Improving seismic data completion and efficiency using tensors

Jonathan Popa, Susan Minkoff and Yifei Lou, University of Texas at Dallas*

SUMMARY

The data completion problem involves recovering missing traces from an under-sampled observation. This problem can be solved using the tensor nuclear norm complexity measure and the alternating direction method of multipliers (ADMM) optimization algorithm. The computational cost of ADMM is dominated by the tensor singular value decomposition (tSVD). Reordering data in an $n_1 \times n_2 \times n_3$ tensor so that n_2 is the smallest dimension and n_3 is the largest yields the most cost efficient orientation. Random sampling produces a higher rank observation, making low rank recovery meaningful, whereas regular sampling produces a lower rank observation, for which low rank recovery fails. Data redundancy also results in improved recovery. We validate these claims by applying our tensor completion algorithm to the Viking Graben Field, offshore Norway.

INTRODUCTION

Seismic surveys cannot fully sample a field of interest since it is not possible to have a receiver at every physical location. Traces can also be incomplete or missing when receivers fail. Spatial under-sampling negatively impacts data processing and analysis. Data completion aims to recover missing traces to mitigate the impact of incomplete sampling. Mathematically, this problem can be formulated as

$$\mathbf{Y} = A(\mathbf{X}) + \mathbf{N}, \tag{1}$$

where $\mathbf{Y}$ is an observation produced by applying a sampling operator A to the true seismic data $\mathbf{X}$, and $\mathbf{N}$ is noise. The sampling operator has the same dimensions as $\mathbf{X}$, with ones where data is sampled and zeros elsewhere. Here we will only consider the case where $\mathbf{N} = 0$ to examine the recovery problem without denoising.

There is a rich literature on data completion methods for a variety of applications including seismic data reconstruction (Kreimer and Sacchi (2012a)), remote sensing (Ng et al. (2017)), and completion of color images and videos (Long et al. (2019)). In particular, seismic data often contains many redundancies, which means the data can be well approximated by a low-rank tensor. For matrices, the rank is the number of non-zero singular values. Unfortunately, the rank is NP-hard to minimize and a typical approach is to approximate the rank by the summation of singular values (Semerci et al. (2014)). For tensors we instead use the tensor nuclear norm (TNN) (Ely et al. (2015)).

Kumar et al. (2015) proposed three practical principles for using low rank optimization techniques for data completion. First, the original data needs to be low rank, which often naturally occurs in seismic data. Second, the sampling scheme needs to increase the singular values; otherwise the observation has even lower rank than the true data, thus preventing low rank recovery from being meaningful. The last principle is that the optimization algorithm needs to restore low rank structure. A successful method to minimize the TNN is the alternating direction method of multipliers (ADMM) (Boyd et al. (2010)), which has been used for seismic tensor reconstruction (Ely et al. (2015), Kreimer et al. (2013)). Other techniques include higher-order singular value decomposition methods (Kreimer and Sacchi (2012b), Gao and Sacchi (2018)), parallel matrix factorization schemes (Gao et al. (2015), Sacchi and Cheng (2017)), randomized QR decomposition (Carozzi and Sacchi (2017), Cheng and Sacchi (2015)), minimum weighted norm interpolation (Sacchi et al. (2017)), and adaptive weighted tensor completion (Ng et al. (2017)).

This paper provides a more thorough analysis of low-rank tensor completion for seismic data. First, we analyze the computational cost of ADMM, which suggests the most efficient way of ordering the data into tensor form. Second, we demonstrate that high redundancy occurs both when more data is available and when data is collected physically close together, improving seismic data recovery. Third, we examine two types of sampling operators: regular sampling and random sampling. Regular sampling refers to measuring data with equal spacing in each of the 3 dimensions, while random sampling means taking measurements arbitrarily. We show that random sampling increases the TNN of the original data, and thus, minimizing the TNN gives better recovery results than with regular sampling.

OPTIMIZATION AND COST ANALYSIS

We consider the general case where $\mathbf{Y}$ and $\mathbf{X}$ are tensors. The order of a tensor refers to the dimension of the tensor. We focus on order-3 tensors (with the three dimensions corresponding to time, offset, and bin number), but our results generalize to higher order tensors. An order-3 tensor can be split into matrices, each of which is referred to as a slice.

From the data completion problem (1), we can formulate the following optimization problem to recover low-rank real data $\mathbf{X}$:

$$\min h(\mathbf{X}) \quad \text{s.t.}\, ||\mathbf{Y} - A(\mathbf{X})|| < c. \tag{2}$$

Here h is a complexity measure and c is a threshold bounding the residual. If the complexity measure h is chosen to be rank, the optimization problem is non-convex and NP-hard (Semerci et al. (2014)), so instead we use the tensor nuclear norm (TNN) for the complexity measure, defined as (Ely et al. (2015)):

$$||\mathbf{X}||_{TNN} = \sum_{i=1}^{N} ||\hat{\mathbf{X}}(:,:,i)||_{nuc}, \tag{3}$$

where $\hat{\mathbf{X}}$ is the Fourier transform of $\mathbf{X}$, $\hat{\mathbf{X}}(:,:,i)$ is the i^{th} slice of $\hat{\mathbf{X}}$, and $||\cdot||_{nuc}$ is the matrix nuclear norm. Furthermore, we set the threshold $c = 0$ to express the optimization problem as:

$$\min ||\mathbf{X}||_{TNN} \quad \text{s.t.}\, \mathbf{Y} = A(\mathbf{X}). \tag{4}$$

With the use of the TNN, (4) is a convex optimization problem (Zhang et al. (2014)). To solve this optimization problem, we consider the ADMM algorithm (Boyd et al. (2010)).

For brevity we do not present the full ADMM algorithm in this paper, but note that the dominant part of the computational cost of the ADMM algorithm applied to (4) is attributed to the tensor singular value decomposition (tSVD). The tSVD is a generalization of the matrix SVD algorithm, decomposing a tensor into a product of three tensors. As depicted in Figure 1

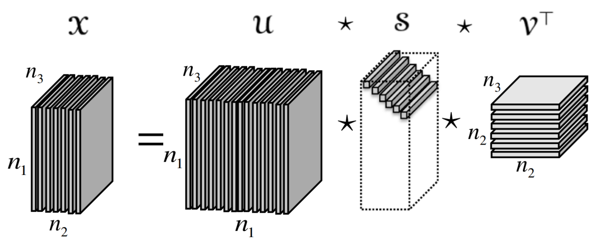
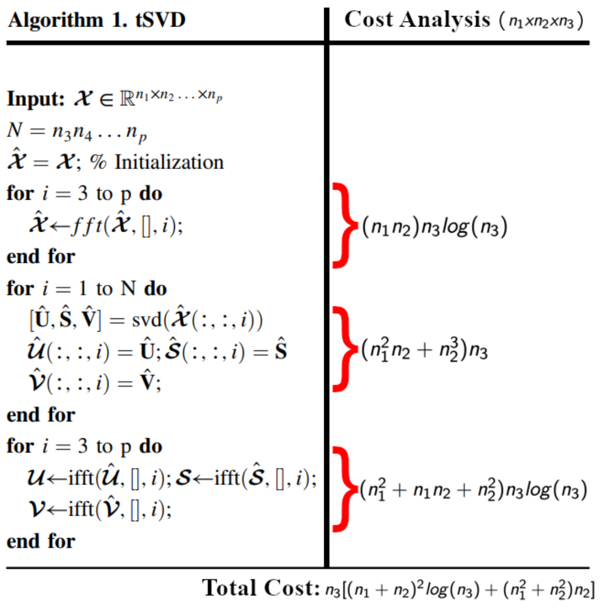


Figure 1: Illustration of tSVD (Ely et al. (2015))

we have $\mathbf{X} = \mathbf{U} * \mathbf{S} * \mathbf{V}$, where $\mathbf{S}$ is a diagonal tensor, in which each slice is a diagonal matrix, and tensors $\mathbf{U}$ and $\mathbf{V}$ are orthogonal tensors, for which the product of each with their own transpose results in the identity tensor. The identity tensor has the identity matrix as its first slice and zeros for all other slices.



```
Algorithm 1. tSVD                                   | Cost Analysis ($n_1 x n_2 x n_3$)
Input: $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \ldots \times n_p}$
$N = n_3 n_4 \ldots n_p$
$\hat{\mathcal{X}} = \mathcal{X}$; % Initialization
for $i = 3$ to p do
    $\hat{\mathcal{X}} \leftarrow fft(\hat{\mathcal{X}}, [], i)$;       } $(n_1 n_2) n_3 log(n_3)$
end for
for $i = 1$ to N do
    $[\hat{\mathbf{U}}, \hat{\mathbf{S}}, \hat{\mathbf{V}}] = \mathrm{svd}(\hat{\mathcal{X}}(:,:,i))$
    $\hat{\mathcal{U}}(:,:,i) = \hat{\mathbf{U}}; \hat{\mathcal{S}}(:,:,i) = \hat{\mathbf{S}}$   } $(n_1^2 n_2 + n_2^3) n_3$
    $\hat{\mathcal{V}}(:,:,i) = \hat{\mathbf{V}}$;
end for
for $i = 3$ to p do
    $\mathcal{U} \leftarrow \mathrm{ifft}(\hat{\mathcal{U}}, [], i); \mathcal{S} \leftarrow \mathrm{ifft}(\hat{\mathcal{S}}, [], i)$;   } $(n_1^2 + n_1 n_2 + n_2^2) n_3 log(n_3)$
    $\mathcal{V} \leftarrow \mathrm{ifft}(\hat{\mathcal{V}}, [], i)$;
end for
                    Total Cost: $n_3[(n_1 + n_2)^2 log(n_3) + (n_1^2 + n_2^2) n_2]$
```

Figure 2: tSVD Algorithm and Cost Analysis

Figure 2 presents our cost analysis of the tSVD algorithm for order-3 tensors. First, the fast Fourier transform (FFT) is computed along each tube of the tensor. A tube is a vector oriented along the third dimension of the tensor. The cost of FFT on a vector of length n is $\mathcal{O}(n \log(n))$. The Fourier transform is taken for $n_1 \times n_2$ many vectors, each of length n_3, thus costing $\mathcal{O}((n_1 n_2) n_3 \log(n_3))$. The next part of the tSVD algorithm takes the matrix SVD of each slice. Matrix SVD for a matrix of size $n_1 \times n_2$ costs $\mathcal{O}(n_1^2 n_2 + n_2^3)$, and since the matrix SVD is performed for n_3 slices, the cost is $\mathcal{O}((n_1^2 n_2 + n_2^3) n_3)$. The last loop of the algorithm takes the inverse Fourier transform (IFFT) of the tensors $\mathbf{U}$, $\mathbf{S}$, and $\mathbf{V}$ along the tubal direction. Again multiplying the number of tubes by the cost of the FFT results in a cost of $\mathcal{O}((n_1^2 + n_1 n_2 + n_2^2) n_3 \log(n_3))$. Adding these costs and simplifying we obtain the total cost of the tSVD algorithm:

$$\mathcal{O}(n_3[(n_1 + n_2)^2 \log(n_3) + (n_1^2 + n_2^2) n_2]). \tag{5}$$

It is important to note here that n_1 appears quadratically, n_2 cubically, and n_3 as linear and logarithmic. This cost analysis

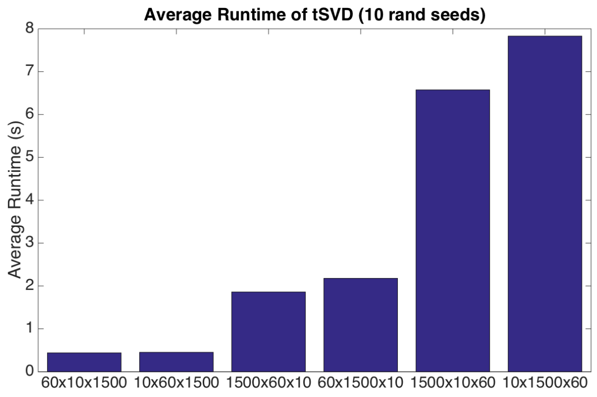


Figure 3: Average Runtime of tSVD

implies that orientation of the tensor impacts runtime of the tSVD algorithm. To demonstrate the runtime variations, we generate ten random tensors of size $1500 \times 60 \times 10$. For each tensor we permute the dimensions and calculate the runtime of the tSVD. Figure 3 shows the average runtime for each orientation of the data. We note that it is most efficient when the smallest dimension is n_2 and the largest dimension is n_3, as these are the cubic and linear/logarithmic terms respectively in the above cost analysis.

DISCUSSION OF RESULTS

We examine the seismic data completion problem using real data from the Viking Graben Field, offshore Norway. The data set contains seismic streamer pressure data recorded in one acquisition line. Each CMP gather contains 60 traces with offsets ranging from 262 m to 3212 m. All traces have a 6 s time window and 0.004 s sampling. In total there are 1900 CMP gathers, each of which can be considered as a slice of an order-3 tensor. For the following experiments, only a subset of this data is used. Except where otherwise stated, the sampling operator we use randomly selects 60% of the traces in each slice, then randomly removes 90% of each selected trace. This sampling operator reflects the real world scenario when receivers fail.

We begin by examining the performance of the ADMM algorithm on the first 100 slices of the CMP data. Three slices of

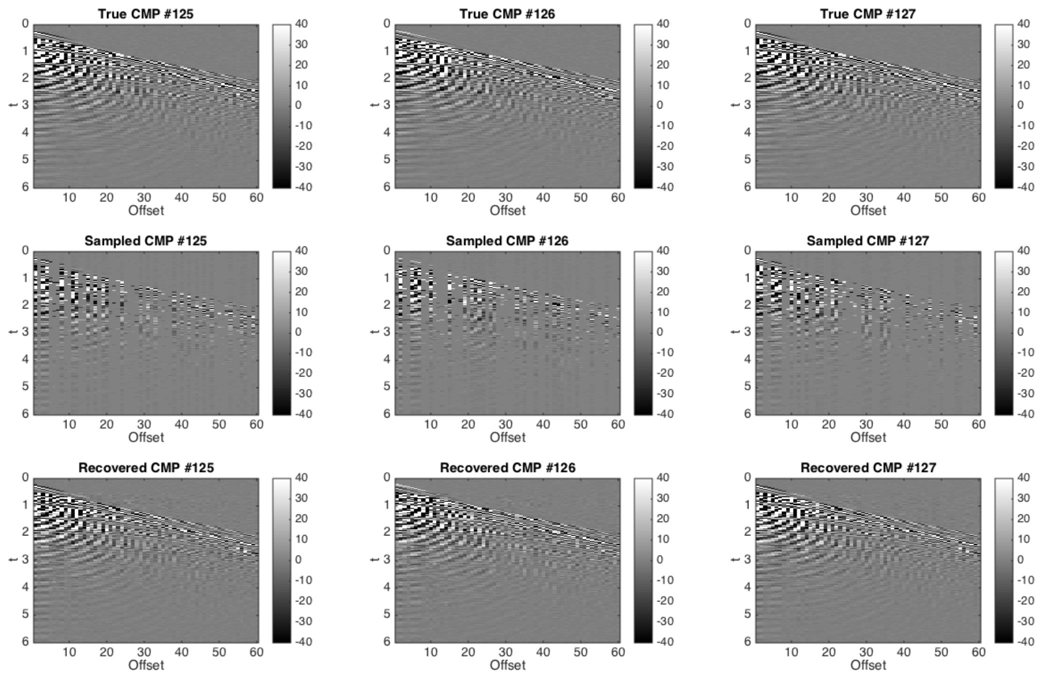


Figure 4: From top to bottom: the first three slices of the true CMP gathers, the sampled gathers, where 60% of the traces in each gather have 90% of the trace removed, and the recovered gathers after 20 iterations of ADMM

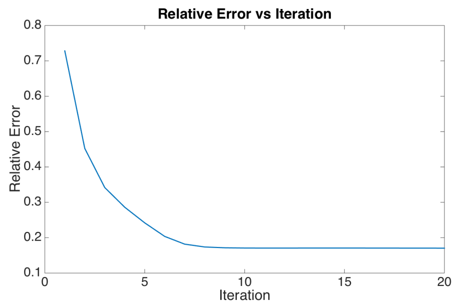


Figure 5: Error plot over 20 iterations of ADMM using 100 slices of data. ADMM converges with error around 17%.

the original data are shown on the top row of Figure 4. After applying the sampling operator, the resulting measurements, referred to as sampled data, are given on the middle row of Figure 4. The last row of Figure 4 shows the recovered gathers after 20 iterations of ADMM. We also plot relative error to the ground-truth versus ADMM iterations in Figure 5, which shows that ADMM converges in 20 iterations and the final relative error is around 17%. As little additional gain in recovery occurs after this number of iterations, for the rest of the experiments in this paper, we fix the number of iterations of ADMM at 20.

As Figure 4 demonstrates satisfactory recovery results using 100 CMP gathers (or slices), it is natural to ask to what extent the number of slices affects the recovery. We therefore vary the number of slices and the stride spacing between slices from the full dataset of 1900 CMP gathers. Stride spacing refers to the indexing of the slices, for instance, stride spacing 1 uses consecutive slices; stride spacing 2 uses every other slice, etc. From the full CMP gather dataset, we use either 2, 3, 5, 10, 20, 30, 60, or 100 gathers with stride spacing 1, 10, or 19. For each combination of gathers and stride, we randomly generate 10 different sampling operators. The average errors after recovery of the 10 random realizations are plotted in Figure 6, which shows that error decreases with more gathers and smaller stride. While increasing the number of slices up to about 10 results in improved recovery, beyond this number little additional improvement occurs. We note also that the computational complexity grows at least linearly with the number of slices indicating a tradeoff between number of slices and cost.

To understand why smaller stride also helps recovery, we examine the TNN values for different stride spacing in Figure 7. As the stride spacing increases, the TNN increases, which implies that the ground-truth signal is not well approximated by a low rank tensor. In seismic surveys data collected closer together physically (i.e. small stride spacing) is often correlated, thus having higher redundancies and lower rank. Both more slices and smaller strides indicate that the original signal has a low-rank structure, thus improving our chances of recovery.

For compressive sensing Candes (2006) showed random sampling has higher probability of success than regular sampling. Regular sampling uniformly selects traces in each slice to be removed (i.e. removing 50% would remove every other trace). In Figure 8, we compare the effect of regular and random sampling schemes using stride spacing 1 and varying the number of slices used. Specifically in Figure 8(a), we observe that even

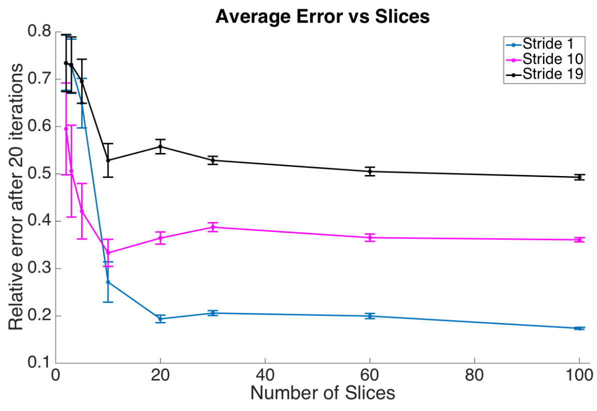


Figure 6: Final error after 20 iterations of ADMM was averaged over 10 random seeds for each combination of either 2, 3, 5, 10, 20, 30, 60, or 100 gathers with stride spacing 1 (blue), 10 (magenta), or 19 (black). The error bars show the standard deviation. In general, error decreases with the use of more slices and with smaller stride.

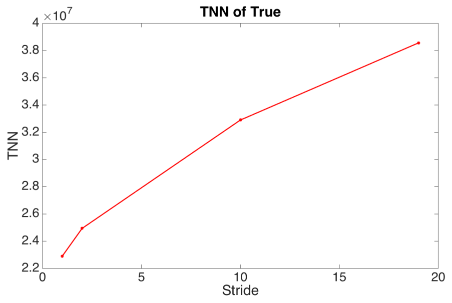


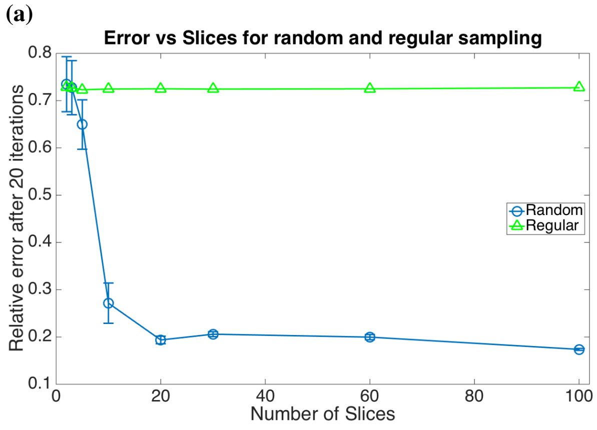
Figure 7: TNN calculation for one-hundred slices of CMP with different stride spacing. TNN increases as stride increases, resulting in tensors which are not as well approximated by low-rank tensors.

as the number of slices increases, the change in error for regular sampling is nearly constant. This result differs from random sampling, for which the error decreases as more slices are used. To examine why regular sampling performs worse, we plot the TNN value of the true, randomly and regularly sampled data sets for varying amounts of data in Figure 8(b), which shows that random sampling increases the TNN, whereas regular sampling decreases the TNN. These results confirm that the sampling operator needs to increase the singular values in order for low rank optimization techniques to succeed.

CONCLUSIONS

Based on our cost analysis, we are able to determine the most computationally efficient orientation of a tensor for the ADMM algorithm. By choosing the orientation with n_2 as the smallest dimension and n_3 as the largest dimension, the runtime can be drastically reduced. Our empirical results support the fact that as more data (CMP gathers) are used, more redundancy occurs in the data set, which leads to better recovery. However,

(a)



(b)

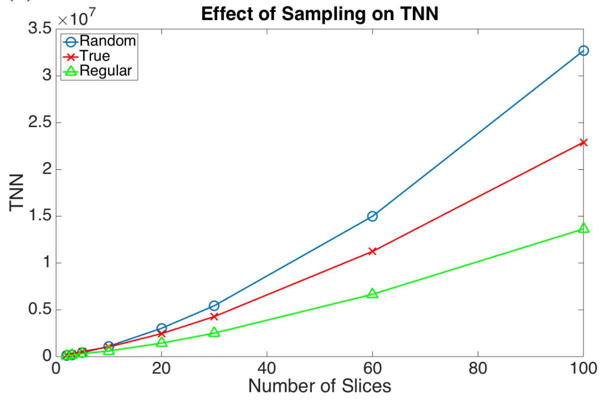


Figure 8: **(a)** Final error after 20 iterations of ADMM for random (blue) and regular (green) sampling of 100 slices of CMP with stride spacing 1. Using more slices does not improve recovery in the case of regular sampling. **(b)** TNN of 100 slices of CMP with stride spacing 1 before sampling (red), with random sampling (blue) and with regular sampling (green). Regular sampling decreases the TNN, hence low rank recovery is not meaningful in this case.

it is worth noting that the improvement for this dataset after 10 gathers is diminishing while the runtime increases at least linearly. The results also validate that data closer together physically (i.e. small stride spacing) contributes to better recovery than larger strides, again because more redundancies leads to lower rank. As for sampling, random sampling produces observations with a greater TNN value than the true, allowing for low rank recovery to successfully minimize the TNN. In contrast, regular sampling produces observations with lower TNN value, thereby reducing the ability of low rank recovery to succeed.

ACKNOWLEDGEMENTS

We thank Mobil Oil (now ExxonMobil) for making publicly available the seismic data set from the Viking Graben Field. We also thank Wei Zhou and David Lumley for providing this CMP data for our experiments.

REFERENCES

Boyd, S., N. Parikh, E. Chu, B. Peleato, and J. Eckstein, 2010, Distributed optimization and statistical learning via the alternating direction method of multipliers: Foundations and Trends in Machine Learning, **3**, 1–122.

Candes, E., J. Romberg, and T. Tao, 2006, Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information: IEEE Transactions on Information Theory, **52**, 489–509.

Carozzi, F., and M. D. Sacchi, 2017, 5D seismic reconstruction via PMF with randomized QR decomposition: SEG, 4251–4256.

Cheng, J., and M. D. Sacchi, 2015, A fast rank-reduction algorithm for 3D deblending via randomized QR decomposition: SEG, 3830–3835.

Ely, G., S. Aeron, N. Hao, and M. E. Kilmer, 2015, 5D seismic data completion and denoising using a novel class of tensor decompositions.: Geophysics, **80**, V83 – V95.

Gao, J., A. Stanton, and M. D. Sacchi, 2015, Parallel matrix factorization algorithm and its application to 5D seismic reconstruction and denoising: Geophysics, **80**, 173–187.

Gao, W., and M. D. Sacchi, 2018, PP-wave and PS-wave 5D reconstruction and denosing to assist registration: SEG, 2397–2401.

Kreimer, N., and M. D. Sacchi, 2012a, Tensor completion via nuclear norm minimization for 5D seismic data reconstruction: SEG, 1–5.

———, 2012b, A tensor higher-order singular value decomposition for prestack seismic data noise reduction and interpolation: Geophysics, **77**, 113–122.

Kreimer, N., A. Stanton, and M. D. Sacchi, 2013, Tensor completion based on nuclear norm minimization for 5D seismic data reconstruction: Geophysics, **78**, 273–284.

Kumar, R., C. Da Silva, O. Akalin, A. Y. Aravkin, H. Mansour, B. Recht, and F. J. Herrmann, 2015, Efficient matrix completion for seismic data reconstruction: Geophysics, **80**, V97 – V113.

Long, Z., Y. Liu, L. Chen, and C. Zhu, 2019, Low rank tensor completion for multiway visual data: Signal Processing, 301–316.

Ng, M. K.-P., Q. Yuan, L. Yan, and J. Sun, 2017, An adaptive weighted tensor completion method for the recovery of remote sensing images with missing data: IEEE Transactions on Geoscience and Remote Sensing, **55**, 3367–3381.

Sacchi, M. D., and J. Cheng, 2017, 5D reconstruction via robust tensor completion: GeoConvention, 1–5.

Sacchi, M. D., G. Garabito, H. Schots, and J. Caldeira, 2017, Preconditioning and denoising prestack onshore seismic data via 5D reconstruction: application to 3D data from the parnaba basin: SBGf, 1384–1388.

Semerci, O., N. Hao, M. E. Kilmer, and E. L. Miller, 2014, Tensor-based formulation and nuclear norm regularization for multienergy computed tomography: IEEE Transactions on Image Processing, **23**, 1678–1693.

Zhang, Z., G. Ely, S. Aeron, N. Hao, and M. Kilmer, 2014, Novel methods for multilinear data completion and denoising based on tensor-SVD: IEEE Conference on Computer Vision and Pattern Recognition, 3842–3849.