FBST-128-16-I and PBST-256-16-I, the image-based transform outperforms the patch-based transform by up to 0.3 dB despite containing half as many channels.

Table II shows that, on average, FBST performs slightly better than the MRF-based methods (EPLL/CSF/FoE), but worse than non-local methods, especially for $\sigma = 30$. Note that STROLLR is competitive with the other non-local methods. Combining the flexibility of our proposed filter bank sparsifying transforms with STROLLR is left for future work. NSOLT gives the lowest denoising performance. We conjecture that the linear phase constraints severely limit the representation power of the NSOLT. Further, it is difficult to train a large NSOLT: training NSOLT-D2-C24-O2 required several days on our workstation.

The performance of shorter or longer filters is dependent on the image. For most images, the 8×8 filters performed as well or better than the longer filters, but we see significant improvement when using long filters on `barbara`. The MRF-based methods perform poorly on this image, with the FBST-I methods gaining well over a full dB of PSNR improvement. In contrast, the MRF methods outperform FBST on `man`.

Increasing the number of channels beyond the filter size K^2 provides marginal improvement. For low noise, denoising by transform-domain thresholding and iterative denoising using Algorithm 2 perform equally well. As the noise level increases, the iterative denoising algorithm outperforms the simpler thresholding scheme.

B. Learning on subset of patches

One advantage of patch-based formulation is that the model can be trained using a large set images by randomly selecting a few patches from each image. We can use the same approach when learning a sparsifying filter bank: the data matrix X in (19) is formed by extracting and vectorizing patches from many images. We can no longer view WX as a convolution.

We learned a transform using $200,000$ randomly extracted patches from the training images in Fig. 5. The learned transform performed nearly identically to a transform learned using all patches from the training images.

C. Image Adaptivity

To test the influence of the training set, we learned a filter bank using 256^2 patches of size 8×8 chosen at random from the 200 training images in the BSDS300 training set [72]. The learned filter bank consists of Gabor-like filters, much like filter banks learned from the images in Fig. 5. Gabor-like filters are naturally promoted by the regularizers J_1 and J_2: their narrow support in the frequency domain leads to low coherence, and their magnitude responses can tile frequency space leading to a well-conditioned transform. As expected, all but one filter has zero-mean.

We wondered if we have regularized our learning problem so strongly that the data no longer plays a role. Fortunately, this is not so: Fig. 9 illustrates a 64 channel filter bank of 16×16 filters learned from a highly symmetric and geometric image. The learned filters include oriented edge detectors, as in the natural image case, but also filters with a unique structure that sparsify the central region of the image. Note that most of our learned filters in Figs. 8 and 9 are not strictly symmetric or antisymmetric, and thus cannot be captured by individual NSOLT channels.

VII. Remarks

Adaptive analysis/transform sparsity based image denoising algorithms can be coarsely divided into two camps: supervised and unsupervised. In both cases, one learns a signal model by minimizing an objective function.

In the supervised case, this minimization occurs over a set of training data. In a denoising application one typically corrupts a clean input with noise, passes it through the denoiser, and uses the difference between the clean and denoised signal to adapt various components of the denoising algorithm: the

analysis operator, thresholding parameters, mixture weights, the number of iterations, and so on. It is not necessary to regularize the learning procedure to preclude degenerate solutions, such as a transform of all zeros; such a transform would not perform well at the denoising task, and thus would not be learned by the algorithm [17], [18], [29].

In the unsupervised case, the objective function has two components. The first is a surrogate for the true, but unavailable, metric of interest. In this paper, we use the combination of sparsity and sparsification error to act as a surrogate for reconstruction PSNR. The second part of the objective is a regularizer that prevents degenerate solutions, as discussed in Sections II-B and II-C. Even in the "universal" case, our learning is essentially unsupervised, as the learning process is not guided by the denoising PSNR.

The TNRD algorithm [19] is a supervised approach that resembles iterative denoising using Algorithm 2, but where the filter coefficients, nonlinearities, and regularization parameter are allowed to vary as a function of iteration. However, the TNRD approach has no requirements that the filters form a well-conditioned frame or have low coherence; "poor" filters are simply discouraged by the supervised learning process. Denoising with the TNRD algorithm outperforms the learned filter bank methods presented here.

One may ask if it is necessary that the learned transform be a frame. Indeed, the matrix to be inverted when denoising using Algorithm 2 is full-rank even if the filter bank itself is not perfect reconstruction. The proposed regularizer, while less restrictive than previous transform learning regularizers, may still overly constrain the set of learnable sparsifying transforms. However, our highly regularized approach has a benefit of its own. Whereas the TNRD algorithm is trained on hundreds of images, and can take upwards of days to train, our algorithm can be applied to a single image and requires only a few minutes. The tradeoff offered by the TNRD algorithm is acceptable for image denoising tasks, as massive sets of natural images are publicly available for use in machine learning applications. However, such data sets may not be available for new imaging modalities, in which case a tradeoff closer to that offered by our filter bank learning algorithm may be preferred. Finding a balance between our highly regularized and unsupervised approach and competing supervised learning methods is the subject of ongoing work.

VIII. Conclusions

We have developed an efficient method to learn sparsifying transforms that are structured as undecimated, multi-dimensional perfect reconstruction filter banks. Unlike previous transform learning algorithms, our approach can learn a transform with fewer rows than columns. We anticipate this flexibility will be important when learning a transform for high dimensional data. Numerical results show our filter bank sparsifying transforms outperform existing patch-based methods in image denoising. Future work might fully embrace the filter bank perspective and learn filter bank transforms with various length filters and/or non-square impulse responses.

Appendix A

We explicitly show the link between filter banks and applying a sparsifying transform to a patch matrix. We assume a 1D signal $x \in \mathbb{R}^N$ to simplify notation. The extension to multiple dimensions is tedious, but straightforward.

Let $W \in \mathbb{R}^{N_c \times K}$ be a given transform, and let w^i indicate the i-th row of this matrix. Suppose we extract patches with a patch stride of s and we assume s evenly divides N. The j-th column of the patch matrix $X \in \mathbb{R}^{K \times M}$ is the vector $[x_{sj+K-1}, x_{sj+K-2}, \dots, x_{sj}]^T$. The number of columns, M, depends on the boundary conditions used. Linear and circular convolution are obtained by setting $x_i = 0$ or $x_i = x_{N-i-1}$, respectively, when $i < 0$. For cyclic convolution, we have $M = N/s$. The i,j-th element of the sparsified signal WX is

$$[WX]_{i,j} = \sum_{k=1}^{K} W_{i,k} X_{k,j} = \sum_{k=1}^{K} W_{i,k} x_{sj+K-1-i} \tag{34}$$

$$= (w^i * x)[sj + K - 1]. \tag{35}$$

Thus the i-th row of WX is the convolution between the filter with impulse response w^i and signal x, followed by downsampling by a factor of s, and shifted by $K-1$. The filter bank has N_c channels with impulse responses given by the rows of W. The shift of $K-1$ can be incorporated into the definition of the patch extraction procedure. For 1D signals, the "first" patch should be $[x_{K-1}, \dots x_0]^T$, while for 2D signals, the lower-right pixel of the "first" patch is $x[0,0]$.

Appendix B
Proof of Proposition 2

The function $J_1(W)$ in (21) acts only on the magnitude responses of the filters in $\mathcal{H}$. Let $V \triangleq \left|\bar{F} W^T\right|^2 \in \mathbb{R}^{N_F^2 \times N_c}$. The sum of the i-th column of V is equal to the norm of the i-th filter and, by Lemma 1, the eigenvalues of $\mathcal{H}^*\mathcal{H}$ are equal to the row sums of V. Thus, V is generated by a UNTF if and only if the row sums and column sums are constant.

Let $V^\sharp$ be a stationary point of J_1. For each $1 \le r \le N_F^2$ and $1 \le s \le N_c$, we have

$$\frac{\partial}{\partial V_{r,s}} J_1(V^\sharp) = \frac{1}{2} - \frac{1}{\sum_{j=1}^{N_c} V^\sharp_{r,j}} - \frac{1}{\sum_{i=1}^{N_F^2} V^\sharp_{i,s}} = 0. \tag{36}$$

Note that $J_1(V) = +\infty$ if either a row or column of V is identically zero, so $V^\sharp$ is a minimizer only if there is at least one non-zero in each row and column of $V^\sharp$. Subtracting $\frac{\partial}{\partial V_{r',s}} J_1(V^\sharp)$ from $\frac{\partial}{\partial V_{r,s}} J_1(V^\sharp)$ yields $\sum_{j=1}^{N_c} V^\sharp_{r,j} = \sum_{j=1}^{N_c} V^\sharp_{r',j} \triangleq a$. Similarly, subtracting $\frac{\partial}{\partial V_{r,s}} J_1(V^\sharp)$ from $\frac{\partial}{\partial V_{r,s'}} J_1(V^\sharp)$ yields $\sum_{i=1}^{N_F^2} V^\sharp_{i,s} = \sum_{i=1}^{N_F^2} V^\sharp_{i,s'} \triangleq b$. As the row and column sums are uniform for each r and s, we conclude $V^\sharp$ is a UNTF. Next, we have

$$\sum_{i=1}^{N_F^2} \sum_{j=1}^{N_c} V^\sharp_{i,j} = \sum_{i=1}^{N_F^2} \left(\sum_{j=1}^{N_c} V^\sharp_{i,j} \right) = N_F^2 a \tag{37}$$

$$= \sum_{j=1}^{N_C} \left(\sum_{i=1}^{N_F^2} V^\sharp_{i,j} \right) = N_c b, \tag{38}$$

from which we conclude $b = \frac{N_F^2}{N_c} a$. Substituting into (36), we find

$$a = 2\left(1 + \frac{N_c}{N_F^2}\right), \qquad b = 2\left(\frac{N_F^2}{N_c} + 1\right), \tag{39}$$

and this completes the proof.

REFERENCES

[1] E. J. Candes and D. L. Donoho, "New tight frames of curvelets and optimal representations of objects with piecewise C^2 singularities," *Commun. Pure Appl. Math.*, vol. 57, pp. 219–266, 2004.

[2] S. Ravishankar and Y. Bresler, "Learning sparsifying transforms," *IEEE Trans. Signal Process.*, vol. 61, pp. 1072–1086, 2013.

[3] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. Image Process.*, vol. 15, pp. 3736–45, Dec. 2006.

[4] S. Ravishankar and Y. Bresler, "Closed-form solutions within sparsifying transform learning," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[5] ——, "Learning doubly sparse transforms for images," *IEEE Trans. Image Process.*, vol. 22, pp. 4598–4612, 2013.

[6] ——, "Learning overcomplete sparsifying transforms for signal processing," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 3088–3092.

[7] B. Wen, S. Ravishankar, and Y. Bresler, "Structured overcomplete sparsifying transform learning with convergence guarantees and applications," *International Journal of Computer Vision*, Oct. 2014.

[8] 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.

[9] S. Ravishankar and Y. Bresler, "Sparsifying transform learning for compressed sensing MRI," in *International Symposium on Biomedical Imaging*, 2013.

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

[11] ——, "Adaptive sparsifying transforms for iterative tomographic reconstruction," in *International Conference on Image Formation in X-Ray Computed Tomography*, 2014.

[12] R. Rubinstein, T. Peleg, and M. Elad, "Analysis K-SVD: A dictionary-learning algorithm for the analysis sparse model," *IEEE Trans. Signal Process.*, vol. 61, pp. 661–677, Feb. 2013.

[13] M. Yaghoobi, S. Nam, R. Gribonval, M. E. Davies *et al.*, "Analysis operator learning for overcomplete cosparse representations," in *European Signal Processing Conference (EUSIPCO'11)*, 2011.

[14] M. Yaghoobi, S. Nam, R. Gribonval, and M. E. Davies, "Noise aware analysis operator learning for approximately cosparse signals," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 5409–5412.

[15] ——, "Constrained overcomplete analysis operator learning for cosparse signal modelling," *IEEE Trans. Signal Process.*, vol. 61, pp. 2341–2355, May 2013.

[16] S. Hawe, M. Kleinsteuber, and K. Diepold, "Analysis operator learning and its application to image reconstruction," *IEEE Trans. Image Process.*, vol. 22, pp. 2138–2150, Jun. 2013.

[17] S. Roth and M. J. Black, "Fields of experts," *International Journal of Computer Vision*, vol. 82, pp. 205–229, 2009.

[18] Y. Chen, R. Ranftl, and T. Pock, "Insights into analysis operator learning: From patch-based sparse models to higher order MRFs," *IEEE Trans. Image Process.*, vol. 23, pp. 1060–1072, Mar. 2014.

[19] Y. Chen and T. Pock, "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, pp. 1256–1272, 2016.

[20] J.-F. Cai, H. Ji, Z. Shen, and G.-B. Ye, "Data-driven tight frame construction and image denoising," *Applied and Computational Harmonic Analysis*, vol. 37, pp. 89–105, 2014.

[21] M. A. T. Figueiredo, "Synthesis versus analysis in patch-based image priors," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2017, pp. 1338–1342.

[22] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2010, pp. 2528–2535.

[23] V. Papyan, J. Sulam, and M. Elad, "Working locally thinking globally: Theoretical guarantees for convolutional sparse coding," *IEEE Transactions on Signal Processing*, vol. 65, pp. 5687–5701, 2017.

[24] B. Wohlberg, "Efficient algorithms for convolutional sparse representations," *IEEE Trans. Image Process.*, vol. 25, pp. 301–315, 2016.

[25] C. Garcia-Cardona and B. Wohlberg, "Convolutional dictionary learning: a comparative review and new algorithms," *IEEE Transactions on Computational Imaging*, vol. 4, pp. 366–381, 2018.

[26] S. Muramatsu, "Structured dictionary learning with 2-D non-separable oversampled lapped transform," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014.

[27] S. Muramatsu, M. Ishii, and Z. Chen, "Efficient parameter optimization for example-based design of nonseparable oversampled lapped transform," in *2016 IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 3618–3622.

[28] S. Muramatsu, K. Furuya, and N. Yuki, "Multidimensional nonseparable oversampled lapped transforms: Theory and design," *IEEE Transactions on Signal Processing*, vol. 65, pp. 1251–1264, 2017.

[29] G. Peyré and J. M. Fadili, "Learning analysis sparsity priors," in *The 9th International Conference on Sampling Theory and Applications (SampTA)*, 2011.

[30] Y. Chen, T. Pock, and H. Bischof, "Learning ℓ_1-based analysis and synthesis sparsity priors using bi-level optimization," in *Workshop on Analysis Operator Learning vs Dictionary Learning, NIPS 2012*, 2012.

[31] O. Christensen, *An introduction to frames and Riesz bases*. Birkhäuser, 2003.

[32] P. Vaidyanathan, *Multirate systems and filter banks*. Prentice Hall, 1992.

[33] M. N. Do, "Multidimensional filter banks and multiscale geometric representations," *Foundations and Trends in Signal Processing*, vol. 5, pp. 157–164, 2012.

[34] Z. Cvetkovic and M. Vetterli, "Oversampled filter banks," *IEEE Trans. Signal Process.*, vol. 46, pp. 1245–1255, May 1998.

[35] G. Strang and T. Nguyen, *Wavelets and Filter Banks*. Wellesley College, 1996.

[36] H. Bolcskei, F. Hlawatsch, and H. Feichtinger, "Frame-theoretic analysis of oversampled filter banks," *IEEE Trans. Signal Process.*, vol. 46, pp. 3256–3268, 1998.

[37] J. K. Martin Vetterli, *Wavelets and subband coding*. Prentice-Hall, 1995.

[38] S. Venkataraman and B. Levy, "State space representations of 2-D FIR lossless transfer matrices," *IEEE Trans. Circuits Syst. II*, vol. 41, pp. 117–132, 1994.

[39] J. Zhou, M. Do, and J. Kovacevic, "Multidimensional orthogonal filter bank characterization and design using the Cayley transform," *IEEE Trans. Image Process.*, vol. 14, pp. 760–769, 2005.

[40] F. Delgosha and F. Fekri, "Results on the factorization of multidimensional matrices for paraunitary filterbanks over the complex field," *IEEE Trans. Signal Process.*, vol. 52, pp. 1289–1303, 2004.

[41] H. S. Malvar, *Signal Processing with Lapped Transforms*. Artech Print on Demand, 1992.

[42] A. D. Poularikas, Ed., *Transforms and applications handbook*. CRC press, 2010.

[43] V. Goyal, M. Vetterli, and N. Thao, "Quantized overcomplete expansions in R^N: analysis, synthesis, and algorithms," *IEEE Trans. Inf. Theory*, vol. 44, pp. 16–31, 1998.

[44] R. R. Coifman and D. L. Donoho, "Translation-invariant de-noising." Springer-Verlag, 1995, pp. 125–150.

[45] K. G. Murty and S. N. Kabadi, "Some NP-complete problems in quadratic and nonlinear programming," *Mathematical Programming*, vol. 39, pp. 117–129, 1987.

[46] P. A. Parrilo, "Semidefinite programming relaxations for semialgebraic problems," *Mathematical Programming*, vol. 96, pp. 293–320, 2003.

[47] L. Pfister and Y. Bresler, "Bounding multivariate trigonometric polynomials with applications to filter bank design," *CoRR*, 2018, `arXiv:1802.09588 [eess.SP]`.

[48] ——, "Bounding extremal values of multivariate trigonometric polynomials," *IEEE Trans. Signal Process.*, Submitted.

[49] H.-Y. Gao and A. G. Bruce, "Waveshrink with firm shrinkage," *Statistica Sinica*, pp. 855–874, 1997.

[50] R. Chartrand, "Shrinkage mappings and their induced penalty functions," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 1026–1029.

[51] ——, "Fast algorithms for nonconvex compressive sensing: MRI reconstruction from very few data," in *2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, 2009, pp. 262–265.

TABLE II: Reconstruction PSNR for test images averaged over 10 noise realizations. The column **mean** reports the mean PSNR over the set of test images. The highest PSNR in each column shaded gray and in bold; the second highest result is shaded gray. FBST-128-16 indicates a filter bank sparsifying transform with 128 channels and 16×16 filters and denoised according to (33). The -I suffix indicates denoising with the iterative Algorithm 2. $\mathrm{CSF}_{7\times7}$ indicates a cascaded shrinkage field with 7×7 filters. $\mathrm{FoE}_{3\times3}$ denotes the Field of Experts using 3×3 filters. NSOLT-D2-C48-O2 indicates an NSOLT with downsampling by 2, 48 channels, and polyphase order 2.

σ	10	20	30	10	20	30	10	20	30	10	20	30	10	20	30	10	20	30
Input PSNR	28.1	22.1	18.6	28.1	22.1	18.6	28.1	22.1	18.6	28.1	22.1	18.6	28.1	22.1	18.6	28.1	22.1	18.6
Method	**mean**			**baboon**			**barbara**			**boat**			**man**			**peppers**		
WNNM	**33.8**	**30.6**	**28.8**	**30.7**	**26.7**	24.6	**35.4**	**32.2**	**30.3**	**34.1**	**31.0**	**29.1**	**34.2**	**30.7**	**28.9**	**34.8**	**32.6**	**31.1**
BM3D	33.6	30.4	28.6	30.5	26.5	24.4	35.0	31.7	29.8	33.9	30.8	29.0	34.0	30.6	28.8	**34.8**	32.5	31.0
STROLLR	33.6	30.4	28.6	30.5	26.6	**24.7**	35.1	31.9	29.9	33.8	30.7	28.9	33.8	30.5	28.7	**34.8**	32.4	30.9
EPLL	33.3	29.9	28.1	30.5	26.6	24.6	33.6	29.7	27.5	33.6	30.6	28.8	33.9	30.6	28.8	34.6	32.3	30.6
$\mathrm{CSF}_{5\times5}$	33.1	29.7	27.7	30.3	26.3	24.1	33.4	29.3	26.8	33.6	30.5	28.6	33.8	30.4	28.6	34.6	32.0	30.3
$\mathrm{CSF}_{7\times7}$	33.1	29.7	27.7	30.4	26.3	24.2	33.4	29.3	26.8	33.5	30.4	28.6	33.7	30.4	28.6	34.6	32.1	30.3
$\mathrm{FOE}_{3\times3}$	32.8	29.1	27.1	30.1	25.9	23.7	32.6	28.1	25.5	33.3	30.1	28.2	33.5	30.0	28.2	34.3	31.5	29.8
PBST-64-8	33.1	29.5	27.5	30.3	26.0	23.8	34.0	30.0	27.8	33.5	30.1	28.2	33.4	29.8	28.0	34.4	31.7	29.9
PBST-256-16	33.3	29.7	27.7	30.4	26.1	23.9	34.2	30.3	28.0	33.6	30.2	28.2	33.6	30.0	28.0	34.6	31.9	30.1
PBST-64-8-I	33.2	29.8	27.8	30.3	26.1	24.1	34.1	30.3	28.0	33.6	30.3	28.4	33.6	30.1	28.3	34.6	32.0	30.4
PBST-256-16-I	33.4	29.9	27.9	30.4	26.4	24.2	34.2	30.5	28.2	33.7	30.4	28.4	33.8	30.2	28.4	34.7	32.2	30.5
FBST-64-8	33.3	29.8	27.8	30.3	26.1	23.9	34.4	30.5	28.2	33.7	30.3	28.3	33.7	30.1	28.2	34.5	31.9	30.2
FBST-128-8	33.3	29.8	27.8	30.4	26.1	24.0	34.3	30.5	28.3	33.7	30.3	28.4	33.7	30.1	28.3	34.6	31.9	30.2
FBST-196-8	33.3	29.8	27.8	30.4	26.2	24.0	34.3	30.5	28.2	33.7	30.3	28.4	33.7	30.2	28.2	34.5	31.9	30.1
FBST-64-16	33.3	29.8	27.8	30.2	26.0	23.9	34.5	30.8	28.6	33.6	30.2	28.2	33.5	30.0	28.1	34.4	31.9	30.1
FBST-128-16	33.3	29.9	28.0	30.3	26.1	24.0	34.6	31.0	28.8	33.6	30.3	28.4	33.6	30.1	28.2	34.6	32.0	30.4
FBST-196-16	33.4	30.0	28.0	30.3	26.1	24.0	34.7	31.1	29.0	33.7	30.3	28.4	33.6	30.1	28.2	34.6	32.0	30.4
FBST-64-8-I	33.4	30.0	28.1	30.4	26.3	24.2	34.4	30.7	28.5	33.8	30.5	28.6	33.8	30.3	28.5	34.7	32.2	30.6
FBST-128-8-I	33.4	30.0	28.1	30.4	26.4	24.3	34.3	30.7	28.5	33.8	30.5	28.7	33.8	30.3	28.4	34.7	32.2	30.6
FBST-196-8-I	33.4	29.9	28.0	30.5	26.4	24.2	34.3	30.6	28.3	33.7	30.4	28.5	33.8	30.3	28.4	34.6	32.1	30.5
FBST-64-16-I	33.4	30.1	28.1	30.4	26.3	24.2	34.6	31.1	29.0	33.8	30.4	28.5	33.6	30.3	28.5	34.7	32.2	30.6
FBST-128-16-I	33.4	30.1	28.3	30.4	26.4	24.3	34.7	31.2	29.2	33.7	30.6	28.7	33.7	30.2	28.5	34.7	32.3	30.7
FBST-196-16-I	33.5	30.2	28.3	30.4	26.3	24.4	34.8	31.4	29.3	33.8	30.5	28.7	33.7	30.3	28.4	34.7	32.3	30.8
NSOLT-D4-C12-O2	30.3	26.0	23.8	29.1	24.3	21.9	30.1	25.7	23.4	30.4	26.2	24.1	30.5	26.6	24.5	31.2	27.1	24.9
NSOLT-D2-C12-O2	30.9	26.8	24.7	29.3	24.6	22.4	30.8	26.4	24.2	31.2	27.3	25.2	31.3	27.3	25.3	32.0	28.3	26.2
NSOLT-D4-C12-O4	31.0	26.8	24.7	29.3	24.6	22.4	30.9	26.5	24.3	31.2	27.3	25.2	31.3	27.3	25.3	32.1	28.4	26.3
NSOLT-D4-C24-O2	30.8	26.6	24.5	29.2	24.5	22.3	30.9	26.7	24.4	30.9	26.9	24.8	31.0	27.0	25.0	31.8	27.9	25.9
NSOLT-D4-C24-O4	31.1	27.0	24.9	29.3	24.7	22.5	31.0	26.7	24.4	31.4	27.5	25.5	31.4	27.6	25.6	32.2	28.6	26.6
NSOLT-D2-C12-O4	29.9	25.6	23.4	28.9	24.1	21.7	29.8	25.4	23.2	30.0	25.8	23.7	30.1	26.0	24.1	30.8	26.6	24.5

[52] 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.

[53] L. Pfister and Y. Bresler, "Automatic parameter tuning for image denoising with learned sparsifying transforms," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[54] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *Proceedings of the 21th International Conference on Artificial Neural Networks*. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 52–59.

[55] M. Tsatsanis and G. Giannakis, "Principal component filter banks for optimal multiresolution analysis," *IEEE Trans. Signal Process.*, vol. 43, pp. 1766–1777, 1995.

[56] P. Moulin and M. Mihcak, "Theory and design of signal-adapted FIR paraunitary filter banks," *IEEE Trans. Signal Process.*, vol. 46, pp. 920–929, 1998.

[57] B. Xuan and R. Bamberger, "Multi-dimensional, paraunitary principal component filter banks," in *1995 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 2, 1995, pp. 1488–1491.

[58] ——, "FIR principal component filter banks," *IEEE Trans. Signal Process.*, vol. 46, pp. 930–940, 1998.

[59] M. A. Unser, "Extension of the Karhunen-Loeve transform for wavelets and perfect reconstruction filterbanks," in *Mathematical Imaging: Wavelet Applications in Signal and Image Processing*, Nov. 1993, pp. 45–56.

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

[61] R. H. Chan, J. G. Nagy, and R. J. Plemmons, "Circulant preconditioned Toeplitz least squares iterations," *SIAM Journal on Matrix Analysis and Applications*, vol. 15, pp. 80–97, Jan. 1994.

[62] T. Strohmer, *Finite-and Infinite-Dimensional Models for Oversampled Filter Banks*. Boston, MA: Birkhäuser Boston, 2001, pp. 293–315.

[63] B. Sharif and Y. Bresler, "Generic feasibility of perfect reconstruction with short FIR filters in multichannel systems," *IEEE Transactions on Signal Processing*, vol. 59, pp. 5814–5829, 2011.

[64] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, pp. 183–202, Jan. 2009.

[65] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *IEEE Trans. Image Process.*, vol. 16, pp. 2080–2095, 2007.

[66] S. Gu, L. Zhang, W. Zuo, and X. Feng, "Weighted nuclear norm minimization with application to image denoising," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2014, pp. 2862–2869.

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

[68] U. Schmidt, Q. Gao, and S. Roth, "A generative perspective on MRFs in low-level vision," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010, pp. 1751–1758.

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

[70] ——, "The power of complementary regularizers: Image recovery via transform learning and low-rank modeling," *CoRR*, 2018, `arXiv:1808.01316 [cs.CV]`.

[71] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, pp. 600–12, Apr. 2004.

[72] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. 8th Int'l Conf. Computer Vision*, vol. 2, Jul. 2001, pp. 416–423.