Seismic data interpolation and denoising by learning a tensor tight frame

Lina Liu[1,2] **Gerlind Plonka**[2] **Jianwei Ma**[1]

1,Department of Mathematics,Harbin Institute of Technology, Harbin, China
2,Institute for Numerical and Applied Mathematics,University of Göttingen, Göttingen, Germany, UK

E-mail: jma@hit.edu.cn

May 2017

Abstract. Seismic data interpolation and denoising plays a key role in seismic data processing. These problems can be understood as sparse inverse problems, where the desired data are assumed to be sparsely representable within a suitable dictionary.

In this paper, we present a new method based on a data-driven tight frame of Kronecker type (KronTF) that avoids the vectorization step and considers the multidimensional structure of data in a tensor-product way. It takes advantage of the structure contained in all different modes (dimensions) simultaneously. In order to overcome the limitations of a usual tensor-product approach we also incorporate data-driven directionality (KronTFD). The complete method is formulated as a sparsity-promoting minimization problem. It includes two main steps. In the first step, a hard thresholding algorithm is used to update the frame coefficients of the data in the dictionary. In the second step, an iterative alternating method is used to update the tight frame (dictionary) in each different mode. The dictionary that is learned in this way contains the principal components in each mode. Furthermore, we apply the proposed tight frames of Kronecker type to seismic interpolation and denoising. Examples with synthetic and real seismic data show that the proposed method achieves better results than the traditional projection onto convex sets (POCS) method based on the Fourier transform and the previous vectorized data-driven tight frame (DDTF) methods. In particular, the simple structure of the new frame construction makes it essentially more efficient.

1. Introduction

Seismic data processing is the essential bridge connecting the seismic data acquisition and interpretation. Many processes benefit from fully sampled seismic volumes. Examples of the latter are multiple suppression, migration, amplitude versus offset analysis, and shear wave splitting analysis. However, sampling is often limited by the high cost of the acquisition, obstacles in the field, and sampled data contain missing traces.

Various interpolation methods have been proposed to handle the arising aliasing effects. We especially refer to the f-x domain prediction method by Spitz [30] that is based on predictability of linear events without a priori knowledge of the directions of lateral coherence of the events. This approach can be also extended to 3D regular data.

Interpolating 3D data on a regular grid, but with an irregular pattern of missing traces, is more delicate. In that case undesired attenuate artifacts can arise from improper wave field sampling.

However, because of its practical importance, interpolation and denoising of irregular missing seismic data have become an important topic for the seismic data-processing community.

Seismic data interpolation and data denoising can also be regarded as an inverse problem where we employ sparsity constraints. Using the special structure of seismic data, we want to exploit that it can be represented sparsely in a suitable basis or frame, i.e., the original signal can be well approximated using only a small number of significant frame coefficients. In this paper, we restrict ourselves to finite-dimensional tight frames. Generalizing the idea of orthogonal transforms, a frame transform is determined by a transform matrix $D \in \mathbb{C}^{M_1 \times N_1}$ with $M_1 \geq N_1$, such that for given data $y \in \mathbb{C}^{N_1}$ the vector $u = Dy \in \mathbb{C}^{M_1}$ contains the frame coefficients. For tight frames, D possesses a left inverse of the form $\frac{1}{c}D^* = \frac{1}{c}\overline{D}^T \in \mathbb{C}^{N_1 \times M_1}$, i.e., $D^*D = cI$, where I is the identity operator. The constant c is the frame bound that can be set to 1 by suitable normalization. In this case the tight frame is called Parseval frame, and we assume in the following that $c = 1$.

The problem of finding a sparse representation of the data y can be formulated as the following ℓ_0-subnorm [10] minimization

$$\min_u ||u||_0, \qquad s.t. \qquad y = D^*u, \tag{1.1}$$

where $||u||_0$ represents the number of nonzero elements of u.

Since the ℓ_0-norm minimization is an NP-hard problem for computation, it is often replaced by the ℓ_1-norm minimization for its relaxation

$$\min_u ||u||_1, \qquad s.t. \qquad y = D^*u, \tag{1.2}$$

where $||u||_1$ stands for the sum of the absolute values of nonzero elements of u.

The idea to employ sparsity of the data in a certain basis or frame has been already used in seismic data processing. A fundamental question is here the choice of the dictionary transform D.

Dictionary transforms can be mainly divided into two categories: fixed-basis/frame transforms and adaptive learning dictionaries. Transforms in the first category are data independent and include e.g. Radon transform [32, 21], Fourier transform [28, 25, 38, 2, 33], curvelet transform [17, 29, 16, 27], shearlet transform [20], and others.

In recent years, adaptive learning dictionary methods came up for data processing. The corresponding transforms are data dependent and therefore usually more expensive. On the other hand they often achieve essentially better results.

Transforms of this type include Principal Component Analysis (PCA) [18] and generalized PCA [37], the method of optimal directions (MOD) [13, 3], the K-SVD method (K-mean singular value decomposition) [1], and others. Particularly, K-SVD is a very popular tool for training the dictionary. However, the K-SVD method treats the given training data as a signal vector and in each step uses the SVD decomposition of the arising transform matrix to update each column of the dictionary. By the vectorization of the data, the transform matrix is huge such that this approach requires a large computational effort. In order to reduce the computational complexity, a new method, named data-driven tight frame (DDTF) has been proposed in [8], which learns a dictionary with a prescribed block Toeplitz structure. The DDTF method updates the complete transform matrix by one SVD decomposition in one iteration step in contrast to the K-SVD that employs an SVD decomposition to update each column of the transform matrix consecutively. The DDTF method has also been applied to seismic data interpolation and denoising of two-dimensional and high-dimensional seismic data [24, 40].

Fixed dictionary methods enjoy the high efficiency advantage, while adaptive learning dictionary methods can take use of the information of data itself. By combining the seislets [14] and DDTF, a double sparsity method was proposed for seismic data processing [39], which can benefit from both fixed and learned dictionary methods. However, a vast majority of existing dictionary learning methods deals with vectors and therefore does not fully exploit the data structure, i.e., spatial correlation of the data pixels.

To overcome this problem, a Kronecker-based dictionary has been applied to dynamic computed tomography [31] treating the input data as tensors instead of vectors and applying a learned dictionary based on tensor decomposition. The tensor decomposition has been also used for seismic interpolation [22].

Generalizing these ideas, in this paper we present a data-driven tight frame construction of Kronecker type (KronTF), that inherits the simple tensor-product structure to ensure fast computation also for 3D tensors. This approach is further generalized by incorporating adaptive directionality (KronTFD).

The separability of dictionary atoms makes the resulting dictionaries highly scalable. The orthonormality among dictionary atoms leads to very efficient sparse coding computation, as each sub-problem encountered during the iterations for solving the resulting optimization problem has a simple closed-form solution. These two characteristics, i.e., the computational efficiency and scalability, make the proposed method very suitable for processing tensor data. The main contribution of our work includes the following aspects: 1) To avoid the vectorization step in previous DDTF method, we unfold the seismic data in each mode and use one SVD of small size per iteration to learn the dictionary. Therefore, the dictionary contains structures of data sets in different modes. 2) Further, we introduce a simple method to incorporate significant directions of the data structure into the dictionary. 3) Finally the proposed new data-driven dictionaries KronTF and KronTFD are applied to seismic

data interpolation and denoising.

This paper is organized as follows. We first introduce the notation borrowed from tensor algebra and definitions pertaining tensors that are used throughout our paper. In Section 2, we shortly recall the basic idea of construction a data-driven frame and present an iterative scheme that is based on alternating updates of the frame coefficients for sparse representation of the given training data on the one hand and of the data-dependent frame matrix on the other hand. Section 3 is devoted to the construction of a new data-driven frame that is based on the tensor-product structure, and is therefore essentially cheaper to learn. The employed idea can be simply transferred to the case of 3-tensors. In order to make the construction more flexible, we extend the data-driven Kronecker frame construction by incorporating also directionality in Section 4. The two data-driven dictionaries are applied to data interpolation/reconstruction and to denoising in Section 5, and the corresponding numerical tests for synthetic 2D and 3D data and real 2D and 3D data are presented in Section 6. The paper finishes with a short conclusion on the achievements in this paper and further open problems.

1.1. Notations

In this paper, we denote vectors by lowercase letters and matrices by uppercase letters. For $x = (x_i)_{i=1}^N \in \mathbb{C}^N$ let $\|x\|_2 := \left(\sum_{i=1}^N |x_i|^2\right)^{1/2}$ be the Euclidean norm, and for $X = (x_{i,j})_{i=1,j=1}^{N_1,N_2} \in \mathbb{C}^{N_1\times N_2}$ we denote by $\|X\|_F = \left(\sum_{i=1}^{N_1}\sum_{j=1}^{N_2} |x_{i,j}|^2\right)^{1/2}$ the Frobenius norm. For a third order tensor $\mathcal{X} = (x_{i,j,k})_{i=1,j=1,k=1}^{N_1,N_2,N_3} \in \mathbb{C}^{N_1\times N_2\times N_3}$ we introduce the three matrix unfoldings that involve the tensor dimensions N_1, N_2 and N_3 in a cyclic way,

$$\begin{aligned}
\mathcal{X}_{(1)} &:= \left((x_{i,j,1})_{i=1,j=1}^{N_1,N_2}, (x_{i,j,2})_{i=1,j=1}^{N_1,N_2}, \ldots, (x_{i,j,N_3})_{i=1,j=1}^{N_1,N_2}\right) \in \mathbb{C}^{N_1\times N_2N_3},\\
\mathcal{X}_{(2)} &:= \left((x_{1,j,k})_{j=1,k=1}^{N_2,N_3}, (x_{2,j,k})_{j=1,k=1}^{N_2,N_3}, \ldots, (x_{N_1,j,k})_{j=1,k=1}^{N_2,N_3}\right) \in \mathbb{C}^{N_2\times N_1N_3},\\
\mathcal{X}_{(3)} &:= \left((x_{i,1,k})_{k=1,i=1}^{N_3,N_1}, (x_{i,2,k})_{k=1,i=1}^{N_3,N_1}, \ldots, (x_{i,N_2,k})_{k=1,i=1}^{N_3,N_1}\right) \in \mathbb{C}^{N_3\times N_1N_2}.
\end{aligned}$$

The multiplication of a tensor $\mathcal{X}$ with a matrix is defined by employing the ν-mode product, $\nu = 1, 2, 3$, as defined in [23]. For $\mathcal{X} \in \mathbb{C}^{N_1\times N_2\times N_3}$ and matrices $V^{(1)} = (v^{(1)}_{i',i})_{i'=1,i=1}^{M_1,N_1} \in \mathbb{C}^{M_1\times N_1}$, $V^{(2)} = (v^{(2)}_{j',j})_{j'=1,j=1}^{M_2,N_2} \in \mathbb{C}^{M_2\times N_2}$, and $V^{(3)} = (v^{(3)}_{k',k})_{k'=1,k=1}^{M_3,N_3} \in \mathbb{C}^{M_3\times N_3}$ we define

$$\begin{aligned}
\mathcal{X} \times_1 V^{(1)} &:= \left(\sum_{i=1}^{N_1} x_{i,j,k}\, v^{(1)}_{i',i}\right)_{i'=1,j=1,k=1}^{M_1,N_2,N_3} \in \mathbb{C}^{M_1\times N_2\times N_3},\\
\mathcal{X} \times_2 V^{(2)} &:= \left(\sum_{j=1}^{N_2} x_{i,j,k}\, v^{(2)}_{j',j}\right)_{i=1,j'=1,k=1}^{N_1,M_2,N_3} \in \mathbb{C}^{N_1\times M_2\times N_3},\\
\mathcal{X} \times_3 V^{(3)} &:= \left(\sum_{k=1}^{N_3} x_{i,j,k}\, v^{(3)}_{k',k}\right)_{i=1,j=1,k'=1}^{N_1,N_2,M_3} \in \mathbb{C}^{N_1\times N_2\times M_3}.
\end{aligned}$$

Generalizing the usual singular value decomposition for matrices it has been shown in [23] that every $(N_1 \times N_2 \times N_3)$-tensor $\mathcal{X}$ can be written as a product

$$\mathcal{X} = \mathcal{S} \times_1 V^{(1)} \times_2 V^{(2)} \times_3 V^{(3)}$$

with unitary matrices $V^{(1)} \in \mathbb{C}^{N_1 \times N_1}$, $V^{(2)} \in \mathbb{C}^{N_2 \times N_2}$, and $V^{(3)} \in \mathbb{C}^{N_3 \times N_3}$ and with a sparse core tensor $\mathcal{S} \in \mathbb{C}^{N_1 \times N_2 \times N_3}$. By unfolding, this factorization can be represented as

$$\begin{aligned} \mathcal{X}_{(1)} &= V^{(1)} \mathcal{S}_{(1)} (V^{(3)} \otimes V^{(2)})^T, \\ \mathcal{X}_{(2)} &= V^{(2)} \mathcal{S}_{(2)} (V^{(1)} \otimes V^{(3)})^T, \\ \mathcal{X}_{(3)} &= V^{(3)} \mathcal{S}_{(3)} (V^{(2)} \otimes V^{(1)})^T, \end{aligned} \tag{1.3}$$

where the $\mathcal{S}_{(\nu)}$ denote the corresponding ν-modes of the tensor $\mathcal{S}$. Further, $\otimes$ denotes the Kronecker product, see [5], which is for two matrices $A = (a_{i,j})_{i=1,j=1}^{M,N} \in \mathbb{C}^{M \times N}$ and $B \in \mathbb{C}^{L \times K}$ defined as

$$A \otimes B = (a_{i,j} B)_{i=1,j=1}^{M,N} \in \mathbb{C}^{ML \times NK}.$$

In order to compute a generalized SVD for the tensor $\mathcal{X}$ one may first consider the SVDs of the unfolded matrices $\mathcal{X}_{(1)}$, $\mathcal{X}_{(2)}$ and $\mathcal{X}_{(3)}$. Interpreting the equations in (1.3) as SVD decompositions, we obtain the ν-mode orthogonal matrices $V^{(\nu)}$, $\nu = 1, 2, 3$, directly from the left singular matrices of these SVDs, see [23].

In this paper we consider a generalization of the idea of SVD and are interested in a dictionary decomposition of a finite set of tensors $\mathcal{X}_k$, $k = 1, \ldots, K$, i.e., we want to derive dictionary matrices $D_1 \in \mathbb{C}^{M_1 \times N_1}$, $D_2 \in \mathbb{C}^{M_2 \times N_2}$, and $D_3 \in \mathbb{C}^{M_3 \times N_3}$ with $M_\nu \geq N_\nu$, $\nu = 1, 2, 3$, such that

$$\mathcal{X}_k \approx \mathcal{U}_k \times_1 D_1^* \times_2 D_2^* \times_3 D_3^*.$$

That means, we want to find a dictionary such that $\mathcal{U}_k \times_1 D_1 \times_2 D_2 \times_3 D_3$ is a good approximation of $\mathcal{X}_k$ such that the core tensors $\mathcal{U}_k \in \mathbb{C}^{M_1 \times M_2 \times M_3}$ are all simultaneously sparse. For this purpose, we want to iteratively improve a fixed initial dictionary depending on the data tensors $\mathcal{X}_k$, $k = 1, \ldots, K$ in Section 3. Moreover, generalizing this tensor product structure, we will propose to incorporate directional adaptivity by multiplication with data-driven block permutations in Section 4.

2. Data-driven Parseval frames (DDPF)

First, we want to describe an idea for construction of data-driven Parseval frames for two-dimensional data (images), similarly as proposed in [8, 40]. Generalizing the approach of unitary matrices, we say that a matrix $D \in \mathbb{C}^{M_1 \times N_1}$, $M_1 \geq N_1$, represents a Parseval frame of $\mathbb{C}^{N_1}$, if $D^* D = I_{N_1}$, i.e., $D^* = \overline{D}^T$ is the Moore Penrose inverse of D. Obviously, each vector $a \in \mathbb{C}^{N_1}$ can be represented as a linear combination of the M_1 columns of D^*, i.e., there exists a vector $c \in \mathbb{C}^{M_1}$ with $a = D^* c$, and for $M_1 > N_1$, the coefficient vector is usually not longer uniquely defined. One canonical

choice for c would be $c = Da$, since obviously $D^*c = D^*Da = a$ is true. In frame theory, D^* is called synthesis operator while D is the analysis operator. Within the last time, many frame transforms have been developed for signal and image processing, as e.g. wavelet framelets, curvelet frames, shearlet frames etc. In [8] a data-driven tight frame construction has been proposed for image denoising, where the frame matrix D^* is composed of block-wise Toeplitz matrices being defined by suitable initial filter sequences.

We adopt this idea and consider the following model. Assume that we have given a series of images $Y_1, \dots, Y_K$ that are vectorized to $y_1, \dots, y_K \in \mathbb{C}^{N_1}$. We now aim at finding a sparse dictionary representation for $y_1, \dots, y_K$ simultaneously. Let $\mathbf{Y} = (y_1 y_2 \dots y_K) \in \mathbb{C}^{N_1 \times K}$ and $\mathbf{C} = (c_1 c_2 \dots c_K) \in \mathbb{C}^{M_1 \times K}$ with columns $c_k \in \mathbb{C}^{M_1}$ be the matrix of arising dictionary coefficients. We consider the model

$$\min_{D \in \mathbb{C}^{M_1 \times N_1}, \mathbf{C} \in \mathbb{C}^{M_1 \times K}} \|D\mathbf{Y} - \mathbf{C}\|_F^2 + \lambda \|\mathbf{C}\|_\star \quad \text{s.t.} \quad D^*D = I_{N_1}, \tag{2.1}$$

where $\|\mathbf{C}\|_\star$ denotes either $\|\mathbf{C}\|_0$, the number of nonzero entries of $\mathbf{C}$, or $\|\mathbf{C}\|_1$, the sum of all absolute values of entries in $\mathbf{C}$. The parameter $\lambda > 0$ balances the approximation term and the sparsity term. Observe that in the above model not only $\mathbf{C}$, the matrix of sparse vectors c_k representing y_k in the dictionary domain, but also D has to be learned, where the side condition $D^*D = I_N$ ensures that D is a Parseval frame.

To solve this optimization problem, we employ an alternating iterative scheme that in turns updates the coefficient matrix $\mathbf{C}$ and the frame matrix D, where we start with a suitable initial frame $D = D^0$.

Step 1. For a fixed frame $D \in \mathbb{C}^{M_1 \times N_1}$ we have to solve the problem

$$\min_{\mathbf{C} \in \mathbb{C}^{M_1 \times K}} \|D\mathbf{Y} - \mathbf{C}\|_F^2 + \lambda \|\mathbf{C}\|_\star. \tag{2.2}$$

For $\|\cdot\|_1$ we obtain the solution $\mathbf{C} = S_{\lambda/2}(D\mathbf{Y})$, applied componentwisely, with the soft threshold operator

$$S_{\lambda/2}(x) = \begin{cases} (\operatorname{sign} x)(|x| - \lambda/2) & \text{for } |x| > \lambda/2, \\ 0 & \text{for } |x| \le \lambda/2. \end{cases} \tag{2.3}$$

For for $\|\cdot\|_0$ we obtain $\mathbf{C} = T_\gamma(D\mathbf{Y})$, applied componentwisely, with the hard threshold operator

$$T_\gamma(x) = \begin{cases} x & \text{for } |x| > \gamma, \\ 0 & \text{for } |x| \le \gamma, \end{cases}$$

where the threshold parameter γ depends on λ and on the data $D\mathbf{Y}$.

Remark 2.1 *The above model* (2.2) *is the so-called syntheses problem that is usually simpler to solve. The corresponding analysis problem reads*

$$\min_{\mathbf{C} \in \mathbb{C}^{M_1 \times K}} \|\mathbf{Y} - D^*\mathbf{C}\|_F^2 + \lambda \|\mathbf{C}\|_\star. \tag{2.4}$$

For the analysis approach, the solutions obtained by applying the threshold functions do not longer exactly minimize (2.4) *but can be seen as good approximations, see e.g. [12]. In case of unitary matrices D, the two approaches coincide.*

Step 2. For given $\mathbf{C}$ we now want to update the frame matrix D by solving the optimization problem

$$\min_{D\in\mathbb{C}^{M_1\times N_1}} \|D\mathbf{Y}-\mathbf{C}\|_F^2 \qquad \text{s.t.} \quad D^*D = I_{N_1}. \tag{2.5}$$

Similarly as in [41] we can show

Theorem 2.2 *The optimization problem* (2.5) *is equivalent to the problem*

$$\max_{D\in\mathbb{C}^{M_1\times N_1}} \mathrm{Re}\,(\mathrm{tr}\,(D\mathbf{Y}\mathbf{C}^*)) \qquad \text{s.t.} \quad D^*D = I_{N_1},$$

where tr *denotes the trace of a matrix, i.e., the sum of its diagonal entries, and* Re *the real part of a complex number. If* $\mathbf{Y}\mathbf{C}^*$ *has full rank* N_1*, this optimization problem is solved by* $D_{opt} = V_{N_1}U^*$*, where* $U\Lambda V^*$ *denotes the singular value decomposition of* $\mathbf{Y}\mathbf{C}^* \in \mathbb{C}^{N_1\times M_1}$ *with the unitary matrices* $U \in \mathbb{C}^{N_1\times N_1}$*,* $V \in \mathbb{C}^{M_1\times M_1}$*, and* $V_{N_1} \in \mathbb{C}^{M_1\times N_1}$ *is the restriction of* V *to its first* N_1 *columns.*

Proof 2.3 We give the short proof for the complex case that is not contained in [41]. The objective function in (2.5) can be rewritten as

$$\begin{aligned}\|D\mathbf{Y}-\mathbf{C}\|_F^2 &= \mathrm{tr}(((D\mathbf{Y})^*-\mathbf{C}^*)(D\mathbf{Y}-\mathbf{C}))\\ &= \mathrm{tr}(\mathbf{Y}^*D^*D\mathbf{Y}+\mathbf{C}^*\mathbf{C}-\mathbf{C}^*D\mathbf{Y}-\mathbf{Y}^*D^*\mathbf{C})\\ &= \mathrm{tr}(\mathbf{Y}^*\mathbf{Y})+\mathrm{tr}(\mathbf{C}^*\mathbf{C})-2\mathrm{Re}(\mathrm{tr}(D\mathbf{Y}\mathbf{C}^*)),\end{aligned}$$

where we have used that $\mathrm{tr}(D\mathbf{Y}\mathbf{C}^*) = \mathrm{tr}(\mathbf{C}^*D\mathbf{Y}) = \overline{\mathrm{tr}(\mathbf{Y}^*D^*\mathbf{C})}$. Thus (2.5) is equivalent to

$$\max_{D\in\mathbb{C}^{M_1\times N_1}} \mathrm{Re}\,(\mathrm{tr}\,(D\mathbf{Y}\mathbf{C}^*)) \qquad \text{s.t.} \quad D^*D = I_{N_1}.$$

Let now the singular value decomposition of $\mathbf{Y}\mathbf{C}^*$ be given by

$$\mathbf{Y}\mathbf{C}^* = U\Lambda V^*$$

with unitary matrices $U \in \mathbb{C}^{N_1\times N_1}$, $V \in \mathbb{C}^{M_1\times M_1}$, and a diagonal matrix of (real nonnegative) singular values $\Lambda = (\mathrm{diag}(\lambda_1,\dots,\lambda_{N_1}),\mathbf{0}) \in \mathbb{R}^{N_1\times M_1}$. Choosing $D_{opt} := V_{N_1}U^*$, where V_{N_1} is the restriction of V to its first N_1 columns, we simply observe that

$$\mathrm{tr}(D_{opt}\mathbf{Y}\mathbf{C}^*) = \mathrm{tr}(V_{N_1}U^*U\Lambda V^*) = \mathrm{tr}(V_{N_1}\Lambda V^*) = \mathrm{tr}(V^*V_{N_1}\Lambda) = \mathrm{tr}\,\Lambda_{N_1} \in \mathbb{R},$$

where $\Lambda_{N_1} = \mathrm{diag}(\lambda_1,\dots,\lambda_{N_1})$ is the restriction of Λ to its first N_1 columns. Moreover,

$$(D_{opt})^*D_{opt} = UV_{N_1}^*V_{N_1}U^* = I_{N_1}.$$

We show now that the value $\mathrm{tr}\,\Lambda_{N_1}$ is indeed maximal. Let $D \in \mathbb{C}^{M_1\times N_1}$ be an arbitrary matrix satisfying $D^*D = I_{N_1}$. Let U be again the unitary matrix from the SVD $\mathbf{Y}\mathbf{C}^* = U\Lambda V^*$, and let $W_{N_1} := DU$. Then the constraint implies that $W_{N_1}^*W_{N_1} = U^*D^*DU = I_{N_1}$. Thus

$$\mathrm{tr}(D\mathbf{Y}\mathbf{C}^*) = \mathrm{tr}(W_{N_1}\Lambda V^*) = \mathrm{tr}(V^*W_{N_1}\Lambda).$$

Since V is unitary and since W_{N_1} contains N_1 orthonormal columns of length M_1 also $V^*W_{N_1}$ consists of N_1 orthonormal columns. Particularly, each diagonal entry of $V^*W_{N_1}$ has modulus smaller than or equal to 1. Thus,

$$\mathrm{Re}\left(\mathrm{tr}(V^*W_{N_1}\Lambda)\right) \leq \mathrm{tr}\Lambda_{N_1},$$

and equality only occurs if $W_{N_1} = V_{N_1}$, i.e., if $D = D_{opt}$. □

Remarks 2.4 *1. In order to obtain the frame matrix uniquely, we need at least a set of $K \geq N_1$ image data, since otherwise the matrix product $\mathbf{Y}\mathbf{C}^*$ cannot have full rank N_1. If indeed* $\mathrm{rank}\,(\mathbf{Y}\mathbf{C}^*) < N_1$ *then Λ does not longer contain N positive singular values. Thus V_{N_1} than also contains singular vectors that correspond to vanishing singular values while the order of these singular vectors in V_{N_1} is not longer fixed.*
2. Again, the above model (2.5) *is the so-called syntheses model. The corresponding analysis model reads*

$$\min_{D\in\mathbb{C}^{N_1\times N}} \|\mathbf{Y} - D^*\mathbf{C}\|_F^2 \qquad \text{s.t.} \quad D^*D = I_N,$$

and it is more difficult to solve. A similar approach as above leads to

$$\begin{aligned}\|\mathbf{Y} - D^*\mathbf{C}\|_F^2 &= \mathrm{tr}((\mathbf{Y} - D^*\mathbf{C})^*)(\mathbf{Y} - D^*\mathbf{C}))\\ &= \mathrm{tr}(\mathbf{Y}^*\mathbf{Y} + \mathbf{C}^*DD^*\mathbf{C} - \mathbf{C}^*D\mathbf{Y} - \mathbf{Y}^*D^*\mathbf{C}^*)\\ &= \mathrm{tr}(\mathbf{Y}^*\mathbf{Y}) + \mathrm{tr}(\mathbf{C}^*DD^*\mathbf{C}) - 2\mathrm{Re}\left(\mathrm{tr}(D\mathbf{Y}\mathbf{C}^*)\right).\end{aligned}$$

Thus, a transfer to the maximization problem in Theorem 2.2 *ignores the term* $\mathrm{tr}(\mathbf{C}^*DD^*\mathbf{C})$ *that still depends on D since we usually do not have $DD^* = I_{M_1}$. Only in case of unitary matrices D the approaches coincide.*
3. The analysis model can be tackled by the K-SVD method that also iterates between improving $\mathbf{C}$ and D, see [1]. Here, for finding an approximation of $\mathbf{C}$, the orthogonal matching pursuit algorithm is employed, while for improving the dictionary matrix D, each column of D^* is updated separately using the singular value decomposition. Since greedy algorithms are employed in this method, there are no guarantees for convergence. However, the approach yields very good results in practice.
4. The iterative method for the synthesis model described above converges in the considered finite-dimensional setting. At each iteration step, the value of the functional $\|\mathbf{C} - D\mathbf{Y}\|_F^2 + \lambda\|\mathbf{C}\|_\star$ decreases monotonously, while it is clearly bounded from below by zero.

3. Data-driven Kronecker tight frame

Considering the above model, we observe that by transforming the discrete images of size $\mathbb{C}^{n_1\times n_2}$ to vectors of length $\mathbb{C}^{N_1}$ with $N_1 = n_1\cdot n_2$, we need to learn a huge dictionary matrix D. Compared to that approach, a usual (tensor-product) two-dimensional transform is of the form

$$C = D_1\, Y\, D_2^T, \tag{3.1}$$

where for a given image $Y \in \mathbb{C}^{n_1 \times n_2}$ we may take dictionary matrices $D_1 \in \mathbb{C}^{m_1 \times n_1}$, $D_2 \in \mathbb{C}^{m_2 \times n_2}$. For a sparsifying basis transform like a DCT, DFT, or an orthogonal wavelet basis transform, D_1 and D_2 are quadratic unitary matrices of the same structure. Here, we relax this approach and consider two rectangular dictionary matrices D_1, D_2 with $m_1 \geq n_1$, and $m_2 \geq n_2$, satisfying $D_1^* D_1 = I_{n_1}$ and $D_2^* D_2 = I_{n_2}$. Employing the properties of the Kronecker tensor product [5], we obtain by vectorization of (3.1)

$$c := \operatorname{vec}(C) = \operatorname{vec}(D_1 Y D_2^T) = (D_2 \otimes D_1) \operatorname{vec}(Y) = (D_2 \otimes D_1)\, y,$$

where the operator vec reshapes the matrix into a vector, and $y := \operatorname{vec}(Y)$.

Now, similarly as in Section 2, we consider the following model. For a sequence of two-dimensional training data $Y_1, \ldots, Y_K \in \mathbb{C}^{n_1 \times n_2}$ and corresponding coefficient matrices $C_1, \ldots, C_K \in \mathbb{C}^{m_1 \times m_2}$ we aim at finding dictionary matrices D_1, D_2 such that

$$\min_{D_1, D_2, C_1, \ldots, C_K} \sum_{k=1}^{K} \left(\|D_1 Y_k D_2^T - C_k\|_F^2 + \lambda \|C_k\|_\star \right) \quad \text{s.t.} \quad D_\nu^* D_\nu = I_{n_\nu},\ \nu = 1, 2. \tag{3.2}$$

Equivalently, using the notations $y_k := \operatorname{vec}(Y_k) \in \mathbb{C}^{n_1 \cdot n_2}$, $c_k := \operatorname{vec}(C_k) \in \mathbb{C}^{m_1 \cdot m_2}$ as well as $\mathbf{Y} := (y_1 \ldots y_K) \in \mathbb{C}^{n_1 n_2 \times K}$ and $\mathbf{C} := (c_1 \ldots c_K) \in \mathbb{C}^{m_1 m_2 \times K}$, we can write the model in the form

$$\min_{D_1, D_2, \mathbf{C}} \|(D_2 \otimes D_1)\mathbf{Y} - \mathbf{C}\|_F^2 + \lambda \|\mathbf{C}\|_\star \quad \text{s.t.} \quad D_\nu^* D_\nu = I_{n_\nu},\ \nu = 1, 2. \tag{3.3}$$

Here, as before $\|\cdot\|_\star$ denotes either $\|\cdot\|_0$ or $\|\cdot\|_1$. Observe that, compared to (2.1), the dictionary matrix $(D_2 \otimes D_1) \in \mathbb{C}^{m_1 m_2 \times n_1 n_2}$ has now a Kronecker structure that we did not impose before. Thus, learning the dictionary $(D_2 \otimes D_1)$ requires to determine only $n_1 m_1 + n_2 m_2$ instead of $n_1 n_2 m_1 m_2$ components. Surely, we need to exploit the freedom of choosing rectangular dictionary matrices D_1, D_2 in order to capture the important structures of the images in a sparse manner.

In order to solve the minimization problem (3.2) resp. (3.3), we adopt the alternating minimization scheme proposed in the last section. Now, each iteration step consists of three independent steps:

Step 1. First, for fixed dictionary matrices D_1, D_2, we minimize only with respect to $\mathbf{C}$ by applying either a hard or a soft threshold analogously as in Step 1 for the model (2.1).

Step 2. We fix $\mathbf{C}$ and D_2, and minimize (3.2) resp. (3.3) with respect to D_1. For this purpose, we rewrite (3.2) as

$$\min_{D_1 \in \mathbb{C}^{m_1 \times n_1}} \sum_{k=1}^{K} \|D_1 Y_k D_2^T - C_k\|_F^2 \qquad s.t. \quad D_1^* D_1 = I_{n_1}. \tag{3.4}$$

As in (2.5), this problem is equivalent to

$$\max_{D_1 \in \mathbb{C}^{m_1 \times n_1}} \operatorname{Re}\left(\operatorname{tr}\left(\sum_{k=1}^{K} D_1 (Y_k D_2^T) C_k^* \right) \right) \qquad s.t. \quad D_1^* D_1 = I_{n_1}.$$

We take the singular value decomposition

$$\sum_{k=1}^{K} Y_k D_2^T C_k^* = U_1 \Lambda_1 V_1^*$$

and obtain unitary matrices $U_1 \in \mathbb{C}^{n_1 \times n_1}$, $V_1 \in \mathbb{C}^{m_1 \times m_1}$ and a diagonal matrix of singular values $\Lambda_1 = \Big(\mathrm{diag}(\lambda_1^{(1)}, \ldots, \lambda_{n_1}^{(1)}), \mathbf{0}\Big) \in RR^{n_1 \times m_1}$. Now, similarly as shown in Theorem 2.2, the optimal dictionary matrix is obtained by $D_{1,opt} = V_{1,n_1} U_1^*$, where V_{1,n_1} denotes the restriction of V_1 to its first n_1 columns.

Since $\sum_{k=1}^{K} Y_k D_2^T C_k^*$ is a matrix of size $n_1 \times m_1$, we only need to apply the singular value decomposition of this size here to obtain an update for D_1.

Step 3. Analogously, in the third step we fix $\mathbf{C}$ and D_1 and minimize (3.2) resp. (3.3) with respect to D_2. Here, we observe that

$$\min_{D_2 \in \mathbb{C}^{m_2 \times n_2}} \sum_{k=1}^{K} \|D_1 Y_k D_2^T - C_k\|_F^2 \qquad s.t. \quad D_2^* D_2 = I_{n_2}$$

is equivalent to

$$\min_{D_2 \in \mathbb{C}^{m_2 \times n_2}} \sum_{k=1}^{K} \|D_2 Y_k^T D_1^T - C_k^T\|_F^2 \qquad s.t. \quad D_2^* D_2 = I_{n_2}.$$

The SVD

$$\sum_{k=1}^{K} Y_k^T D_1^T \overline{C}_k = U_2 \Lambda_2 V_2^*$$

with unitary matrices $U_2 \in \mathbb{C}^{n_2 \times n_2}$, $V_2 \in \mathbb{C}^{m_2 \times m_2}$, and $\Lambda_2 = \Big(\mathrm{diag}(\lambda_1^{(2)}, \ldots, \lambda_{n_2}^{(2)}), \mathbf{0}\Big) \in R^{n_2 \times m_2}$ yields the update

$$D_2 = V_{2,n_2} U_2^*,$$

where V_{2,n_2} again denotes the restriction of V_2 to its first n_2 columns.

We outline the pseudo code for learning the tight frame with Kronecker structure (KronTF) in the following Algorithm 1.

Algorithm 1 : KronTF Algorithm

Input: Training set of data $Y_1, Y_2 \cdots Y_K \in \mathbb{C}^{n_1 \times n_2}$, number of iterations T

Output: D_1, D_2

1: Initialize the dictionary matrices $D_1 \in \mathbb{C}^{m_1 \times n_1}$ and $D_2 \in \mathbb{C}^{m_2 \times n_2}$ with $D_1^* D_1 = I_{n_1}$, $D_2^* D_2 = I_{n_2}$.

2: for $k = 1, 2, \ldots, T$

3: Use the hard/soft thresholding to update the coefficient matrix $\mathbf{C} = (c_1 \ldots c_K)$ as given in Step 1.

4: for $n = 1$ to 2 do

5: Use the SVD method to update the dictionary matrices D_n as given in Steps 2 and 3.

6: end for

7: end for

This approach to construct a data-driven tight frame can now easily be extended to third order tensors. Using tensor notion (for 2-tensors) the product in (3.1) reads

$$C = Y \times_1 D_1 \times_2 D_2.$$

Generalizing the concept above to 3-tensors, we want to find dictionary matrices $D_\nu \in \mathbb{C}^{m_\nu \times n_\nu}$, $\nu = 1,2,3$, with $m_\nu \geq n_\nu$ and $D_\nu^* D_\nu = I_{n_\nu}$, $\nu = 1,2,3$, such that for a given sequence of tensors $\mathcal{Y}_k \in \mathbb{C}^{n_1 \times n_2 \times n_3}$, $k = 1, \ldots, K$, the core tensors

$$\mathcal{S}_k = \mathcal{Y}_k \times_1 D_1 \times_2 D_2 \times_3 D_3 \in \mathbb{C}^{m_1 \times m_2 \times m_3}$$

are simultaneously sparse. This is done by solving the minimization problem

$$\min_{D_1, D_2, D_3, \mathcal{S}_1, \ldots, \mathcal{S}_K} \sum_{k=1}^{K} \left(\|\mathcal{Y}_k \times_1 D_1 \times_2 D_2 \times_3 D_3 - \mathcal{S}_k\|_F^2 + \lambda \|\mathcal{S}_k\|_\star \right)$$
$$\text{s.t.} \quad D_\nu^* D_\nu = I_{n_\nu}, \quad \nu = 1,2,3. \tag{3.5}$$

Here, the Frobenius norm of $\mathcal{X} \in \mathbb{C}^{n_1 \times n_2 \times n_3}$ is defined by $\|\mathcal{X}\|_F := \left(\sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \sum_{k=3}^{n_3} |x_{i,j,k}|^2 \right)^{1/2}$, and $\|\mathcal{S}_k\|_\star$ denotes in the case $\star = 0$ the number of nonzero entries of $\mathcal{S}_k$, and for $\star = 1$ the sum of the moduli of all entries of $\mathcal{S}_k$.

The minimization problem (3.5) can be solved by four steps at each iteration level. In step 1, for fixed D_1, D_2, D_3, one minimizes $\mathcal{S}_k$, $k = 1, \ldots, K$, by applying a componentwise threshold procedure as before. In step 2, for fixed D_2, D_3 and $\mathcal{S}_k$, $k = 1, \ldots, K$, we use the unfolding

$$(\mathcal{S}_k)_{(1)} = D_1 (\mathcal{Y}_k)_{(1)} (D_3 \otimes D_2)^T$$

and have to solve

$$\min_{D_1 \in \mathbb{C}^{m_1 \times n_1}} \sum_{k=1}^{K} \|(\mathcal{S}_k)_{(1)} - D_1 (\mathcal{Y}_k)_{(1)} (D_3 \otimes D_2)^T\|_F^2 \quad \text{s.t.} \quad D_1^* D_1 = I_{n_1}.$$

This problem has exactly the same structure as (3.4) and is solved by choosing $D_1 = V_{1,n_1} U_1^*$, where $U_1 \Lambda_1 V_1^*$ is the singular value decomposition of the $(n_1 \times m_1)$-matrix

$$\sum_{k=1}^{K} (\mathcal{Y}_k)_{(1)} (D_3 \otimes D_2)^T (\mathcal{S}_k)_{(1)}^*$$

Analogously, we find in step 3 and step 4 the updates $D_2 = V_{2,n_2} U_2^*$ and $D_3 = V_{3,n_3} U_3^*$ from the singular decompositions

$$\sum_{k=1}^{K} (\mathcal{Y}_k)_{(2)} (D_1 \otimes D_3)^T (\mathcal{S}_k)_{(2)}^* = U_2 \Lambda_2 V_2^*$$

resp.

$$\sum_{k=1}^{K} (\mathcal{Y}_k)_{(3)} (D_2 \otimes D_1)^T (\mathcal{S}_k)_{(3)}^* = U_3 \Lambda_3 V_3^*.$$

Remarks 3.1 *1. This dictionary learning approach requires at each iteration step three SVDs but for matrices of moderate size $n_\nu \times m_\nu$, $\nu = 1, 2, 3$.*

2. One may also connect the two approaches considered in Section 2 and in Section 3 by e.g. enforcing a tensor product structure of the dictionary only in the third direction while employing a more general dictionary for the first and second direction. In this case we can use the unfolding

$$(\mathcal{S}_k)_{(3)} = D_3 \cdot (\mathcal{Y}_k)_{(3)} \cdot \tilde{D}^T,$$

where the dictionary $(D_2 \otimes D_1)$ is replaced by the matrix $\tilde{D} \in \mathbb{C}^{m_1 m_2 \times n_1 n_2}$ that not necessarily has the Kronecker product structure. The dictionary learning procedure is then applied as for two-dimensional tensors.

4. Directional data-driven tight frames with Kronecker structure

While the data-driven Kronecker tight frame considered in Section 3 is of simple structure and therefore dictionary learning is faster to implement, the Kronecker structure is limited for learning directional features, as usual for tensor product approaches. Therefore, we want to propose now a frame construction that contains both, a Kronecker structure for a (data-driven) basis or a frame, and a data-driven directional structure.

It is well-known that tensor-product bases or frames are especially well suited for representing vertical or horizontal structures in an image. Our idea is now to incorporate other favorite directions by mimicking rotation of the image. While an exact image rotation is not possible when we want to stay with the original grid, we apply the following simple procedure.

Let $V : \mathbb{C}^{n_1} \to \mathbb{C}^{n_1}$ be the cyclic shift operator, i.e., for some $x = (x_j)_{j=0}^{n_1-1}$ we have

$$V\,x := (x_{(j+1) \bmod n_1})_{j=0}^{n_1-1}.$$

For a given image $X = (x_0 \dots x_{n_2-1}) \in \mathbb{C}^{n_1 \times n_2}$ with columns $x_0, \dots, x_{n_2-1}$ in $\mathbb{C}^{n_1}$, we consider e.g. the new image

$$X_0 = (x_0, Vx_1, V^2x_2, \dots, V^{n_2-1}x_{n_2-1}),$$

where the j-th column is cyclically shifted by j steps. This procedure for example yields

$$X = \begin{pmatrix} 0 & 1 & 2 & 2 \\ 3 & 1 & 1 & 2 \\ 3 & 3 & 1 & 1 \\ 2 & 3 & 3 & 1 \end{pmatrix} \qquad X_0 = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 3 & 3 & 3 & 2 \\ 3 & 3 & 2 & 2 \\ 2 & 1 & 1 & 1 \end{pmatrix},$$

such that diagonal structures of X turn to horizontal structures of X_0. Vectorizing the

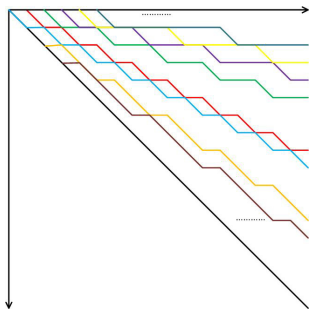


Figure 1. Illustration of different angles in the range $[0, \pi/4]$.

image X into $x = \mathrm{vec}(X)$, it follows that

$$x_0 = \mathrm{vec}X_0 = \mathrm{diag}(V^j)_{j=0}^{n_2-1} x = \begin{pmatrix} I & & & & \\ & V & & & \\ & & V^2 & & \\ & & & \ddots & \\ & & & & V^{n_2-1} \end{pmatrix} x,$$

where $\mathrm{diag}(V^j)_{j=0}^{n_2-1} \in \mathbb{C}^{n_1 n_2 \times n_1 n_2}$ denotes the block diagonal matrix with blocks V^j. Other rotations of X can be mimicked for example by multiplying $x = \mathrm{vec}X$ with

$$\mathrm{diag}\left(V^{\lfloor j\ell/n_1 \rfloor}\right)_{j=0}^{n_2-1}, \qquad \ell = 1, \ldots, n_1$$

for capturing the directions in $[-\pi/4, 0]$ and by

$$\mathrm{diag}\left(V^{-\lfloor j\ell/n_1 \rfloor}\right)_{j=0}^{n_2-1}, \qquad \ell = -n_1 + 1, -n_1 + 2, \ldots, 0$$

for capturing directions in $[0, \pi/4]$. This range is sufficient in order to bring essential edges into horizontal or vertical form. If a priori information about favorite directions (or other structures) in images is available, we may suitably adopt the method to transfer this structure into a linear vertical or horizontal structure. Possible column shifts in the data matrices mimicking directions are illustrated in Figure 1.

Using this idea to incorporate directionality, we generalize the model (3.2) resp. (3.3) for dictionary learning as follows. Instead of only considering the Kronecker tight frame $D_2 \otimes D_1$ we employ the dictionary

$$\begin{pmatrix} (D_2 \otimes D_1)(\mathrm{diag}\left(V^{\lfloor j\alpha_1 \rfloor}\right)_{j=0}^{n_2-1}) \\ \vdots \\ (D_2 \otimes D_1)(\mathrm{diag}\left(V^{\lfloor j\alpha_R \rfloor}\right)_{j=0}^{n_2-1}) \end{pmatrix} \in \mathbb{C}^{Rm_1 m_2 \times n_1 n_2}$$

with R constants $\alpha_1, \ldots, \alpha_R \in \{-1 + \frac{1}{n_1}, -1 + \frac{2}{n_1}, \ldots \frac{n_1-1)}{n_1}, 1\}$ capturing R favorite directions. Then the model reads

$$\min_{\substack{D_1,D_2,\mathbf{C}_R,\\ \alpha_1\dots,\alpha_R}} \sum_{r=1}^{R} \left(\|(D_2 \otimes D_1)(\mathrm{diag} V^{\lfloor j\alpha_r \rfloor})_{j=0}^{n_2-1}\mathbf{Y} - \mathbf{C}_r\|_F^2 + \lambda \|\mathbf{C}_r\|_\star \right)$$
$$\text{s.t.} \quad D_\nu^* D_\nu = I_{n_\nu},\ \nu = 1,2, \tag{4.1}$$

where $\mathbf{C}_r \in \mathbb{C}^{m_1 m_2 R \times K}$ contains the blocks of transform coefficients for each direction, and where $\|.\|_\star$ denotes either the 1-norm or the 0-subnorm as before. Compared to (3.3), the dictionary matrix is now composed of a Kronecker matrix $(D_2 \otimes D_1) \in \mathbb{C}^{m_1 m_2 \times n_1 n_2}$ and block diagonal matrices $(\mathrm{diag} V^{\lfloor j\alpha_r \rfloor})_{j=0}^{n_2-1}$, $r = 1,\dots,R$ capturing different directions of the image. Particularly, learning this dictionary only requires to determine $n_1 m_1 + n_2 m_2$ components of D_1, D_2 and R components α_r to fix the directions.

The minimization problem in (4.1) can again be solved by alternating minimization, where we determine the favorite directions already in a preprocessing step.

Preprocessing step. For fixed dictionary matrices D_1 and D_2 we solve the problem

$$\min_{\mathbf{C}_{\alpha_r}} \|(D_2 \otimes D_1)(\mathrm{diag} V^{\lfloor j\alpha_r \rfloor})_{j=0}^{n_2-1}\mathbf{Y} - \mathbf{C}_{\alpha_r}\|_F^2 + \lambda \|\mathbf{C}_{\alpha_r}\|_\star$$

for each direction α_r from a predetermined set of possible directions by applying either a hard or a soft threshold operator to the transformed sequence of images $\mathbf{Y} = (y_1 \dots y_K)$. We emphasize, that computing $(\mathrm{diag}\, V^{\lfloor j\alpha \rfloor})_{j=0}^{n_2-1}\mathbf{Y}$ for some fixed $\alpha \in (-1,1]$ is very cheap, it just means a cyclic shifting of columns in the matrices Y_k, such that $(D_2 \otimes D_2)(\mathrm{diag} V^{\lfloor j\alpha \rfloor})_{j=0}^{n_2-1}\mathbf{Y}$ corresponds to applying the two-dimensional dictionary transform to the images that are obtained from Y_k, $k = 1,\dots,K$, by taking cyclic shifts.

We a priori fix the number R of considered directions (in practice often just $R = 1$ or $R = 2$) and find the R favorite directions, e.g. by comparing the energies of $\mathbf{C}_{\alpha_r}$ for fixed thresholds or by comparing the PSNR values of the reconstructions of $\mathbf{Y}$ after thresholding.

Once the directions $\alpha_1,\dots,\alpha_R$ are fixed, we start the iteration process as before.

Step 1. At each iteration level, we first minimize $\mathbf{C}_{\alpha_r}$, $r = 1,\dots,R$, while the complete dictionary (directions and D_1, D_2) is fixed. This is done by applying the thresholding procedure. This step is not necessary at the first level since we can use here $\mathbf{C}_{\alpha_r}$ obtained in the preprocessing step.

Step 2. For fixed directions $\alpha_1,\dots,\alpha_R$, corresponding $\mathbf{C}_{\alpha_r}$, $r = 1,\dots,R$, and fixed D_2 we consider the minimization problem for D_1. We recall that $\mathbf{C}_{\alpha_r}$ consists of K columns, where the k-th column is the vector of transform coefficients of Y_k. We reshape

$$\mathbf{C}_{\alpha_r} \in \mathbb{C}^{m_1 m_2 \times K} \quad \Rightarrow \quad (C_{\alpha_r,1}, \dots C_{\alpha_r,K}) \in \mathbb{C}^{m_1 \times m_2 K}$$

and

$$\mathrm{diag}\left(V^{\lfloor j\alpha_r \rfloor}\right)_{j=0}^{n_2-1}\mathbf{Y} \in \mathbb{C}^{n_1 n_2 \times K} \quad \Rightarrow \quad (Y_{\alpha_r,1}, \dots, Y_{\alpha_r,K}) \in \mathbb{C}^{n_1 \times n_2 K},$$

where the k-th column of $\mathbf{C}_{\alpha_r}$ of length $m_1 m_2$ is reshaped back to an $(m_1 \times m_2)$-matrix $C_{\alpha_r,k}$ by inverting the vec operator, and analogously, the k-th column of $\operatorname{diag}\left(V^{\lfloor j\alpha_r \rfloor}\right)_{j=0}^{n_2-1} y_k \in \mathbb{C}^{n_1 n_2}$ is reshaped to a matrix $Y_{\alpha_r,k} \in \mathbb{C}^{n_1 \times n_2}$. Now we have to solve

$$\min_{D_1 \in \mathbb{C}^{m_1 \times n_1}} \sum_{r=1}^{R} \sum_{k=1}^{K} \|D_1(Y_{\alpha_r,k} D_2^T - C_{\alpha_r,k}\|_F^2 \quad s.t. \quad D_1^* D_1 = I_{n_1}$$

with similar structure as in (3.4). As before, we apply the singular value decomposition

$$\sum_{r=1}^{R} \sum_{k=1}^{K} Y_{\ell_r,k} D_2^T C_{\ell_r,k}^* = U_1 \Lambda_1 V_1^*$$

with unitary matrices $U_1 \in \mathbb{C}^{n_1 \times n_1}$, $V_1 \in \mathbb{C}^{m_1 \times m_1}$ and a diagonal matrix of singular values $\Lambda_1 = \left(\operatorname{diag}(\lambda_1^{(1)}, \ldots, \lambda_{n_1}^{(1)}), \mathbf{0}\right) \in RR^{n_1 \times m_1}$. Now, as shown in Theorem 2.2, the optimal dictionary matrix is obtained by $D_{1,opt} = V_{1,n_1} U_1^*$, where V_{1,n_1} denotes the restriction of V_1 to its first n_1 columns.

Step 3. For fixed directions $\alpha_1, \ldots, \alpha_R$, corresponding $\mathbf{C}_{\alpha_r}$, $r = 1, \ldots, R$, and fixed D_1 we consider the minimization problem for D_2. With the same notations as above, we can write

$$\min_{D_2 \in \mathbb{C}^{m_2 \times n_2}} \sum_{r=1}^{R} \sum_{k=1}^{K} \|D_2 Y_{\alpha_r,k}^T D_1^T - C_{\alpha_r,k}^T\|_F^2 \quad s.t. \quad D_2^* D_2 = I_{n_2}$$

and obtain the optimal solution $D_{2,opt} = V_{2,n_2} U_2^*$ from the singular value decomposition

$$\sum_{r=1}^{R} \sum_{k=1}^{K} Y_{\alpha_r,k}^T D_1^T \overline{C}_{\alpha_r,k} = U_2 \Lambda_2 V_2^*,$$

where V_{2,n_2} denotes the restriction of V_2 to its first n_2 columns. We outline the pseudo code for learning the tight frame with Kronecker structure and one optimal direction in Algorithm 2.

5. Application to data reconstruction and denoising

We want to apply the new data-driven dictionary constructions to 2D and 3D data reconstruction (resp. data interpolation) and to data denoising. Let X denote the complete correct data, and let Y be the observed data. We assume that these data are connected by the following relation

$$Y = A \circ X + \gamma \xi. \tag{5.2}$$

Here $A \circ X$ denotes the pointwise product (Hadamard product) of the two matrices A and X, ξ denotes an array of normalized Gaussian noise with expectation 0, and $\gamma \geq 0$ determines the noise level. The matrix A contains only the entries 1 or 0 and is called *trace sampling operator*. If $\gamma = 0$ then the above relation models a seismic

Algorithm 2 : KronTFD Algorithm

Input: Training set of data $Y_1, Y_2, \cdots, Y_K \in \mathbb{C}^{n_1 \times n_2}$, number of iterations T
Output: D_1, D_2, optimal *angle*
1: Initialize the dictionary matrices $D_1 \in \mathbb{C}^{m_1 \times n_1}$ and $D_2 \in \mathbb{C}^{m_2 \times n_2}$ with $D_1^* D_1 = I_{n_1}$, $D_2^* D_2 = I_{n_2}$
First: Find optimal angle direction of the training data.
2: for $k = 1, 2, \ldots, K$
3: for *angle* $= -45, -40, \ldots, 45$
4: Adjust the data Y_k by cyclic shifting of columns by the angle.
5: Apply the dictionary transform to Y_k, $k = 1, \ldots, K$, and use the hard/soft thresholding to update the coefficient matrix $(C_1 \ldots C_K)$.
6: Apply the inverse dictionary transform for data recovery and record the achieved SNR value.
7: end for
8: The largest SNR value yields the optimal *angle* direction of training data.
9: Adjust the data Y_k by cyclic shifting of columns by the optimal angle.
10: end for
Second: Learn the dictionary.
11: for $k = 1, 2, \ldots, T$
12: Use the hard/soft thresholding to update the coefficient matrix $(C_1 \ldots C_K)$.
13: for $n = 1$ to 2 do
14: Use the SVD method to update the dictionary matrices D_n.
15: end for
16: end for

interpolation problem, and the task is to reconstruct the missing data. If $A = I_1$, where I_1 is the matrix containing only ones, and $\gamma > 0$, it models a denoising problem. The two problems can be solved by a sparsity-promoting minimization method, see e.g. [8]. We assume that our desired data X can be sparsely represented by the dictionary that has been learned beforehand, either by

$$u = \mathrm{vec}(U) = (D_2 \otimes D_1)\mathrm{vec}(X), \quad \text{i.e.} \quad x = \mathrm{vec}(X) = (D_2^* \otimes D_1^*)u$$

for the data-driven tight frame in Section 3 or by

$$\begin{aligned} u \quad &= \mathrm{vec}(U) = (D_2 \otimes D_1)\mathrm{diag}(V^{\lfloor j\alpha \rfloor})_{j=0}^{n_2}\mathrm{vec}(X), \\ \text{i.e., } x &= \mathrm{vec}(X) = (\mathrm{diag}(V^{\lfloor j\alpha \rfloor})_{j=0}^{n_2})^T (D_2^* \otimes D_1^*)u \end{aligned}$$

for a suitable $\alpha \in (-1, 1]$, see Section 4. In the next section on numerical simulations, we consider some examples and show, how a suitable data-driven dictionary can be obtained from the observed incomplete or noisy data employing Algorithm 1 or Algorithm 2.

For the given data Y we now have to solve the minimization problem

$$u^* = \underset{u}{\operatorname{argmin}} \, ||\mathrm{vec}(Y) - \tilde{A}((\mathrm{diag}(V^{\lfloor j\alpha \rfloor})_{j=0}^{n_2})^T (D_2^* \otimes D_1^*)u)||_2^2 + \lambda ||u||_1, \quad (5.3)$$

where $\tilde{A}$ denotes the vectorization of the sampling operator A. Afterwards, the desired image X is obtained from u^* by the inverse transform as indicated above.

There exist many iterative algorithms to solve such a minimization problem, as e.g. the FISTA algorithm [4] or a first-order primal-dual algorithm, see [9, 11].

In Geophysics alternating projection algorithms as POCS (projection onto convex sets) are very popular. The approach in [2] for Fourier bases (instead of frames) can be interpreted as follows. We may try to formulate the interpolation problem as a feasibility problem. We look for a solution X of (5.2) that on the one hand satisfies the interpolation condition $A \circ X = Y$, i.e., is contained in the set of all data

$$M := \{Z : \, A \circ Z = Y\}$$

possessing the observed data Y. This constraint can be enforced by applying the projection operator onto M,

$$P_M X = (I_1 - A) \circ X + Y$$

that leaves the unobserved data unchanged and projects the observed traces to Y.

On the other hand, we want to ensure that the solution X has a sparse representation in the constructed data-driven frame. The sparsity constraint is enforced by applying a (soft) thresholding to the transformed data, i.e. we compute

$$P_{D_\lambda} X := \mathcal{D}^{-1} S_\lambda \mathcal{D} X,$$

where $\mathcal{D}$ denotes the dictionary operator that maps X to the dictionary coefficients, and S_λ is the soft threshold operator as in (2.3). In our case, we had e.g.

$$U = \mathcal{D}X = D_1 X D_2^T, \qquad \mathcal{D}^{-1}U = D_1^* U \overline{D}_2 = X$$

in Chapter 3, and one can easily also incorporate the directional sensitivity as in Chapter 4. We observe however, that P_{D_λ} is not longer a projector and therefore this approach already generalizes a usual alternating projection method.

The complete iteration scheme can be obtained by alternating application of P_M and $P_{D_{\lambda_k}}$,

$$X_{k+1} = P_M(P_{D_{\lambda_k}} X_k) = (I_1 - A) \circ (P_{D_{\lambda_k}} X_k) + Y, \quad (5.4)$$

where λ_k is the threshold value that can vary at each iteration step. We recall that all elements of the matrix I_1 are one. To show convergence of this scheme in the finite-dimensional setting one can transfer ideas from [26], where an iterative projection scheme had been applied to a phase retrieval problem incorporating sparsity in a shearlet frame.

To improve the convergence of this iteration scheme in numerical experiments, we adopt the following exponentially decreasing thresholding parameters, see [15],

$$\lambda_k = \lambda_{max} e^{b(k-1)}, \;\; k = 1, 2, \ldots, iter,$$

where $\lambda_1 = \lambda_{max}$ represents the maximum parameter, $\lambda_{iter} = \lambda_{min}$ the minimum parameter, and b is chosen as $b = (\frac{-1}{iter-1}) \ln(\frac{\lambda_{max}}{\lambda_{min}})$, where *iter* is the fixed number of iterations in the scheme.

For data denoising, we only apply an iterative thresholding procedure given by

$$X_{k+1} = D_1(S_\lambda(D_1^T X_k D_2))D_2^T, \tag{5.5}$$

where λ is the threshold parameter related to the noise level γ. Our numerical experiments show that the λ choose about 3γ is a good value for denoising.

6. Numerical Simulations

In this section we want to apply Algorithm 1 and Algorithm 2 for data-driven dictionary learning to interpolation and denoising of seismic images. In a first illustration, we compare the dictionaries learned by KronTF in Algorithm 1 with the dictionary obtained by the DDTF method in [8] and the fixed Fourier dictionary used e.g. in the POCS algorithm, [2]. As initially known data, we use the seismic data of size (128×128) in Figure 5(b), where 50 % of the traces are missing.

We shortly explain the procedure to evaluate the dictionary in this example. In a first step, we employ pre-interpolation to the data, where each missing trace is recovered by the nearest adjacent given trace. If for a missing trace both, the left and the right neighbor trace are given, we take the left trace. Having filled the incomplete data we obtain the interpolation $P = (p_{jk})_{j,k=0}^{127} \in \mathbb{C}^{128\times128}$ in this way. Next, we collect 64×64 patches Y_k out of these data. For the special example, we use the overlapping patches

$$(p_{j+8\ell_1,k+8\ell_2})_{j,k=0}^{63}, \qquad \ell_1, \ell_2 = 0, \ldots, 7,$$

and obtain $K = 64$ patches $Y_k \in \mathbb{C}^{64\times64}$. Thus, $\mathbf{Y} = (y_1 \ldots y_{64}) \in \mathbb{C}^{64^2\times64}$, where $y_k = \mathrm{vec}(Y_k) \in \mathbb{C}^{4096}$. We employ now the Fourier basis, i.e., $D_1 \otimes D_2 = \frac{1}{64} F_{64} \otimes F_{64}$ with $F_{64} := (\omega_{64}^{jk})_{j,k=0}^{63}$ and $\omega_{64} := \exp(-\frac{2\pi \mathrm{i}}{64})$, as initial dictionary, which in this case is even a unitary transform. Then we apply Algorithm 1 in Section 3 using $T = 2$ iterations to update D_1 and D_2. The obtained two dictionary matrices D_1 and D_2 of size 64×64 are displayed in Figures 2(a) and 2(b).

While the obtained KronTF dictionary $D_1 \otimes D_2$ can be applied now to 64×64 images, a corresponding DDTF dictionary in [8] would need a 4096×4096 dictionary matrix to cope with vectorized 64×64 images, and the evaluation of such a dictionary matrix is not feasible in practice. Therefore, in [8], only 8×8 (or 16×16) training patches Y_k are considered. Figure 2(c) shows a dictionary obtained using training patches from P (the number of patches is 4096),

$$(p_{j+\ell_1,k+\ell_2})_{j,k=0}^{7}, \qquad \ell_1, \ell_2 = 0, \ldots, 63,$$

using the procedure of Algorithm 1 in [8] with 3 iterations and starting with an initial dictionary given by tensor linear spline framelet with filter size 8×8. For comparison, we

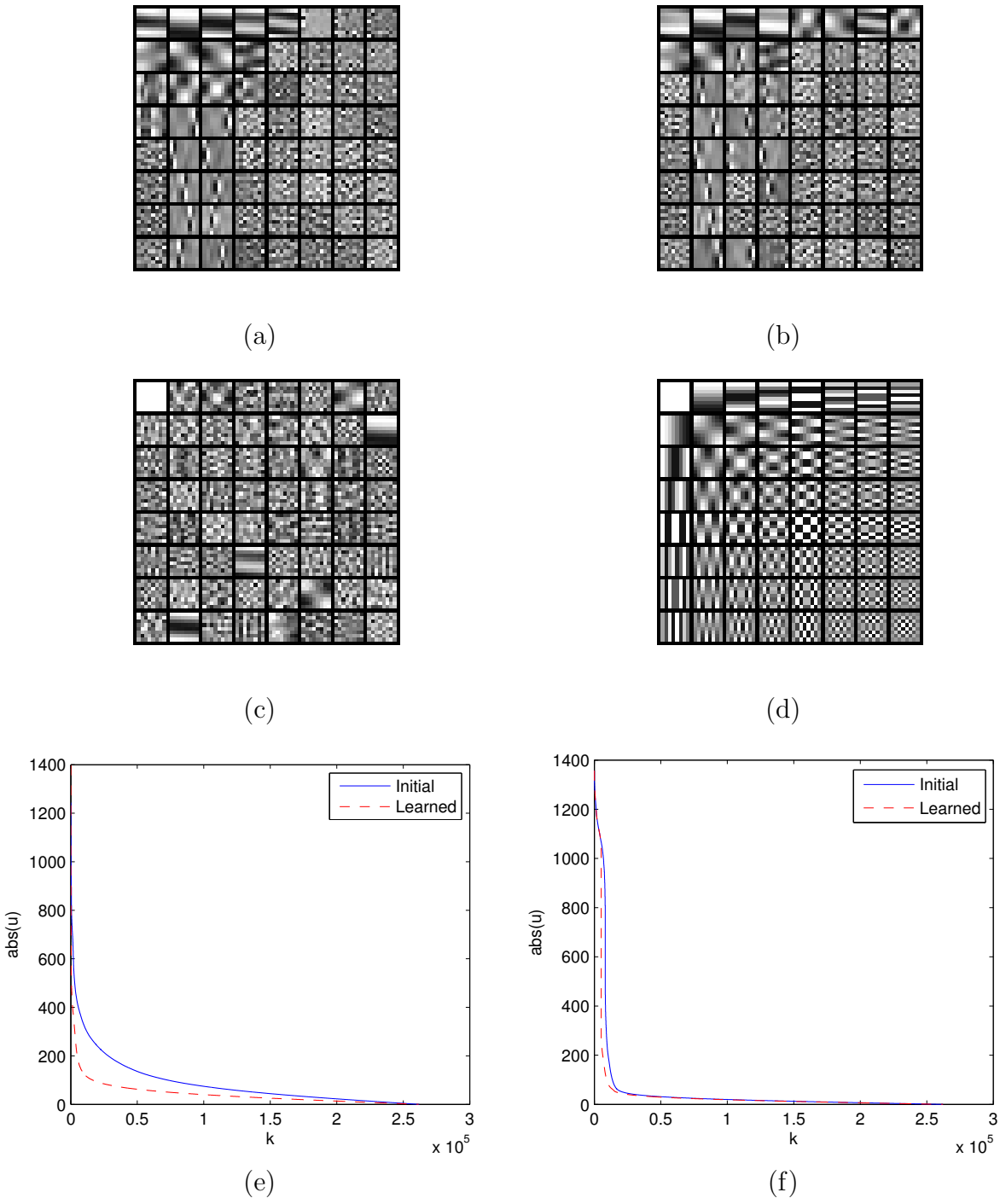
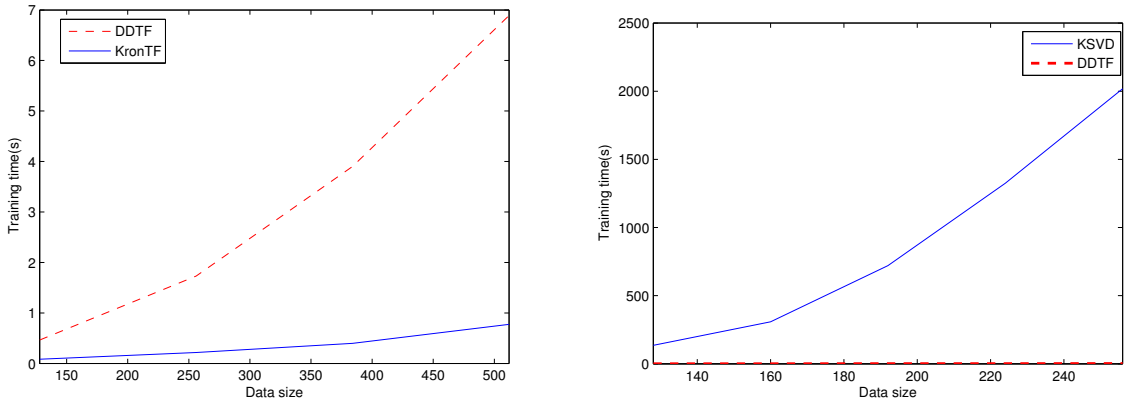


Figure 2. (a) Dictionary $D_1 \in \mathbb{C}^{64\times 64}$ learned via KronTF. (b) Dictionary $D_2 \in \mathbb{C}^{64\times 64}$ learned via KronTF. The learned dictionaries are based on training data from Figure 5(b). (c) Learned dictionary via DDTF of size 64×64. (d) Fourier dictionary $(F_8 \otimes F_8)$. (e)-(f) Absolute values of frame coefficients using KronTF (left) and DDTF (right). Solid lines denote absolute values of frame coefficients using initial dictionaries.



(a) (b)

Figure 3. Comparison of time costs for dictionary learning.

also show the fixed dictionary obtained from $\frac{1}{8}(F_8 \otimes F_8) \in \mathbb{C}^{64\times 64}$ (with $F_8 = (\omega_8^{jk})_{j,k=0}^{7}$) in Figure 2(d). Such a fixed Fourier basis is used in the POCS algorithm [2].

Figures 2(e) and (f) show, how well the data in the used training patches can be sparsified by the learned dictionaries compared to the initial dictionaries. Here we have computed the absolute values of coefficients of $u = (D_2 \otimes D_1)\mathrm{vec}(X)$ with the found matrices D_1 and D_2 for KronTF in comparison to $u = \frac{1}{64}(F_{64} \otimes F_{64})\mathrm{vec}(X)$ in Figure 2(e). To apply the DDTF that works only for 8×8 blocks, we split X into 64 blocks of size 8×8 and apply the obtained DDTF dictionary in Figure 2(b) separately to each of these blocks. The absolute coefficients are illustrated in Figure 2(f), compared to the coefficients obtained using the initial frame, see [8].

We observe that DDTF works here slightly better than KronTF with regard to sparsification, but requires a much higher computational effort for dictionary learning. The comparison of time costs is illustrated in Figure 3. Here, we also show the comparison to K-SVD that is even more expensive since it incorporates many SVDs, see Remark 2.3. A Matlab implementation for the K-SVD methods is available from the authors of [1], see `http://www.cs.technion.ac.il/~elad/software/`.

We want to use the new data-driven tight frames of Kronecker type (KronTF) and the data-driven directional frames of Kronecker type (KronTFD) for interpolation and denoising of 2D and 3D seismic data, and compare the performance with the results using the POCS algorithm based on the Fourier transform [2], curvelet frames and data-driven tight frames (DDTF) method [24].

For the POCS method, where a two-dimensional Fourier transform is applied to seismic data blocks of size 64×64, we use overlapping patches to suppress the periodicity artifacts resulting from the Fourier transform. For the DDTF method, always a tensor linear spline framelet as proposed in [8] is applied as the initial dictionary, and the dictionary size is 8×8.

The quality of the reconstructed images is compared using the PSNR (peak signal-

to-noise ratio) value, given by

$$PSNR = 10\log(\frac{\max_X - \min_X}{\frac{1}{MN}\sum_{i,j}(X_{i,j} - \tilde{X}_{i,j})}), \tag{6.6}$$

where $X \in \mathbb{C}^{M\times N}$ denotes the original seismic data and $\tilde{X} \in \mathbb{C}^{M\times N}$ is the recovered seismic data.

In a first test we consider synthesis data of size 512×512, see Figure 4. Figure 4(a) shows the original real data and Figure 4(b) the sub-sampled data with 50% random traces missing.

We compare the reconstructions (interpolation) using the DDFT method in Figure 4(c), the KronTF method in Figure 4(d) and the KronTFD method in Figure 4(e). Here, the dictionaries are evaluated as described before using training patches from the pre-interpolated data in Figure 4(b), where this time, $10201 = 101^2$ patches of size 64×64 are used as training patches for KronTF and KronTFD. The dictionaries are realized by Algorithm 1 resp. Algorithm 2, where in the second case only one favorite direction is fixed from the set of 15 predefined angles. In both algorithms, we have taken only $T = 2$ iterations. In comparison, we show DDTF applied to 8×8 image blocks in Figure 4(c). For the given dictionaries, the image is reconstructed by solving (5.3) and applying the inverse transform. Figure 4(f) illustrates the result for one fixed trace.

In a next example, we consider real seismic data. In Figure 5 and Figure 6 we present the interpolation results using different reconstruction methods. Figures 5(c)-(f) show the interpolation results by the POCS method and the curvelet transform as well as the difference images.

In Figure 6, we show the corresponding interpolation results for DDTF, KronTF, and KronTFD (with one favorite direction) together with the error between the recovery and original data. Here, the dictionaries shown in Figure 2 have been applied.

For a further comparison of the recovery results, we display a single trace in Figure 7. In Table 1, we list the comparisons of reconstruction results obtained from incomplete data with different sampling ratios.

In a next experiment, we consider the denoising performance of the method. Here the seismic data have been corrupted by white noise with noise level 20, see Figure 8(b). In order to construct the dictionary matrices D_1 and D_2 of size 64×64 we use again 64 training patches of size 64×64 from the noisy image, similarly as explained before for the case of interpolation. Then Algorithm 1 (resp. Algorithm 2 with one favorite direction) is applied with $T = 2$ iterations to achieve the data-driven dictionaries D_1 and D_2 from the initial Fourier dictionary. For DDTF method, we also proceed as for the interpolation application using now the noisy image instead of the the pre-interpolated image P to extract the training patches.

Figures 8 and 9 show denoising results for the data in Figure 8(b). We compare the results of denoising by the POCS method [2], denoising by the curvelet transform, DDTF method, and our method based on the new frames KronTF and KronTFD, respectively. For POCS, DDTF, KronTF and KronTFD, which are in our construction

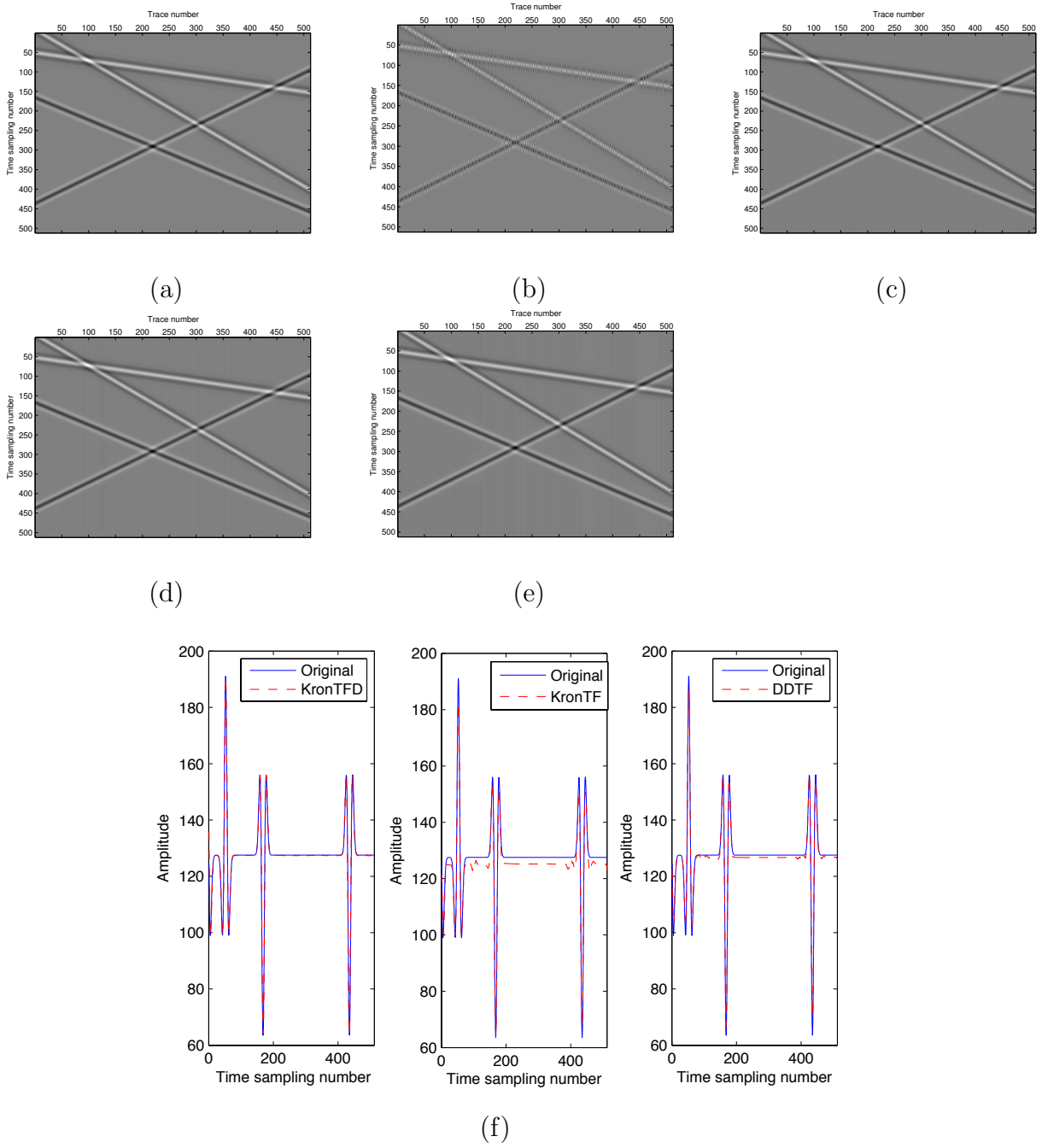


Figure 4. Interpolation results using DDTF and the new KronTF methods for synthesis data with irregular sampling ratio of 0.5. (a) Original seismic data. (b) Seismic data with 50% traces missing. (c) Interpolation by DDTF method. (d) Interpolation by KronTF method. (e) Interpolation by KronTFD method. (f) Single trace comparison of the reconstructions with the original data.

all uniform (non-redundant) transforms, we employ a cycle-spinning here as its is usual for wavelet denoising, i.e., we apply the denoising method (5.5) to shifts of the image blocks and compute the average. With our new data-driven frames, we can achieve better results than the other methods, because the dictionary learning methods contain

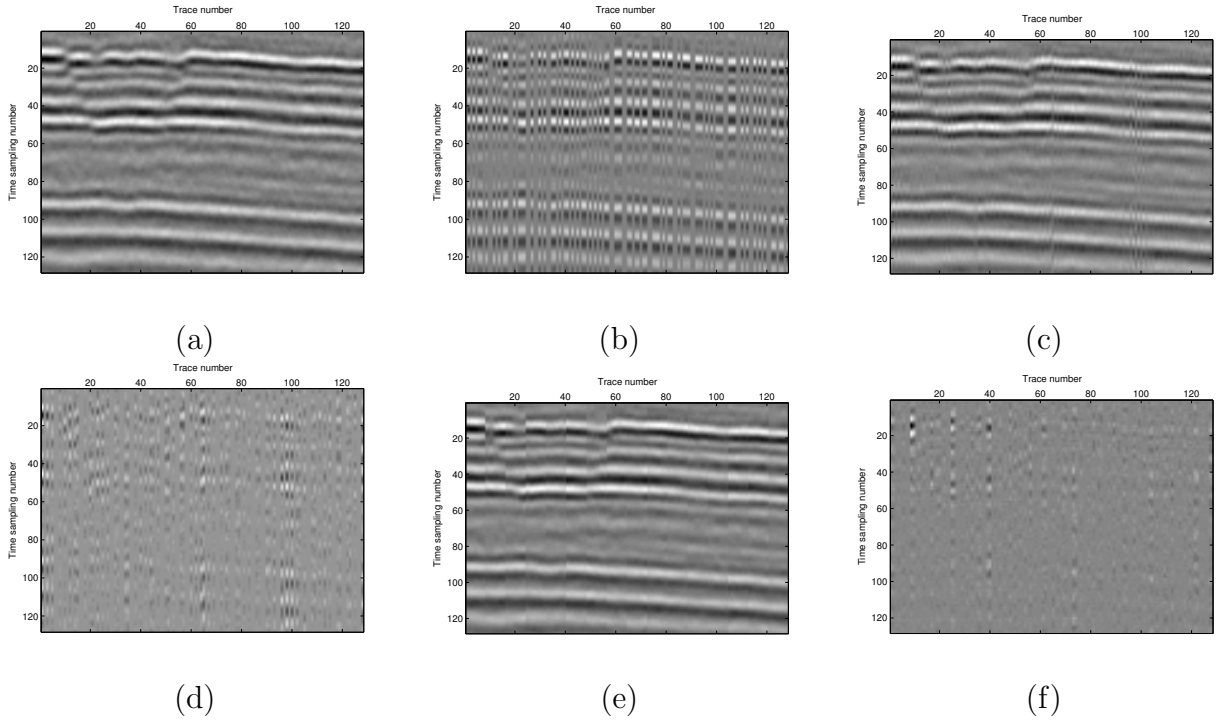


Figure 5. Interpolation results on real data with an irregular sampling ratio of 0.5. (a) Original seismic data. (b) Seismic data with 50% trace missing. (c) Interpolation using the POCS method. (d) Difference between (c) and (a). (e) Interpolation using the curvelet frame. (f) Difference between (e) and (a).

the information on the special seismic data. For better comparison, we also display the single trace comparisons in Figure 10. In Table 2, we list the comparisons of the achieved PSNR value with different noise levels. KronTF and KronTFD achieve competitive results both for interpolation and denoising.

Finally, we test the interpolation of 3D data with 50% randomly missing traces. In Figure 11, we applied the KronTF and the KronTFD tensor technique to the synthetic 3D seismic data of size $178 \times 178 \times 128$). Here, $8 \times 8 \times 8$ patches of the pre-interpolated observed (incomplete) data are used for training. The results of the DDTF method for 3D data are shown in Figures 11(c). In Figure 11(d)-(e) we present the results obtained by the KronTF and KronTFD method, respectively. The single trace comparisons are also shown in Figure 11(f). Figure 12 and Figure 13 show an interpolation comparison of the DDTF method and KronTF method for a real 3D marine data of size $251 \times 401 \times 50$, where we have applied the same methods as explained above.

7. Conclusion

This paper aims at exploiting sparse representation of seismic data for interpolation and denoising. We have proposed a new method to construct data-driven tensor-product tight frames. In order to enlarge the flexibility of the dictionaries, we have also proposed a simple method to incorporate favorite local directions that are learned

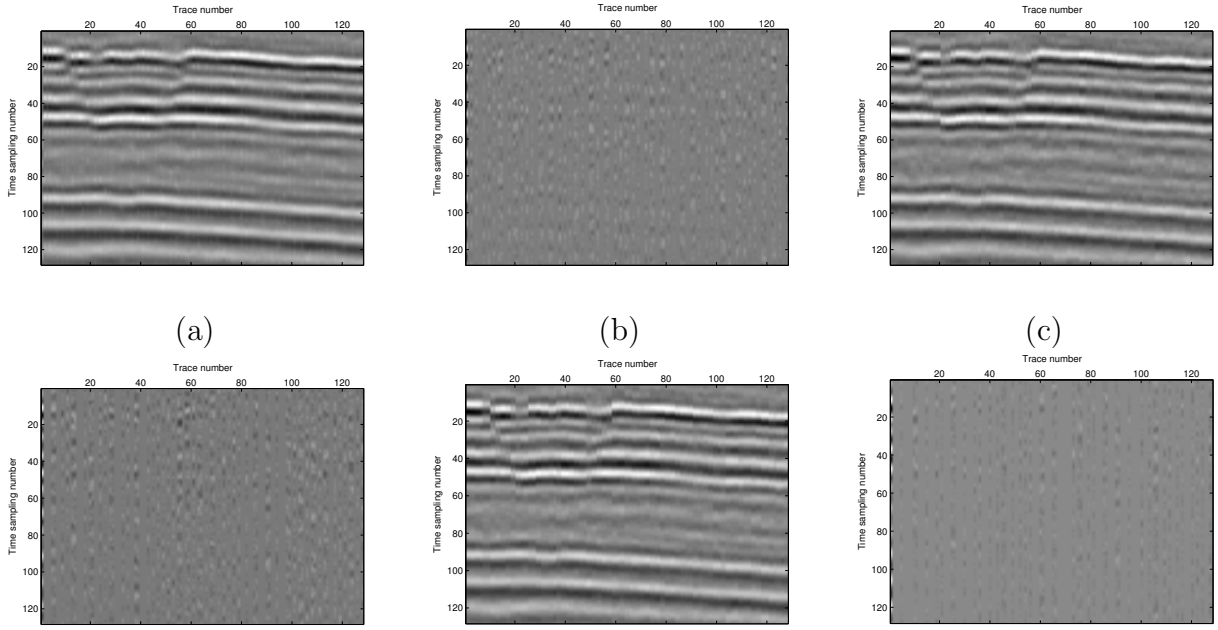


Figure 6. Interpolation results (continued) of Figure 5(b). (a) Interpolation using the DDTF. (b) Difference between (a) and Figure 5(a). (c) Interpolation using the KronTF. (d) Difference between (c) and Figure 5(a). (e) Interpolation using the KronTFD. (f) Difference between (e) and Figure 5 (a).

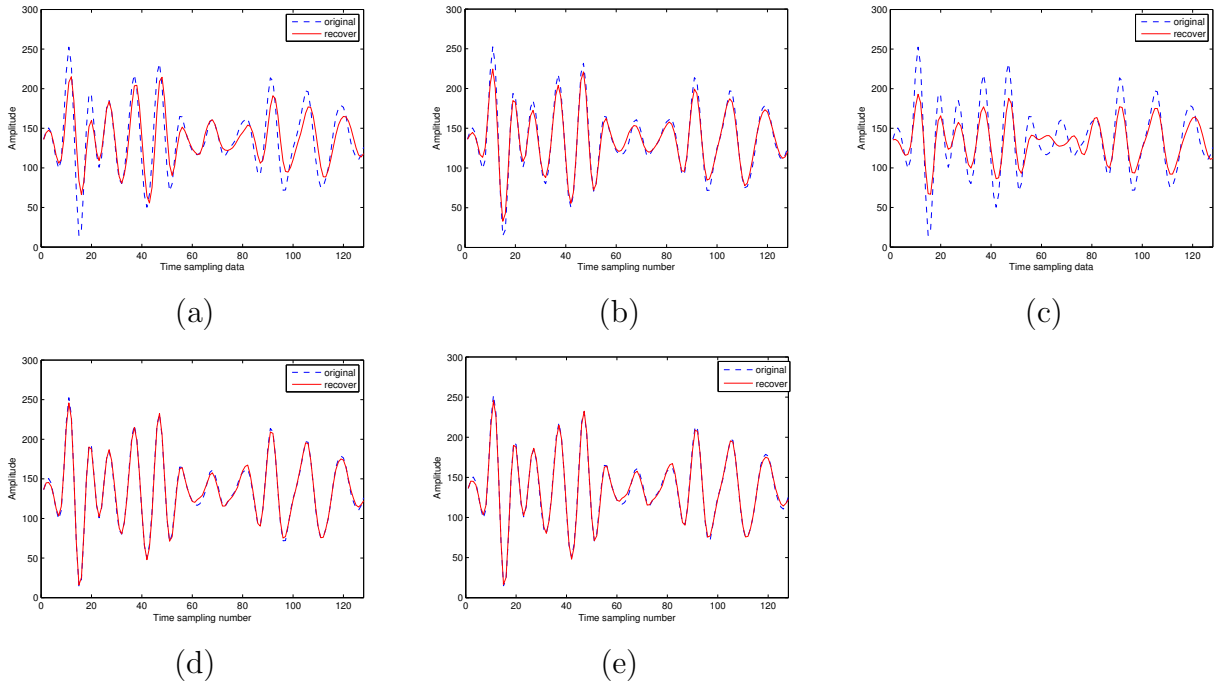


Figure 7. (a)-(f) Single trace comparison of the reconstructions in Figure 5 and 6 with the original Figure 5(a).

Table 1. PSNR comparison of five methods for different sampling ratio.

Sampling ratio	0.2	0.3	0.4	0.5	0.6	0.7	0.8
KronTFD	35.1327	37.3572	38.8862	41.2564	43.0243	44.0132	44.8996
KronTF	34.6198	36.8759	38.3167	40.6312	42.4675	43.8262	44.5342
DDTF	33.2942	35.1486	36.9543	39.0385	40.1064	42.3052	42.5668
Curvelet	31.1674	34.6478	35.6347	37.6377	39.4190	40.8930	41.5727
POCS	27.5122	31.8804	35.5560	38.2300	39.9878	41.0084	41.6816

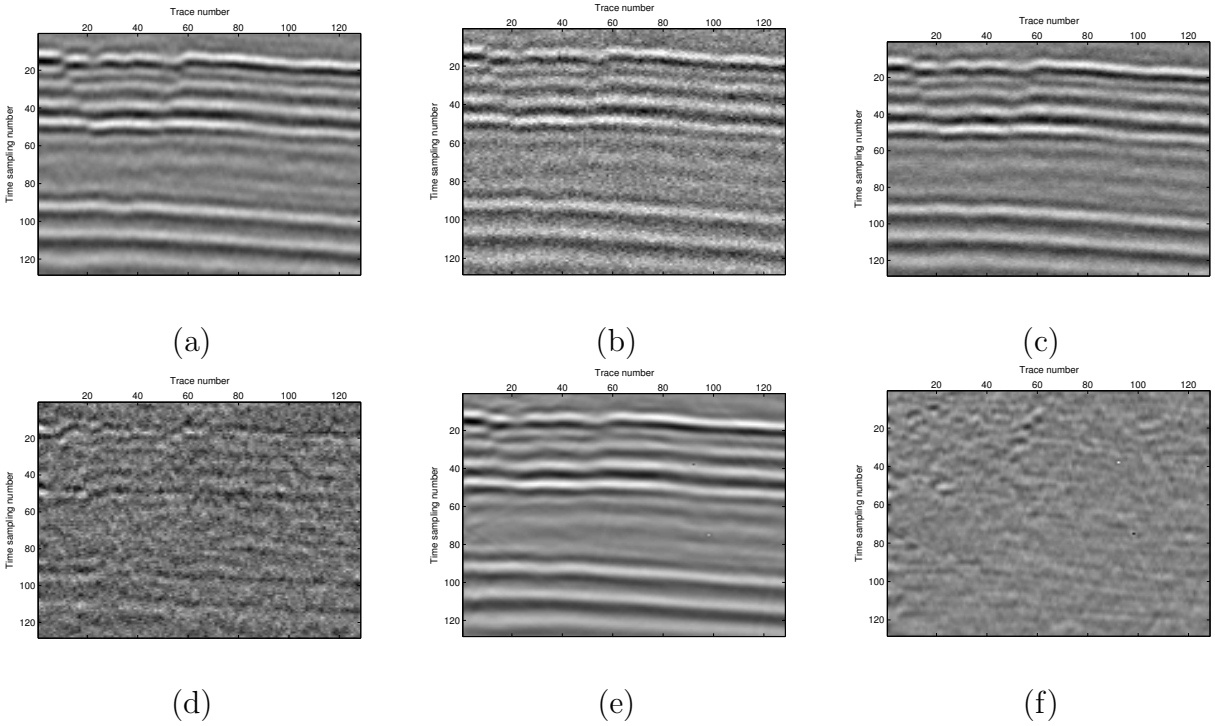


Figure 8. Denoising results of methods on real data with noise level of 20. (a) Original seismic data. (b) Noisy data. (c) Denoising result by POCS method. (d) Difference between (c) and (a). (e) Denoising result by curvelet method. (f) Difference between (e) and (a).

from the data. In the numerical experiments, we employed the new dictionaries to improve both seismic data interpolation and denoising. The main advantage of the proposed dictionary construction is its computational efficiency which is due to the imposed tensor-product structure.

However tensor-based methods possess limitations regarding to their fixed structure. The overall goal remains to connect certain predetermined dictionary structure with data driven components in order to construct dictionaries for sparse data representation that are feasible also in 3D or even higher dimensions. At the same time they need to be flexible enough to provide a suitable tool for data processing. This is of even higher importance, if we want to cope with geophysical data of up to five dimensions.

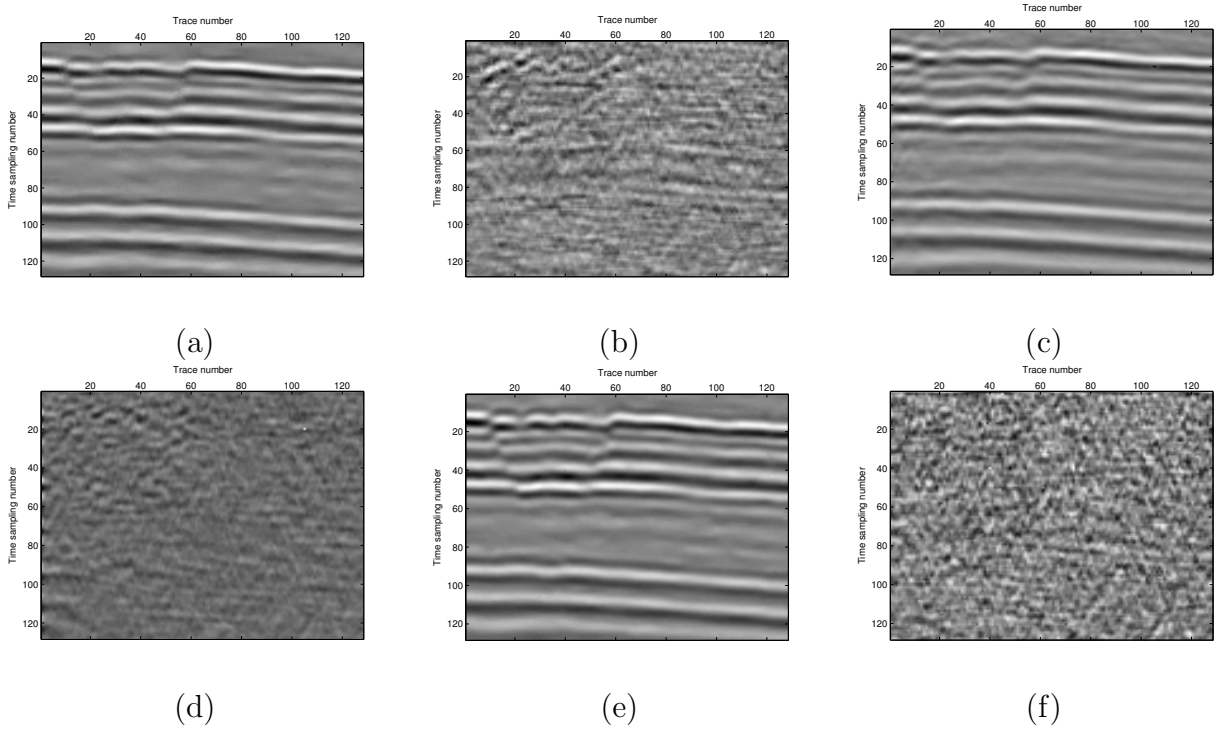


Figure 9. Denoising results for data in Figure 8(b).(a) Denoising by DDTF method. (b) Difference between (a) and 8(a). (c) Denoising by KronTF method. (d) Difference between (c) and 8(a). (e) Denoising by KronTFD method. (f) Difference between (e) and 8(a).

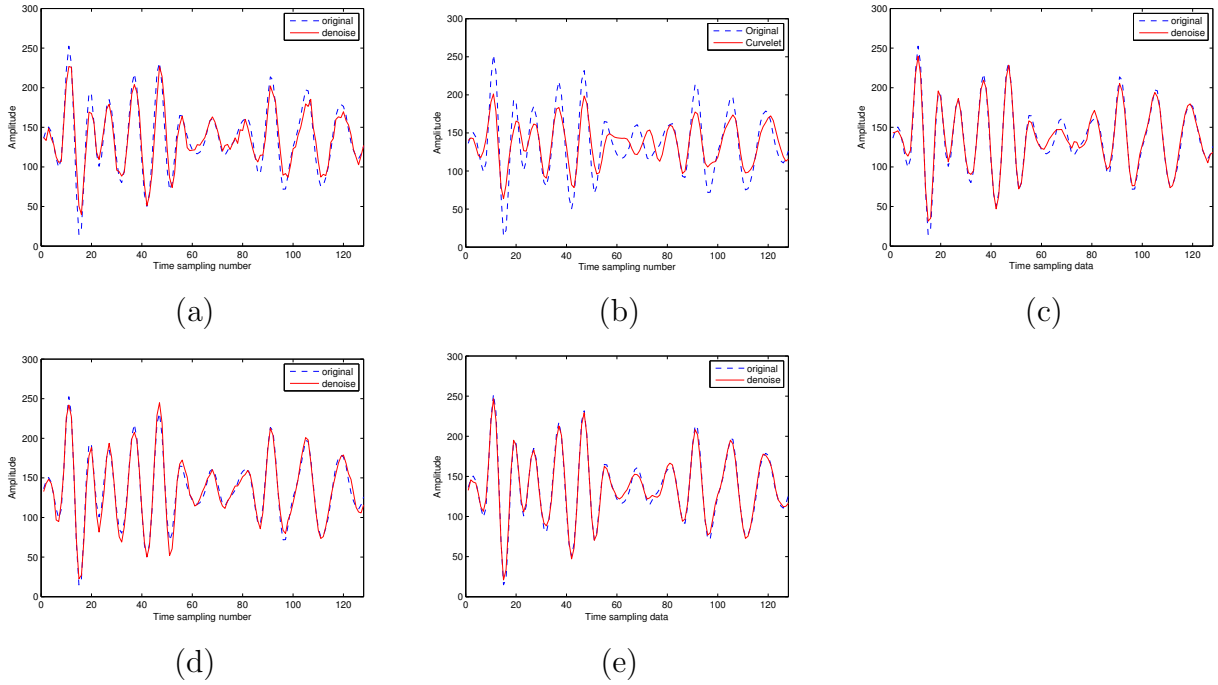


Figure 10. Single trace comparison for the recovery results in Figure 8 and Figure 9 and the original Figure 8 (a). Here we have (a) application of POCS method, (b) curvelet denoising, (c)-(e) application of the data-driven frames DDTF, KronTF and KronTFD.

Table 2. PSNR comparison of five methods for different noise levels.

Noise level	5	10	15	20	25	30	35
KronTFD	42.4425	39.2846	37.0246	36.1475	35.2537	34.5327	33.8631
KronTF	41.5282	38.5876	36.4967	35.7451	34.7496	33.9365	33.2156
DDTF	41.4918	38.4231	36.2578	35.0778	33.7533	32.5935	31.5993
Corvelet	40.7478	38.0749	35.8603	34.4654	33.2531	32.0124	30.6583
POCS	40.6547	36.7751	34.1781	32.0267	30.8375	28.6220	27.9847

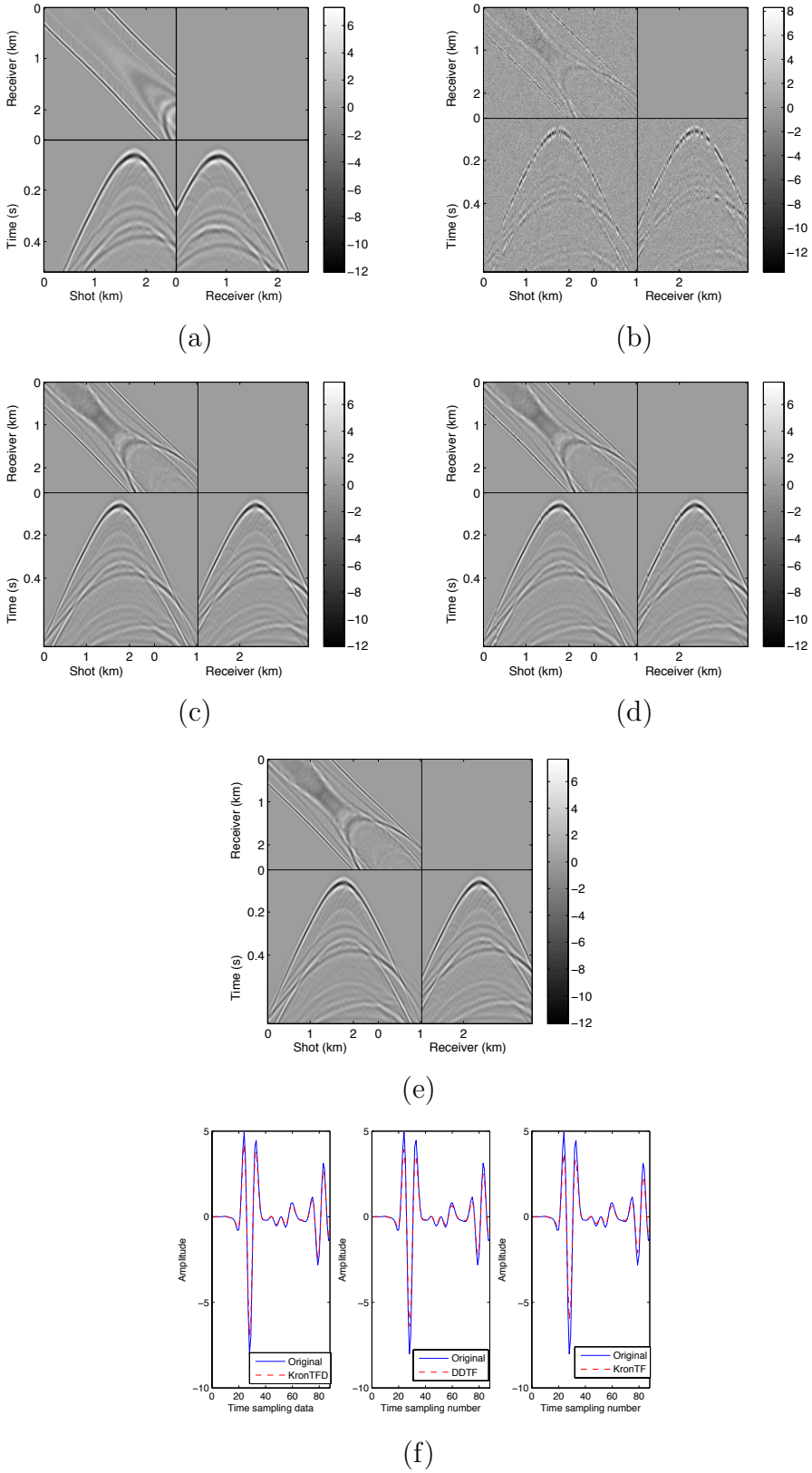
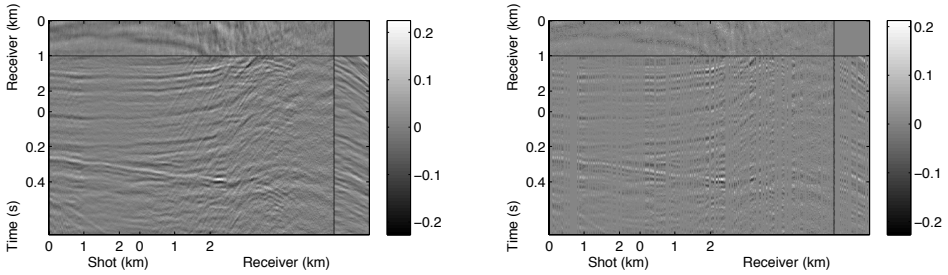
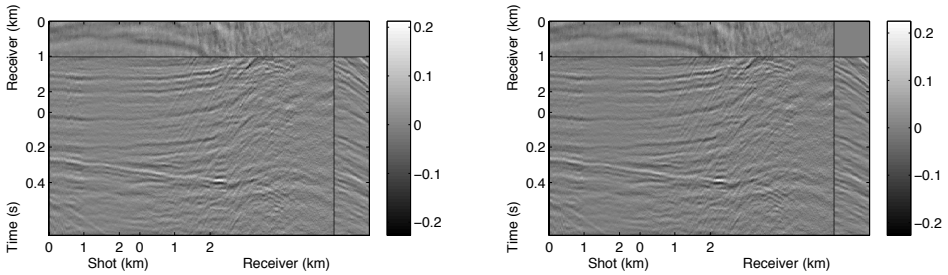


Figure 11. Interpolation of synthetic seismic 3D data with data size $178 \times 178 \times 128$. (a) Original data. (b) Data with 50% randomly missing traces. (c)-(e) Interpolation by DDTF, KronTF and KronTFD.



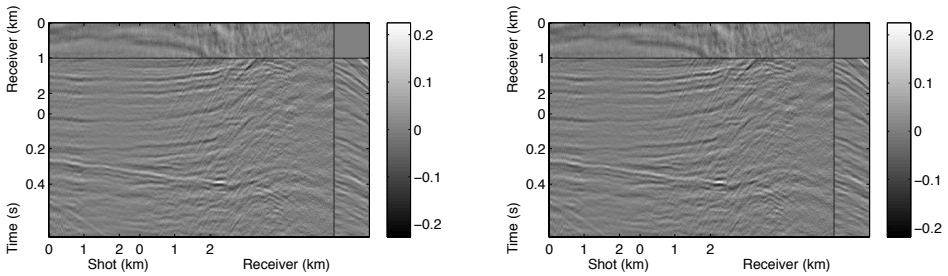
(a)

(b)



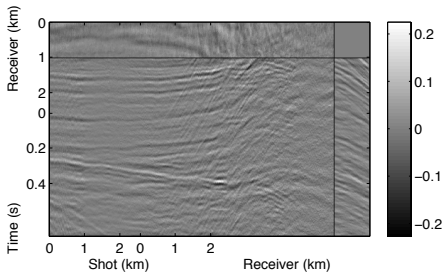
(c)

(d)



(e)

(f)



(g)

Figure 12. Interpolation of real 3D marine data with data size $251 \times 401 \times 50$. (a) Original data; (b) Data with 50% randomly missing traces. (c)-(g) Interpolation by POCS method, Curvelet method, DDTF, KronTF and KronTFD.

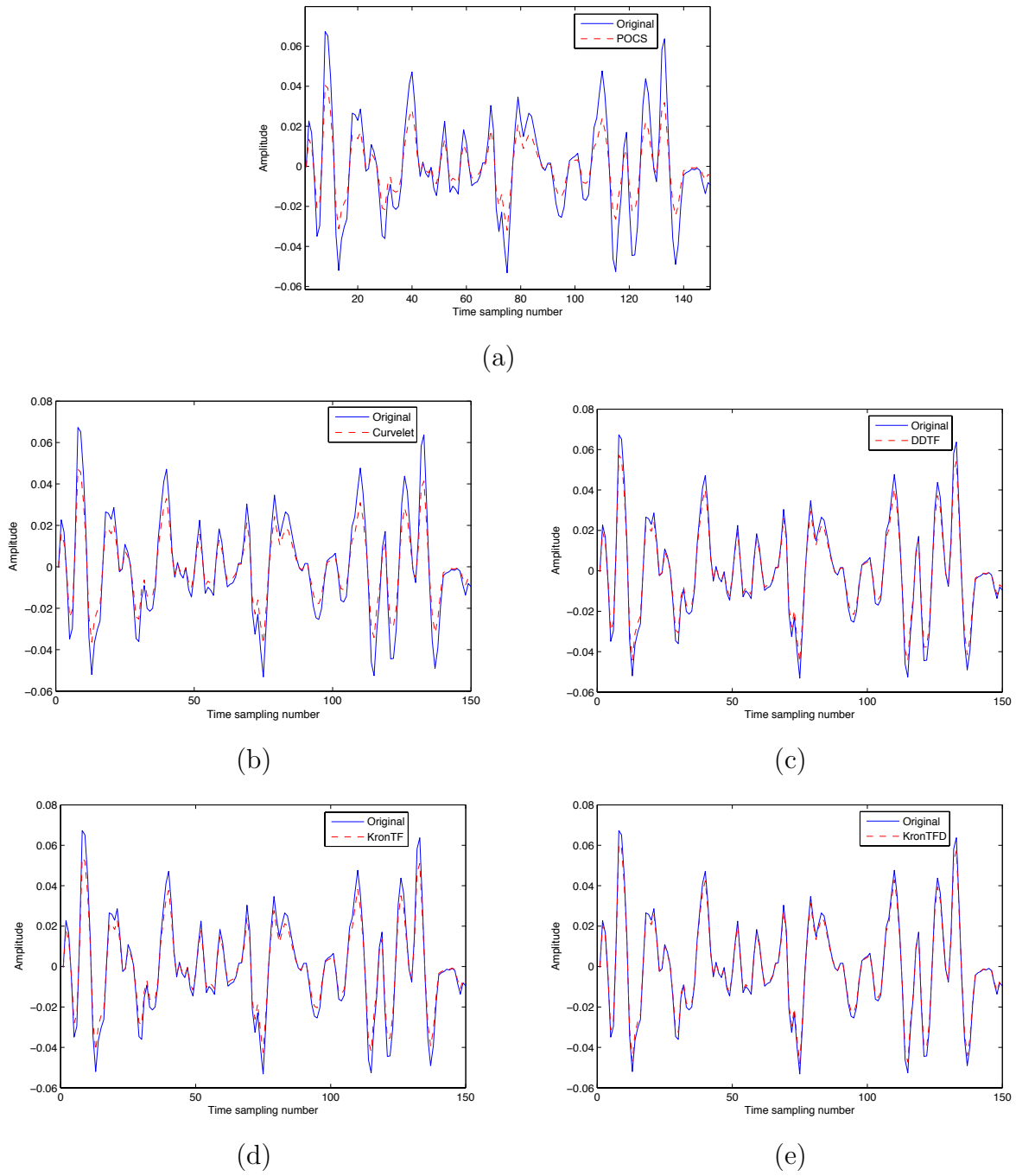


Figure 13. Single trace comparison for the recovery results in Figure 12 and the original Figure 12 (a). Here we have (a)POCS method, (b) curvelet denoising, (c)-(e) application of the data-driven frames DDTF, KronTF and KronTFD.

8. Acknowledgement

This work is supported by NSFC (grant number: NSFC 91330108, 41374121, 61327013), and the Fundamental Research Funds for the Central Universities (grant number: HIT.PIRS.A201501).

9. Reference

[1] Aharon M, Elad M and Bruckstein A 2006 The K-SVD: An algorithm for designing of overcomplete dictionaries for sparse representation IEEE Trans Signal Process **54 4311-4322**
[2] Abma R and Kabir N 2006 3D interpolation of irregular data with a POCS algorithm Geophysics **71 E91-E97**
[3] Bechouche S and Ma J 2014 Simultaneously dictionary learning and denoising for seismic data Geophysics **79 A27-A31**
[4] Beck A and Teboulle M 2009 A fast iterative shrinkage-thresholding algorithm for linear inverse problems SIAM J. Imaging Sci. **2 183-202**
[5] Bellmann R E 1978 Matrix Analysis McGraw-Hill NY
[6] Caiafa C and Cichocki A 2013 Multidimensional compressed sensing and their applications Wiley Interdiscip. Rev. Data Min. Knowl. Discov **3 355-380**
[7] Caiafa C and Cichocki A 2013 Computing sparse representations of multidimensional signals using Kronecker bases Neural Computation **15 186-220**
[8] Cai J, Ji H, Shen Z and Ye G 2014 Data-driven tight frame construction and image denoising Appl. Comput. Harmon. Anal **37 89-105**
[9] Chambolle A and Pock T 2011 A first order primal-dual algorithm for convex problems with applications to imaging J. Math. Imaging Vis. **40 120-145**
[10] Candes E and Romberg J 2007 Sparsity and incoherence in compressive sampling Inverse problems **23 969**
[11] Chen P, Huang J and Zhang X 2013 A primalCdual fixed point algorithm for convex separable minimization with applications to image restoration Inverse Problems **29 025011**
[12] Elad E 2006 Why simple shrinkage is still relevant for redundant representations? IEEE Trans. Inform Theory **52 5559-5569**
[13] Engan K, Aase S O and Husoy J H 1999 Method of optimal directions for frame design IEEE International Conference on **5 2443-2446**
[14] Fomel S and Liu Y 2010 Seislet transform and seislet frame Geophysics **75 V25-V38**
[15] Gao J, Chen X, Li J, Liu G and Ma J 2010 Irregular seismic data reconstruction based on exponential threshold model of POCS method Applied Geophysics **7 229-238**
[16] Herrmann F and Hennenfent G 2008 Non-parametric seismic data recovery with curvelet frames Geophysical Int. J. **173 233-248**
[17] Hennenfent G and Herrmann F 2008 Simply denoise: wavefield reconstruction via jittered undersampling Geophysics **73 V19-V28**
[18] Jolliffe I 2011 Principal component analysis: International Encyclopedia of Statistical Science Springer Berlin Heidelberg **1094-1096**
[19] Kolda T and Bader B 2009 Tensor decompositions and applications SIAM Review **51 455-500**
[20] Häuser S and Ma J 2012 Seismic data reconstruction via shearlet-regularized directional inpainting preprint
[21] Kabir M and Verschuur D 1995 Restoration of missing offsets by parabolic Radon transform Geophysical Prospecting **43 347-368**
[22] Kreimer N and Sacchi M D 2012 A tensor higher-order singular value decomposition for prestack seismic data noise reduction and interpolation Geophysics **77 V113-V122**
[23] De Lathauwer L, de Moor B and Vandewalle J 2000 A multilinear singular value decomposition SIAM J. Matrix Anal. Appl. **21 1253-1278**

[24] Liang J, Ma J and Zhang X 2014 Seismic data restoration via data-driven tight frame Geophysics **79 V65-V74**
[25] Liu B and Sacchi M 2004 Minimum weighted norm interpolation of seismic records Geophysics **69 1560-1568**
[26] Loock S and Plonka G 2014 Phase retrieval for Fresnel measurements using a shearlet sparsity constraint Inverse Problems **30 055005**
[27] Naghizadeh M and Sacchi M 2010 Beyond alias hierarchical scale curvelet interpolation of regularly and irregularly sampled seismic data Geophysics **75 WB189-WB202**
[28] Sacchi M, Ulrych T and Walker C 1998 Interpolation and extrapolation using a high-resolution discrete Fourier transform IEEE Trans. Signal Process. **46 31-38**
[29] Shahidi R, Tang G, Ma J and Herrmann F 2013 Applications of randomized sampling schemes to curvelet-based sparsity-promoting seismic data recovery Geophysical Prospecting **61 973-997**
[30] Spitz S 1991 Seismic trace interpolation in the F-X domain Geophysics **56 785C794**
[31] Tan S, Zhang Y, Wang G, Mou X, Cao G, Wu Z and Yu H 2015 Tensor-based dictionary learning for dynamic tomographic reconstruction Physics in Medicine and Biology **60 2803-2818**
[32] Trad D, Ulrych T and Sacchi M 2002 Accurate interpolation with high-resolution time-variant Radon transforms Geophysics **67 644-656**
[33] Trad D 2009 Five-dimensional interpolation: Recovering from acquisition constraints Geophysics **74 V123-V132**
[34] Tropp J 2004 Greed is good: Algorithmic results for sparse approximation IEEE Trans. Inform. Theory **50 2231-2242**
[35] Tropp J A and Gilbert A C 2007 Signal recovery from random measurements via orthogonal matching pursuit IEEE Trans. Inf. Theory **53 4655-4666**
[36] Van Loan C 2000 The ubiquitous Kronecker product J. Comput. Appl. Math. **123 85-100**
[37] Vidal R, Ma Y and Sastry S 2005 Generalized principal component analysis(GPCA) IEEE Trans. Pattern Anal. Mach. Intell. **27 1945-1959**
[38] Xu S, Zhang Y, Pham D and Lambar G 2005 Antileakage Fourier transform for seismic data regularization Geophysics **70 V87-V95**
[39] Yang K, Ma J and Fomel S 2016 Double sparsity dictionary for seismic noise attenuation Geophysics **81 V103-V116**
[40] Yu S, Ma J, Zhang X and Sacchi M 2015 Denoising and interpolation of high-dimensional seismic data by learning tight frame Geophysics **80 V119-V132**
[41] Zou H, Hastie T and Tibshirani R 2006 Sparse principal component analysis J. Comput. Graph. Statist. **15 265-286**