High-dimensional wave atoms and compression of seismic datasets

Matti Leinonen[1], Russell J. Hewett[2,3(*)], Xiangxiong Zhang[2], Lexing Ying[4], and Laurent Demanet[2,3]
[1]Aalto University, [2] Dept. of Mathematics, MIT, [3] Earth Resources Lab, MIT, [4]Stanford University

SUMMARY

Wave atoms are a low-redundancy alternative to curvelets, suitable for high-dimensional seismic data processing. This abstract extends the wave atom orthobasis construction to 3D, 4D, and 5D Cartesian arrays, and parallelizes it in a shared-memory environment. An implementation of the algorithm for NVIDIA CUDA capable graphics processing units (GPU) is also developed to accelerate computation for 2D and 3D data. The new transforms are benchmarked against the Fourier transform for compression of data generated from synthetic 2D and 3D acoustic models.

INTRODUCTION

Wave atoms are a relatively recent addition to the repertoire of harmonic analysis transforms. They were introduced by Demanet and Ying (2007a,b) for the efficient representation of oscillatory patterns in image processing, e.g., the coda portion of 2D seismic shot gathers. They were also used by Demanet and Ying (2009) to define a fast solver for time-dependent acoustic wave propagation in heterogeneous media, not limited by the CFL condition. The basis of wave atoms is a careful construction of Gaussian-like wave packets that allows fast forward and inverse transforms.

Wave atoms share similarities to the tight frame of curvelets introduced by Candés and Donoho (2004); Candés et al. (2006) and used extensively in seismic imaging (Hennenfent and Herrmann, 2006; Herrmann et al., 2007, 2008):

- The basis elements are multiscale directional wave packets and are localized both in space ($\mathbf{x}$) and in wavevector ($\mathbf{k}$) space.
- As the scale is refined, the wavelength of the oscillations decreases proportional to the square of the diameter of the wave packet's essential support.
- The transform is computed with fast algorithms based on the fast Fourier transform (FFT).

The two transforms are however markedly different. High-quality curvelets have redundancy 7.5 in 2D and 50 in 3D (Candés et al., 2006; Ying et al., 2005), but wave atoms are designed as an orthonormal basis, and hence have redundancy 1 regardless of the dimension. This is advantageous in high-dimensional situations where redundancy causes an intractable memory overhead. While curvelets can efficiently represent narrow, anisotropic bandlimited wavefronts, wave atoms are more similar to plane waves localized by isotropic envelopes, hence can be viewed as geometrically simpler. Additionally, in contrast to Gabor and short-time Fourier transforms, forward and inverse wave atom transforms can both be computed fast. A discussion of how wave atoms compare and contrast with other directional transforms can be found in Demanet (2006). For other wavelet-based alternatives for seismic data compression, see Wavelet Packets in Wu et al. (2006); Wang et al. (2010), high-dimensional Wavelets in Villasenor et al. (1996), Dreamlets in Geng et al. (2009), Seislets in Fomel (2006), and also Duval and Nguyen (1999); Vassiliou and Wickerhouser (1997); Averbuch et al. (2001).

In this note we show that the construction of wave atoms can be extended in a natural way to an arbitrary number of dimensions, with complexity comparable to that of the FFT. Implementations of the resulting transforms for conventional computing clusters and for graphical processing units (GPU) are potentially competitive tools for large-scale seismic data processing. Beyond data compression, possible applications include data denoising, interpolation of missing or unequispaced data, recovery from shot aggregates (encoded sources), and sparse regularization for full waveform inversion.

TRANSFORM ARCHITECTURE

The wave atom basis elements are functions of $\mathbf{x} = (x_1, \ldots, x_d)$ and are indexed by integer vectors $\mu = (j, \mathbf{m}, \mathbf{n})$, with scale j, wave vector index $\mathbf{m} = (m_1, \ldots, m_d)$, and spatial translation index $\mathbf{n} = (n_1, \ldots, n_d)$. The wave atom architecture involves a precise set of 1D template profiles $\psi_m(x)$ defined by Equation 7 in Demanet and Ying (2007a), such that their Fourier transforms $\widehat{\psi}_m(k)$ tile k-space and the spatial translates $\psi_m(x-n)$ form an orthonormal basis. High-dimensional wave atoms are then formed as individual tensor products of these 1D profiles, dilated or contracted to the same dyadic scale j,

$$\varphi_\mu(\mathbf{x}) = 2^{jd/2} \left(\psi_{m_1}(2^j x_1 - n_1) \times \ldots \times \psi_{m_d}(2^j x_d - n_d) \right).$$

The Fourier transform $\widehat{\varphi}_\mu(\mathbf{k})$ consists of 2^d bumps localized in the neighborhood of $(\pm k_{1,\mu}, \ldots, \pm k_{d,\mu})$ for all possible sign combinations, where $\mathbf{k}_\mu = (k_{1,\mu}, \ldots, k_{d,\mu}) = \pi 2^j \mathbf{m}$.

To make the collection of wave atoms an orthobasis, the wave vector index $\mathbf{m}$ has to be further restricted as a function of j, so that together $(j, \mathbf{m})$ span the nodes of a wavelet packet tree. We adopt the same choice of restriction as Demanet and Ying (2007b,a, 2009), that is $C_1 2^j \leq \|\mathbf{m}\|_\infty \leq C_2 2^j$ for adequate constants C_1, C_2.

Note that while individual basis elements are tensor products of 1D functions, the resulting collection of φ_μ is *not a tensor product basis*. It cannot be computed by applying 1D transforms dimension by dimension (like the Fourier transform). A tensor product basis would require as many scale indices $j_1, \ldots, j_d$ as there are dimensions; in contrast, our construction

is a native multiresolution with a single scale index j. Harmonic analysts use the word "nonstandard" to refer to such bases, which are not tensor product bases, but whose elements may individually be tensor products.

Functions of $\mathbf{x}$ can then be expanded in the wave atom basis,

$$f(\mathbf{x}) = \sum_{\mu} c_\mu \varphi_\mu, \qquad c_\mu = \langle \varphi_\mu, f \rangle, \tag{1}$$

or $f = Wc$ and $c = W^* f$ for short. Compression is achieved by setting to zero the coefficients below some threshold. Sparse regularization can be done by penalizing the ℓ_1 norm of the coefficients, $\sum_\mu |c_\mu|$, in addition to fitting seismogram data in the scope of an inversion scheme.

Figure 1 shows an example of a 3D wave atom on a $64 \times 64 \times 64$ cube, though much larger grids are considered below. The three pictures on top show the three 2D coordinate plane cross-sections of a 3D wave atom in $\mathbf{x}$, and the three pictures on bottom show the corresponding cross-sections in the wave-vector domain $\mathbf{k}$. Each wave atom is a superposition of 8 plane waves with wave vectors given by all the possible sign combinations of $(\pm k_{1,\mu}, \pm k_{2,\mu}, \pm k_{3,\mu})$, then windowed by a smooth compactly-supported envelope function in the wave-vector domain. As a result, wave atoms are not compactly supported yet they decay quickly in $\mathbf{x}$.

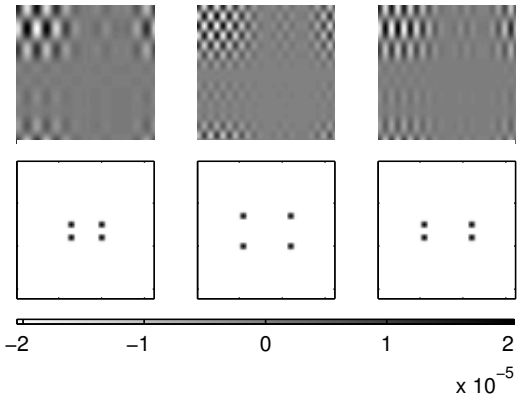

Figure 1: Example of 3D wave atom with scale $j = 1$, wave-vector index $\mathbf{m} = [1,3,5]$, and spatial index $\mathbf{n} = [1,1,1]$. Spatial cross-sections along the coordinate planes (top row) and corresponding wave-vector cross-sections (bottom row).

To put the wave atom construction in perspective, we also implement a variant where the index j is removed, resulting in a "monoscale" wave atom transform. The definition is the same as earlier, with j fixed to an integer value near $\log_2 \sqrt{N}$, for N points per dimension, and without restriction on $\mathbf{m}$. As a consequence, the basis functions indexed by $\mathbf{m}$ now tile $\mathbf{k}$-space in a uniform manner. Monoscale wave atoms can be viewed as a fast Gabor transform with a fast inverse.

At the expense of a redundancy of 2^{d-1}, there exists another "tight-frame" variant of the (multiscale) wave atom with only two bumps in the wave vector domain. For this variant, in which construction involves Hilbert transforms, (1) remains valid. No numerical experiments involving this variant are reported here; see Demanet and Ying (2007b,a, 2009) for details of the 2D case.

ALGORITHM AND PARALLELIZATION

The forward transform $f(\mathbf{x}) \to c_\mu$ is computed a follows. Assume (without loss of generality) that the grid has N points in each dimension, and let L be the width of one bump of $\widehat{\varphi}_\mu(\mathbf{k})$ in $\mathbf{k}$-space.

1. Take a fast Fourier transform (FFT) of size N^d of $f(\mathbf{x})$ to get $\widehat{f}(\mathbf{k})$.
2. For fixed $(j,\mathbf{m})$, wrap $\widehat{f}(\mathbf{k})\overline{\widehat{\varphi}_\mu(\mathbf{k})}$ by periodization to a cube of sides of length L centered at the origin. over the support of $\widehat{\varphi}_\mu(\mathbf{k})$. Then perform a small inverse FFT of size $(L)^d$ of the result, to get the coefficients c_μ indexed by $\mathbf{n}$, for fixed $(j,\mathbf{m})$.
3. Repeat over all $(j,\mathbf{m})$.

The wrapping operation needed to achieve redundancy 1 is detailed in Demanet and Ying (2007b,a). Since wave atoms form an orthonormal basis, the inverse transform is obtained from the adjoint, and is realized by simply undoing the operations above in reverse order. The unwrapping operation, adjoint to wrapping, involves a sum over at most 2^d values of $(j,\mathbf{m})$ for each $\mathbf{k}$, because of the overlap of the basis functions in $\mathbf{k}$-space. Both the forward and inverse transform have overall computational complexity $O(N^d \log N)$, proportional to that of the FFT.

The algorithm for the forward transform is easily parallelized by assigning subsets of $(j,\mathbf{m})$ to different compute nodes. This assignment can be as fine-grained as one $(j,\mathbf{m})$ per node or per CPU. The only necessary communication is the one-to-all broadcast (or "scatter") of the values of $\widehat{f}(\mathbf{k})$ to each $(j,\mathbf{m})$ for which $\varphi_\mu(\mathbf{k}) \neq 0$. The inverse transform results in a slightly more complex parallel algorithm, because it involves an all-to-one reduction (or "gather") operation that overwrites an array in k-space with contributions from different $(j,\mathbf{m})$. Precautions must be taken to avoid two writes to occur simultaneously on the same portion of an array.

The data arrays considered in this paper fit in the shared memory of a single machine. Distributed-memory algorithms for applying the forward and inverse transforms may not be needed in practice, since a larger dataset can always be divided into pieces that can be processed independently.

The GPU implementation of the forward transform involves partitioning the task into *kernels*, each of which execute on *blocks* of lightweight *threads*. Concurrency of threads within a single block is, in a sense, guaranteed, but threads can only communicate within their own block. There is no block level communication: blocks of threads must complete their tasks independently. For the forward wave atom transform, we use one kernel per scale j per quadrant in k-space $k_i^{\pm}$. Each $\mathbf{m}$ is assigned to one block and each $\mathbf{k}$ in the support of $\widehat{\varphi}_\mu$, for that block, is assigned a thread. If the dataset is too large for the

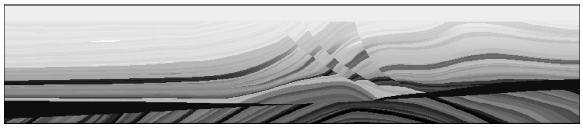

Figure 2: Marmousi2 P-wave velocity model (G. Martin and Marfurt, 2006).

GPU's memory, the kernels can be further refined by partitioning the **m** indices within the pairs $(j,k_i^{\pm})$.

The GPU implementation of the inverse transform is more involved, because multiple blocks contribute to the same locations in the output array. This problem is typically solved by the so-called "atomic add" procedure, but this prohibitively expensive in our case. Instead, we stagger the writes to the output array so that they are sufficiently offset, at the expense of a 2^d-fold increase in the number of kernels.

CUDA-capable GPUs allow for up to three-dimensional block and thread layouts. As such, we do not implement the 4D and 5D transforms on the GPU.

COMPRESSION OF SEISMIC DATA

We generate a shot gather dataset from the two-dimensional Marmousi2 P-wave velocity model, rescaled to a spatial step of 20 m (G. Martin and Marfurt, 2006), shown in Figure 2. The simulations use a constant-density, time-dependent acoustic wave equation discretized using a fourth-order accurate finite difference solver. The domain is surrounded by a perfectly matched layer and 512 equispaced sources and 512 equispaced receivers, per source, at depth $z = 60$ m were considered. The direct wave and first reflection, due to the water-ocean floor interface, are removed from the data set. The simulation results in 8192 time samples per trace, which are then downsampled by a factor 4 to result in a sampling slightly above the Shannon-Nyquist rate. The resulting dataset has the size $512 \times 512 \times 2048$. We also consider a shot gather dataset from a simple three-dimensional synthetic layered salt model, on a $64 \times 64 \times 64$ grid, surrounded by a perfectly matched layer. For 32×32 equispaced sources, and 64×64 equispaced receivers per source, simulated wavefields were sampled 1024 times. The resulting 5D data has the size $32 \times 32 \times 64 \times 64 \times 1024$. The 4D data used in the next section is a slice of the 5D data. All simulations are performed with the authors' Python Seismic Imaging Toolbox (PySIT).

Edge truncation can have a strong effect on compression performance. To quantify edge effects, we consider the original data with and without tapering by a smooth window near the edges in all three coordinates.

Compression performance is measured by the peak signal-to-noise ratio

$$\mathrm{PSNR} = -20\log_{10}\frac{\|d-\widetilde{d}\|}{\|d\|},$$

where $\|\cdot\|$ is the ℓ_2 norm, $\|d-\widetilde{d}\|/\|d\|$ is the relative root mean-squared error (RMSE), and the tilde is the operation of thresholding the small coefficients in modulus. In Figure 4, the PSNR is shown for 3D multiscale wave atoms, 3D monoscale wave atoms, and the 3D Fourier transform, as the function of the proportion of small coefficients dropped in either representation. Both tapered and untapered seismogram data are considered.

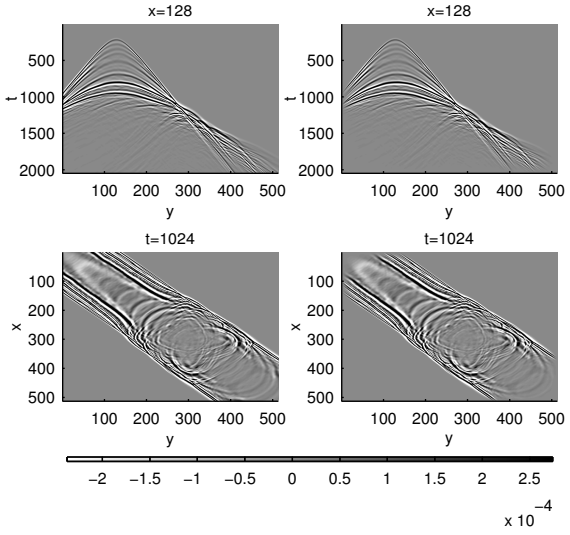

Figure 3: Input data. Top: a single shot at $r = 128$, without and with tapering at the edges. Bottom: time slice at $t = 1024$, without and with tapering at the edges.

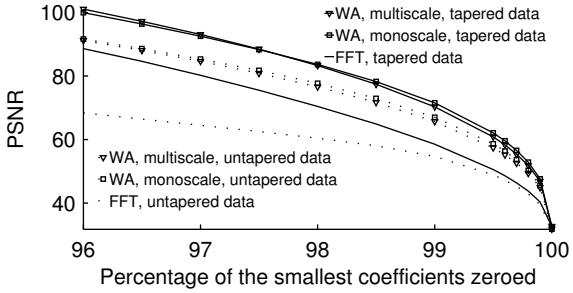

Figure 4: PSNR as the function of the percentage of zeroed small coefficients for various transforms, for both tapered and untapered data.

Figure 4 shows that monoscale and multiscale wave atoms both offer better compression performance than the 3D Fourier transform. Furthermore, the wave atom representations are less sensitive to edge effects than the Fourier transform.

RUNTIME BENCHMARKS

Figure 5 shows runtimes for the forward 3D wave atom computation as the function of the size of the dataset, for both real and complex data, for the conventional (CPU-only) and GPU parallel implementations (complex only). All computa-

tions use double precision. The single core CPU implementation uses 1 FFTW thread and the parallel CPU implementation uses 4 FFTW threads (Frigo and Johnson, 2005). Disk input/output time is not included in the reported times. In the GPU case, the time needed to transfer data from the host to the device, and from the device back to the host, is included. These timings are generated using dual processor Intel Xeon X5690 computer with 96 GB of RAM with an NVIDIA Fermi C2075 GPU.

Figure 5-7 indicate that the computation time grows in proportion to the size of the dataset. Moreover, from Figure 5 indicates that it is possible to obtain around three to five times speed up when using GPU, in contrast to the CPU case with 4 FFTW threads, provided that the transform computation fits on the GPU.

CONCLUSIONS

Wave atoms are a good alternative to the Fourier transform for the task of compression of large seismic datasets. The transform computation parallelizes favorably on shared-memory machines and on GPU.

ACKNOWLEDGMENTS

The authors thank Total SA for supporting this research. LY is grateful to the National Science Foundation. LD is also grateful to the National Science Foundation and the Alfred P. Sloan Foundation.

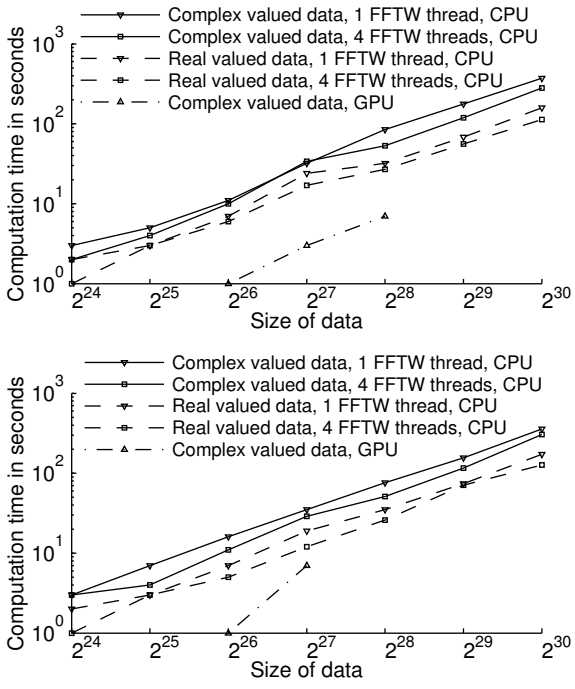

Figure 5: Runtimes for the forward (top) and inverse (bottom) 3D wave atom transform as the function of data volume.

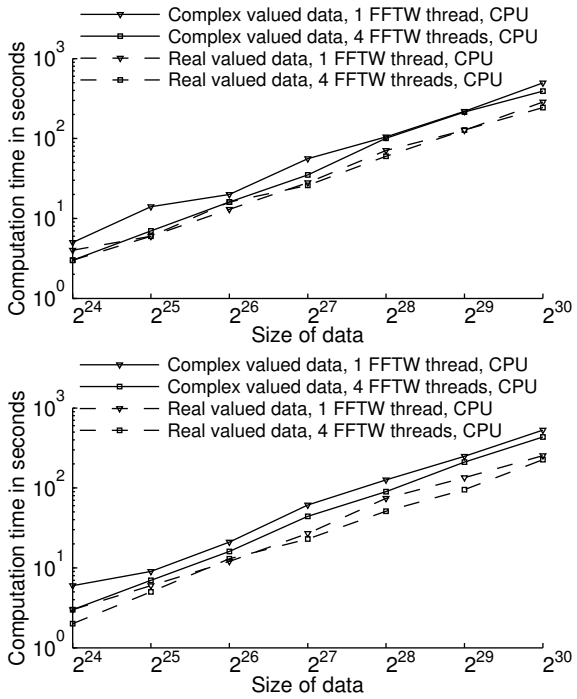

Figure 6: Runtimes for the forward (top) and inverse (bottom) 4D wave atom transform as the function of data volume.

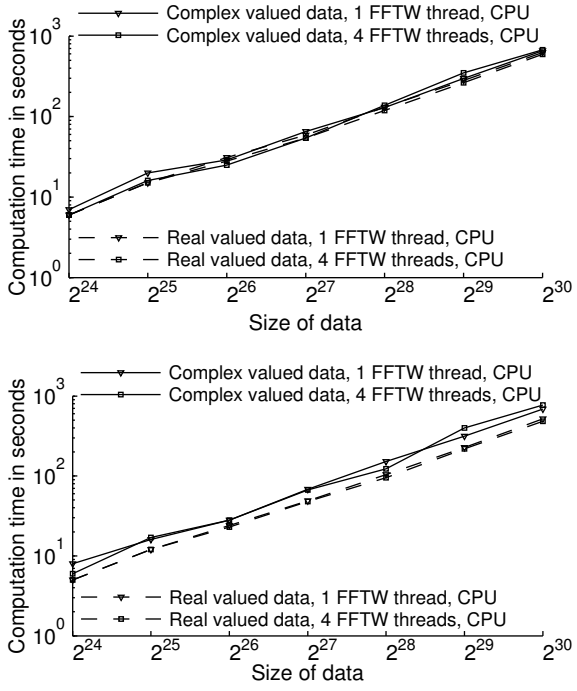

Figure 7: Runtimes for the forward (top) and inverse (bottom) 5D wave atom transform as the function of data volume.

REFERENCES

Averbuch, A., F. Meyer, J.-O. Stromberg, R. Coifman, and A. Vassiliou, 2001, Low bit-rate efficient compression for seismic data: Image Processing, IEEE Transactions on, **10**, 1801–1814.

Candés, E., L. Demanet, D. Donoho, and L. Ying, 2006, Fast discrete curvelet transforms: SIAM Mult. Model. Sim., **5**.

Candés, E., and D. Donoho, 2004, New tight frames of curvelets and optimal representations of objects with piecewise-c^2 singularities: Comm. Pure Appl. Math., **57**.

Demanet, L., 2006, Curvelets, wave atoms and wave equations: PhD thesis, California Institute of Technology.

Demanet, L., and L. Ying, 2007a, Curvelets and wave atoms for mirror-extended images: Optical Engineering+ Applications, International Society for Optics and Photonics, 67010J–67010J.

———, 2007b, Wave atoms and sparsity of oscillatory patterns: Appl. Comput. Harm. Anal., **23**.

———, 2009, Wave atoms and time upscaling of wave equations: Numer. Math., **113**.

Duval, L., and T. Q. Nguyen, 1999, Seismic data compression: a comparative study between GenLot and wavelet compression: Proc. SPIE, Wavelets: Appl. Signal Image Process., SPIE, 802–810.

Fomel, S., 2006, 572, *in* Towards the seislet transform: 2847–2851.

Frigo, M., and S. G. Johnson, 2005, The design and implementation of fftw3: Proceedings of the IEEE, **93**, 216–231.

G. Martin, R. W., and K. Marfurt, 2006, An elastic upgrade for Marmousi: The Leading Edge, Society for Exploration Geophysics, **25**.

Geng, Y., R. Wu, and J. Gao, 2009, 730, *in* Dreamlet transform applied to seismic data compression and its effects on migration: 3640–3644.

Hennenfent, G., and F. J. Herrmann, 2006, Seismic denoising with nonuniformly sampled curvelets: Computing in Science & Engineering, **8**, 16–25.

Herrmann, F. J., U. Böniger, and D. J. E. Verschuur, 2007, Non-linear primary-multiple separation with directional curvelet frames: Geophysical Journal International, **170**, 781–799.

Herrmann, F. J., P. Moghaddam, and C. C. Stolk, 2008, Sparsity-and continuity-promoting seismic image recovery with curvelet frames: Applied and Computational Harmonic Analysis, **24**, 150–173.

Vassiliou, A. A., and M. V. Wickerhouser, 1997, Comparison of wavelet image coding schemes for seismic data compression: Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, 118–126.

Villasenor, J., R. A. Ergas, and P. L. Donoho, 1996, Seismic data compression using high-dimensional wavelet transforms: Data Compression Conference, 1996. DCC '96. Proceedings, 396–405.

Wang, S., J. Li, S. K. Chiu, and P. D. Anno, 2010, 714, *in* Seismic data compression and regularization via wave packets: 3650–3655.

Wu, W., Z. Yang, Q. Qin, and F. Hu, 2006, Adaptive seismic data compression using wavelet packets: Geoscience and Remote Sensing Symposium, 2006. IGARSS 2006. IEEE International Conference on, 787–789.

Ying, L., L. Demanet, and E. Candes, 2005, 3d discrete curvelet transform: Optics & Photonics 2005, International Society for Optics and Photonics, 591413–591413.