Fast Discrete Curvelet Transforms

Emmanuel Candès†, Laurent Demanet†, David Donoho♯ and Lexing Ying†

† Applied and Computational Mathematics, Caltech, Pasadena, CA 91125

♯ Department of Statistics, Stanford University, Stanford, CA 94305

July 2005, revised March 2006

Abstract

This paper describes two digital implementations of a new mathematical transform, namely, the second generation *curvelet transform* [12, 10] in two and three dimensions. The first digital transformation is based on unequally-spaced fast Fourier transforms (USFFT) while the second is based on the wrapping of specially selected Fourier samples. The two implementations essentially differ by the choice of spatial grid used to translate curvelets at each scale and angle. Both digital transformations return a table of digital curvelet coefficients indexed by a scale parameter, an orientation parameter, and a spatial location parameter. And both implementations are fast in the sense that they run in $O(n^2 \log n)$ flops for n by n Cartesian arrays; in addition, they are also invertible, with rapid inversion algorithms of about the same complexity.

Our digital transformations improve upon earlier implementations—based upon the first generation of curvelets—in the sense that they are conceptually simpler, faster and far less redundant. The software CurveLab, which implements both transforms presented in this paper, is available at http://www.curvelet.org.

Keywords. 2D and 3D Curvelet Transforms, Fast Fourier Transforms, Unequispaced Fast Fourier Transforms, Smooth Partitioning, Interpolation, Digital Shear, Filtering, Wrapping.

Acknowledgments. E. C. is partially supported by a National Science Foundation grant DMS 01-40698 (FRG) and by a Department of Energy grant DE-FG03-02ER25529. L. Y. is supported by a Department of Energy grant DE-FG03-02ER25529. We would like to thank Eric Verschuur and Felix Herrmann for providing seismic image data.

1 Introduction

1.1 Classical Multiscale Analysis

The last two decades have seen tremendous activity in the development of new mathematical and computational tools based on multiscale ideas. Today, multiscale/multiresolution ideas permeate many fields of contemporary science and technology. In the information sciences and especially signal processing, the development of wavelets and related ideas led to convenient tools to navigate through large datasets, to transmit compressed data rapidly, to remove noise from signals and images, and to identify crucial transient features in such datasets. In the field of scientific computing, wavelets and related multiscale methods sometimes allow for the speeding up of fundamental scientific computations such as in the numerical evaluation of the solution of partial differential equations [2]. By now, multiscale thinking is associated with an impressive and ever increasing list of success stories.

Despite considerable success, intense research in the last few years has shown that classical multiresolution ideas are far from being universally effective. Indeed, just as people recognized that Fourier methods were not good for all purposes—and consequently introduced new systems such as wavelets—researchers have sought alternatives to wavelet analysis. In signal processing for example, one has to deal with the fact that interesting phenomena occur along curves or sheets, e.g., edges in a two-dimensional image. While wavelets are certainly suitable for dealing with objects where the interesting phenomena, e.g., singularities, are associated with exceptional points, they are ill-suited for detecting, organizing, or providing a compact representation of intermediate dimensional structures. Given the significance of such intermediate dimensional phenomena, there has been a vigorous research effort to provide better adapted alternatives by combining ideas from geometry with ideas from traditional multiscale analysis [17, 19, 4, 31, 14, 16].

1.2 Why a Discrete Curvelet Transform?

A special member of this emerging family of multiscale geometric transforms is the *curvelet transform* [8, 12, 10] which was developed in the last few years in an attempt to overcome inherent limitations of traditional multiscale representations such as wavelets. Conceptually, the curvelet transform is a multiscale pyramid with many directions and positions at each length scale, and needle-shaped elements at fine scales. This pyramid is nonstandard, however. Indeed, curvelets have useful geometric features that set them apart from wavelets and the likes. For instance, curvelets obey a parabolic scaling relation which says that at scale 2^{-j}, each element has an envelope which is aligned along a "ridge" of length $2^{-j/2}$ and width 2^{-j}. We postpone the mathematical treatment of the curvelet transform to Section 2, and focus instead on the reasons why one might care about this new transformation and by extension, why it might be important to develop accurate discrete curvelet transforms.

Curvelets are interesting because they efficiently address very important problems where wavelet ideas are far from ideal. We give three examples:

1. *Optimally sparse representation of objects with edges.* Curvelets provide optimally sparse representations of objects which display *curve-punctuated smoothness*—smoothness except

for discontinuity along a general curve with bounded curvature. Such representations are nearly as sparse as if the object were not singular and turn out to be far more sparse than the wavelet decomposition of the object.

This phenomenon has immediate applications in approximation theory and in statistical estimation. In approximation theory, let f_m be the m-term curvelet approximation (corresponding to the m largest coefficients in the curvelet series) to an object $f(x_1, x_2) \in L^2(\mathbf{R}^2)$. Then the enhanced sparsity says that if the object f is singular along a generic smooth C^2 curve but otherwise smooth, the approximation error obeys

$$\|f - f_m\|_{L^2}^2 \le C \cdot (\log m)^3 \cdot m^{-2},$$

and is optimal in the sense that no other representation can yield a smaller asymptotic error with the same number of terms. The implication in statistics is that one can recover such objects from noisy data by simple curvelet shrinkage and obtain a Mean Squared Error (MSE) order of magnitude better than what is achieved by more traditional methods. In fact, the recovery is provably asymptotically near-optimal. The statistical optimality of the curvelet shrinkage extends to other situations involving indirect measurements as in a large class of ill-posed inverse problems [9].

2. *Optimally sparse representation of wave propagators.* Curvelets may also be a very significant tool for the analysis and the computation of partial differential equations. For example, a remarkable property is that curvelets faithfully model the geometry of wave propagation. Indeed, the action of the wave-group on a curvelet is well approximated by simply translating the center of the curvelet along the Hamiltonian flows. A physical interpretation of this result is that curvelets may be viewed as coherent waveforms with enough frequency localization so that they behave like waves but at the same time, with enough spatial localization so that they simultaneously behave like particles [5, 36].

   This can be rigorously quantified. Consider a symmetric system of linear hyperbolic differential equations of the form

$$\frac{\partial u}{\partial t} + \sum_k A_k(x) \frac{\partial u}{\partial x_k} + B(x) u = 0, \qquad u(0,x) = u_0(x), \tag{1.1}$$

   where u is an m-dimensional vector and $x \in \mathbf{R}^n$. The matrices A_k and B may smoothly depend on the spatial variable x, and the A_k are symmetric. Let E_t be the solution operator mapping the wavefield $u(0,x)$ at time zero into the wavefield $u(t,x)$ at time t. Suppose that (φ_n) is a (vector-valued) tight frame of curvelets. Then [5] shows that the curvelet matrix

$$E_t(n, n') = \langle \varphi_n, E_t \varphi_{n'} \rangle \tag{1.2}$$

   is sparse and well-organized. It is sparse in the sense that the matrix entries in an arbitrary row or column decay nearly exponentially fast (i.e., faster than any negative polynomial). And it is well-organized in the sense that the very few nonnegligible entries occur near a few shifted diagonals. Informally speaking, one can think of curvelets as near-eigenfunctions of the solution operator to a large class of hyperbolic differential equations.

   On the one hand, the enhanced sparsity simplifies mathematical analysis and allows to prove sharper inequalities. On the other hand, the enhanced sparsity of the solution operator in the curvelet domain allows the design of new numerical algorithms with far better asymptotic properties in terms of the number of computations required to achieve a given accuracy [6].

3. *Optimal image reconstruction in severely ill-posed problems.* Curvelets also have special microlocal features which make them especially adapted to certain reconstruction problems with missing data. For example, in many important medical applications, one wishes to reconstruct an object $f(x_1, x_2)$ from noisy and incomplete tomographic data [33], i.e., a subset of line integrals of f corrupted by additive moise modeling uncertainty in the measurements.

   Because of its relevance in biomedical imaging, this problem has been extensively studied (compare the vast literature on computed tomography). Yet, curvelets offer surprisingly new quantitative insights [11]. For example, a beautiful application of the phase-space localization of the curvelet transform allows a very precise description of those features of the object of f which can be reconstructed accurately from such data and how well, and of those features which cannot be recovered. Roughly speaking, the data acquisition geometry separates the curvelet expansion of the object into two pieces

$$f = \sum_{n \in \text{Good}} \langle f, \varphi_n \rangle \varphi_n + \sum_{n \notin \text{Good}} \langle f, \varphi_n \rangle \varphi_n.$$

   The first part of the expansion can be recovered accurately while the second part cannot. What is interesting here is that one can provably reconstruct the "recoverable" part with an accuracy similar to that one would achieve even if one had complete data. There is indeed a quantitative theory showing that for some statistical models which allow for discontinuities in the object to be recovered, there are simple algorithms based on the shrinkage of curvelet-biorthogonal decompositions, which achieve optimal statistical rates of convergence; that is, such that there are no other estimating procedure which, in an asymptotic sense, give fundamentally better MSEs [11].

To summarize, the curvelet transform is mathematically valid, and a very promising potential in traditional (and perhaps less traditional) application areas for wavelet-like ideas such as image processing, data analysis, and scientific computing clearly lies ahead. To realize this potential though, and deploy this technology to a wide range of problems, one would need a fast and accurate discrete curvelet transform operating on digital data. This is the object of this paper.

1.3 A New Discrete Curvelet Transform

Curvelets were first introduced in [8] and have been around for a little over five years by now. Soon after their introduction, researchers developed numerical algorithms for their implementation [37, 18], and scientists have started to report on a series of practical successes, see [39, 38, 27, 26, 20] for example. Now these implementations are based on the original construction [8] which uses a pre-processing step involving a special partitioning of phase-space followed by the ridgelet transform [4, 7] which is applied to blocks of data that are well localized in space and frequency.

In the last two or three years, however, curvelets have actually been redesigned in a effort to make them easier to use and understand. As a result, the new construction is considerably simpler and totally transparent. What is interesting here is that the new mathematical architecture suggests innovative algorithmic strategies, and provides the opportunity to improve upon earlier implementations. This paper develops two new fast discrete curvelet transforms (FDCTs) which are simpler, faster, and less redundant than existing proposals:

- Curvelets via USFFT, and
- Curvelets via Wrapping.

Both FDCTs run in $O(n^2 \log n)$ flops for n by n Cartesian arrays, and are also invertible, with rapid inversion algorithms of about the same complexity. To substantiate the pay-off, consider one of these FDCTs, namely, the FDCT via wrapping: first and unlike earlier discrete transforms, this implementation is a numerical isometry; second, its effective computational complexity is 6 to 10 times that of an FFT operating on an array of the same size, making it ideal for deployment in large scale scientific applications.

1.4 Organization of the Paper

The paper is organized as follows. We begin in Section 2 by rehearsing the main features of the curvelet transform for continuous-time objects with an emphasis on its mathematical architecture. Section 3 introduces the main ideas underlying the USFFT-based and the wrapping-based digital implementations which are then detailed in Sections 4 and 6 respectively. We address the problem of computing Fourier transforms on irregular grids in Section 5. Section 7 discusses refinements and extensions of the ideas underlying the discrete transforms while Section 8 illustrates our methods with a few numerical experiments. Finally, we conclude with Section 9 which introduces open problems, explains connections with the work of others, and outlines possible applications of these transforms.

1.5 CurveLab

The software package CurveLab implements the transforms proposed in this paper, and is available at http://www.curvelet.org. It contains the Matlab and C++ implementations of both the USFFT-based and the wrapping-based transforms. Several Matlab scripts are provided to demonstrate how to use this software. Additionally, three different implementations of the 3D discrete curvelet transform are also included.

2 Continuous-Time Curvelet Transforms

We work throughout in two dimensions, i.e., $\mathbf{R}^2$, with spatial variable x, with ω a frequency-domain variable, and with r and θ polar coordinates in the frequency-domain. We start with a pair of windows $W(r)$ and $V(t)$, which we will call the "radial window" and "angular window," respectively. These are both smooth, nonnegative and real-valued, with W taking positive real arguments and supported on $r \in (1/2, 2)$ and V taking real arguments and supported on $t \in [-1, 1]$.

These windows will always obey the admissibility conditions:

$$\sum_{j=-\infty}^{\infty} W^2(2^j r) = 1, \qquad r \in (3/4, 3/2); \tag{2.1}$$

$$\sum_{\ell=-\infty}^{\infty} V^2(t - \ell) = 1, \qquad t \in (-1/2, 1/2). \tag{2.2}$$

Now, for each $j \geq j_0$, we introduce the frequency window U_j defined in the Fourier domain by

$$U_j(r, \theta) = 2^{-3j/4} W(2^{-j} r)\, V(\frac{2^{\lfloor j/2 \rfloor}\theta}{2\pi}). \tag{2.3}$$

where $\lfloor j/2 \rfloor$ is the integer part of $j/2$. Thus the support of U_j is a polar “wedge” defined by the support of W and V, the radial and angular windows, applied with scale-dependent window widths in each direction. To obtain real-valued curvelets, we work with the symmetrized version of (2.3), namely, $U_j(r, \theta) + U_j(r, \theta + \pi)$.

Define the waveform $\varphi_j(x)$ by means of its Fourier transform $\hat{\varphi}_j(\omega) = U_j(\omega)$ (we abuse notations slightly here by letting $U_j(\omega_1, \omega_2)$ be the window defined in the polar coordinate system by (2.3)). We may think of φ_j as a “mother” curvelet in the sense that all curvelets at scale 2^{-j} are obtained by rotations and translations of φ_j. Introduce

- the equispaced sequence of *rotation angles* $\theta_\ell = 2\pi \cdot 2^{-\lfloor j/2 \rfloor} \cdot \ell$, with $\ell = 0, 1, \ldots$ such that $0 \leq \theta_\ell < 2\pi$ (note that the spacing between consecutive angles is scale-dependent),
- and the sequence of *translation parameters* $k = (k_1, k_2) \in \mathbf{Z}^2$.

With these notations, we define curvelets (as function of $x = (x_1, x_2)$) at scale 2^{-j}, orientation θ_ℓ and position $x_k^{(j,\ell)} = R_{\theta_\ell}^{-1}(k_1 \cdot 2^{-j}, k_2 \cdot 2^{-j/2})$ by

$$\varphi_{j,\ell,k}(x) = \varphi_j\left(R_{\theta_\ell}(x - x_k^{(j,\ell)})\right),$$

where R_θ is the rotation by θ radians and R_θ^{-1} its inverse (also its transpose),

$$R_\theta = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}, \qquad R_\theta^{-1} = R_\theta^T = R_{-\theta}.$$

A curvelet coefficient is then simply the inner product between an element $f \in L^2(\mathbf{R}^2)$ and a curvelet $\varphi_{j,\ell,k}$,

$$c(j, \ell, k) := \langle f, \varphi_{j,\ell,k} \rangle = \int_{\mathbf{R}^2} f(x) \overline{\varphi_{j,\ell,k}(x)}\, dx. \tag{2.4}$$

Since digital curvelet transforms operate in the frequency domain, it will prove useful to apply Plancherel’s theorem and express this inner product as the integral over the frequency plane

$$c(j, \ell, k) := \frac{1}{(2\pi)^2} \int \hat{f}(\omega)\, \overline{\hat{\varphi}_{j,\ell,k}(\omega)}\, d\omega = \frac{1}{(2\pi)^2} \int \hat{f}(\omega)\, U_j(R_{\theta_\ell}\omega) e^{i\langle x_k^{(j,\ell)}, \omega\rangle}\, d\omega. \tag{2.5}$$

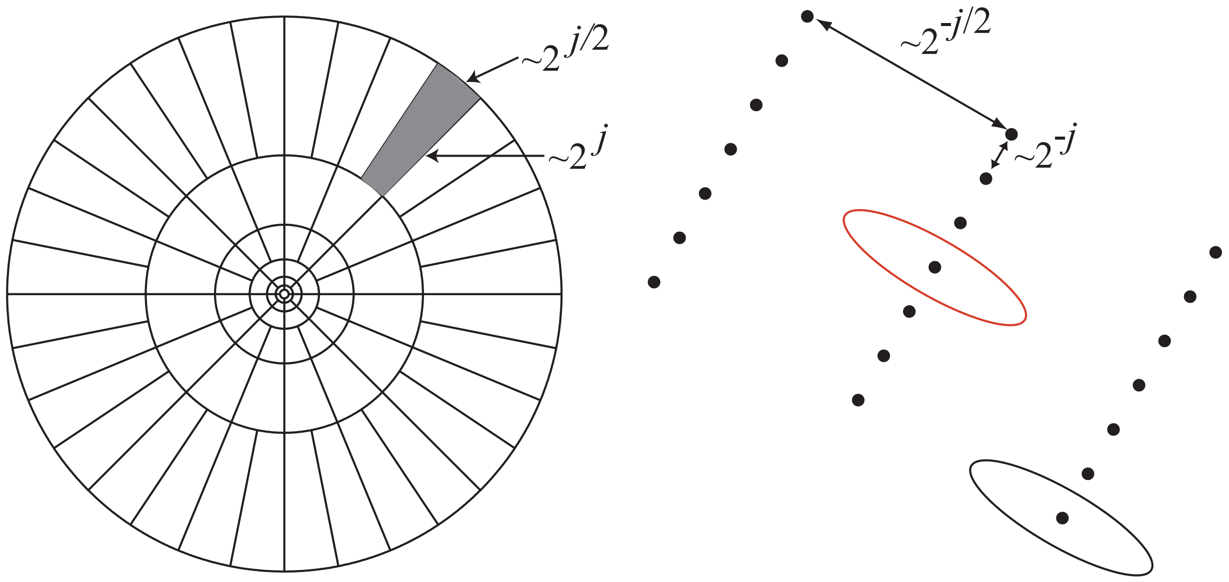


Figure 1: Curvelet tiling of space and frequency. The figure on the left represents the induced tiling of the frequency plane. In Fourier space, curvelets are supported near a "parabolic" wedge, and the shaded area represents such a generic wedge. The figure on the right schematically represents the spatial Cartesian grid associated with a given scale and orientation.

As in wavelet theory, we also have coarse scale elements. We introduce the low-pass window W_0 obeying

$$|W_0(r)|^2 + \sum_{j \geq 0} |W(2^{-j}r)|^2 = 1,$$

and for $k_1, k_2 \in \mathbf{Z}$, define coarse scale curvelets as

$$\varphi_{j_0,k}(x) = \varphi_{j_0}(x - 2^{-j_0}k), \quad \hat{\varphi}_{j_0}(\omega) = 2^{-j_0} W_0(2^{-j_0}|\omega|).$$

Hence, coarse scale curvelets are nondirectional. The "full" curvelet transform consists of the fine-scale directional elements $(\varphi_{j,\ell,k})_{j \geq j_0,\ell,k}$ and of the coarse-scale isotropic father wavelets $(\Phi_{j_0,k})_k$. It is the behavior of the fine-scale directional elements that are of interest here. Figure 1 summarizes the key components of the construction.

We now list a few properties of the curvelet transform.

1. **Tight frame**. Much like in an orthonormal basis, we can easily expand an arbitrary function $f(x_1, x_2) \in L^2(\mathbf{R}^2)$ as a series of curvelets: we have a reconstruction formula

$$f = \sum_{j,\ell,k} \langle f, \varphi_{j,\ell,k} \rangle \varphi_{j,\ell,k}, \tag{2.6}$$

with equality holding in an L^2 sense; and a Parseval relation

$$\sum_{j,\ell,k} |\langle f, \varphi_{j,\ell,k} \rangle|^2 = \|f\|^2_{L^2(\mathbf{R}^2)}, \quad \forall f \in L^2(\mathbf{R}^2). \tag{2.7}$$

(In both (2.6) and (2.7), the summation extends to the coarse scale elements.)

2. **Parabolic scaling**. The frequency localization of φ_j implies the following spatial structure: $\varphi_j(x)$ is of rapid decay away from a 2^{-j} by $2^{-j/2}$ rectangle with major axis pointing in the vertical direction. In short, the effective length and width obey the anisotropy scaling relation
$$\textit{length} \approx 2^{-j/2}, \quad \textit{width} \approx 2^{-j} \quad \Rightarrow \quad \textit{width} \approx \textit{length}^2. \tag{2.8}$$

3. **Oscillatory behavior**. As is apparent from its definition, $\hat{\varphi}_j$ is actually supported away from the vertical axis $\omega_1 = 0$ but near the horizontal $\omega_2 = 0$ axis. In a nutshell, this says that $\varphi_j(x)$ is oscillatory in the x_1-direction and lowpass in the x_2-direction. Hence, at scale 2^{-j}, a curvelet is a little needle whose envelope is a specified "ridge" of effective length $2^{-j/2}$ and width 2^{-j}, and which displays an oscillatory behavior across the main "ridge".

4. **Vanishing moments**. The curvelet template φ_j is said to have q vanishing moments when
$$\int_{-\infty}^{\infty} \varphi_j(x_1, x_2) x_1^n \, dx_1 = 0, \qquad \text{for all } 0 \le n < q, \text{ for all } x_2. \tag{2.9}$$
The same property of course holds for rotated curvelets when x_1 and x_2 are taken to be the corresponding rotated coordinates. Notice that the integral is taken in the direction perpendicular to the ridge, so counting vanishing moments is a way to quantify the oscillation property mentioned above. In the Fourier domain, (2.9) becomes a line of zeros with some multiplicity:
$$\frac{\partial^n \hat{\varphi}_j}{\partial \omega_1^n}(0, \omega_2) = 0, \qquad \text{for all } 0 \le n < q, \text{ for all } \omega_2.$$
Curvelets as defined and implemented in this paper have an infinite number of vanishing moments because they are compactly supported well away from the origin in the frequency plane, as illustrated in Figures 1 and 2.

3 Digital Curvelet Transforms

In this paper, we propose two distinct implementations of the curvelet transform which are faithful to the mathematical transformation outlined in the previous section. These digital transformations are linear and take as input Cartesian arrays of the form $f[t_1, t_2]$, $0 \le t_1, t_2 < n$, which allows us to think of the output as a collection of coefficients $c^D(j, \ell, k)$ obtained by the digital analog to (2.4)
$$c^D(j, \ell, k) := \sum_{0 \le t_1, t_2 < n} f[t_1, t_2] \, \overline{\varphi^D_{j,\ell,k}[t_1, t_2]}, \tag{3.1}$$
where each $\varphi^D_{j,\ell,k}$ is a digital curvelet waveform (here and below, the superscript D stands for "digital"). As is standard in scientific computations, we will actually never build these digital waveforms which are implicitly defined by the algorithms; formally, they are the rows of the matrix representing the linear transformation and are also known as Riesz representers. We merely introduce these waveforms because it will make the exposition clearer and because it provides a useful way to explain the relationship with the continuous-time transformation. The two digital transformations share a common architecture which we introduce first, before elaborating on the main differences.

3.1 Digital Coronization

In the continuous-time definition (2.3), the window U_j smoothly extracts frequencies near the dyadic corona $\{2^j \le r \le 2^{j+1}\}$ and near the angle $\{-\pi \cdot 2^{-j/2} \le \theta \le \pi \cdot 2^{-j/2}\}$. Coronae and rotations are not especially adapted to Cartesian arrays. Instead, it is convenient to replace these concepts by Cartesian equivalents; here, "Cartesian coronae" based on concentric squares (instead of circles) and shears. For example, the Cartesian analog to the family $(W_j)_{j\ge 0}$, $W_j(\omega) = W(2^{-j}\omega)$, would be a window of the form

$$\tilde{W}_j(\omega) = \sqrt{\Phi_{j+1}^2(\omega) - \Phi_j^2(\omega)}, \quad j \ge 0,$$

where Φ is defined as the product of low-pass one dimensional windows

$$\Phi_j(\omega_1, \omega_2) = \phi(2^{-j}\omega_1)\, \phi(2^{-j}\omega_2).$$

The function ϕ obeys $0 \le \phi \le 1$, might be equal to 1 on $[-1/2, 1/2]$, and vanishes outside of $[-2, 2]$. It is immediate to check that

$$\Phi_0(\omega)^2 + \sum_{j \ge 0} \tilde{W}_j^2(\omega) = 1. \tag{3.2}$$

We have just seen how to separate scales in a Cartesian-friendly fashion and now examine the angular localization. Suppose that V is as before, i.e., obeys (2.2) and set

$$V_j(\omega) = V(2^{\lfloor j/2 \rfloor}\omega_2/\omega_1).$$

We can then use $\tilde{W}_j$ and V_j to define the "Cartesian" window

$$\tilde{U}_j(\omega) := \tilde{W}_j(\omega) V_j(\omega). \tag{3.3}$$

It is clear that $\tilde{U}_j$ isolates frequencies near the wedge $\{(\omega_1, \omega_2) : 2^j \le \omega_1 \le 2^{j+1},\ -2^{-j/2} \le \omega_2/\omega_1 \le 2^{-j/2}\}$, and is a Cartesian equivalent to the "polar" window of Section 2. Introduce now the set of equispaced slopes $\tan \theta_\ell := \ell \cdot 2^{-\lfloor j/2 \rfloor}$, $\ell = -2^{\lfloor j/2 \rfloor}, \ldots, 2^{\lfloor j/2 \rfloor} - 1$, and define

$$\tilde{U}_{j,\ell}(\omega) := W_j(\omega) V_j(S_{\theta_\ell}\, \omega),$$

where S_θ is the shear matrix,

$$S_\theta := \begin{pmatrix} 1 & 0 \\ -\tan\theta & 1 \end{pmatrix}.$$

The angles θ_ℓ are not equispaced here but the slopes are. When completed by symmetry around the origin and rotation by $\pm\pi/2$ radians, the $\tilde{U}_{j,\ell}$ define the Cartesian analog to the family $U_j(R_{\theta_\ell}\omega)$ of Section 2. The family $\tilde{U}_{j,\ell}$ implies a concentric tiling whose geometry is pictured in Figure 2.[1]

[1]There are other ways of defining such localizing windows. An alternative might be to select $\tilde{U}_j$ as

$$\tilde{U}_j(\omega) := \psi_j(\omega_1) V_j(\omega), \tag{3.4}$$

where $\psi_j(\omega_1) = \psi(2^{-j}\omega_1)$ with $\psi(\omega_1) = \sqrt{\phi(\omega_1/2)^2 - \phi(\omega_1)^2}$ a bandpass profile, and to define for each $\theta_\ell \in [-\pi/4, \pi/4)$

$$\tilde{U}_{j,\ell}(\omega) := \psi_j(\omega_1) V_j(S_{\theta_\ell}\, \omega) = \tilde{U}_j(S_{\theta_\ell}\, \omega).$$

With this special definition, the windows are shear-invariant at any given scale. In practice, both these choices are almost equivalent since for a large number of angles of interest, many ϕ would actually give identical windows $\tilde{U}_{j,\ell}$.

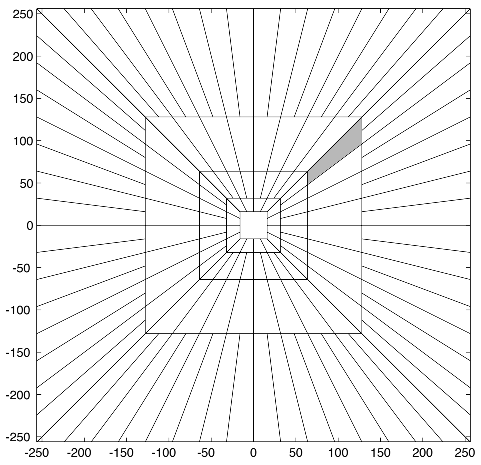


Figure 2: The figure illustrates the basic digital tiling. The windows $\tilde{U}_{j,\ell}$ smoothly localize the Fourier transform near the sheared wedges obeying the parabolic scaling. The shaded region represents one such typical wedge.

By construction, $V_j(S_{\theta_\ell}\,\omega) = V(2^{\lfloor j/2 \rfloor}\omega_2/\omega_1 - \ell)$ and for each $\omega = (\omega_1, \omega_2)$ with $\omega_1 > 0$, say, (2.2) gives

$$\sum_{\ell=-\infty}^{\infty} |V_j(S_{\theta_\ell}\,\omega)|^2 = 1.$$

Because of the support constraint on the function V, the above sum restricted to the angles of interest, $-1 \le \tan\theta_\ell < 1$, obeys $\sum_{\text{all angles}} |V_j(S_{\theta_\ell}\,\omega)|^2 = 1$, for $\omega_2/\omega_1 \in [-1 + 2^{-\lfloor j/2 \rfloor}, 1 - 2^{-\lfloor j/2 \rfloor}]$. Therefore, it follows from (3.2) that

$$\sum_{\text{all scales}} \sum_{\text{all angles}} |\tilde{U}_{j,\ell}(\omega)|^2 = 1. \tag{3.5}$$

There is a way to define "corner" windows specially adapted to junctions over the four quadrants (east, south, west, north) so that (3.5) holds for every $\omega \in \mathbf{R}^2$. We postpone this technical issue to Section 7.2.

The pseudopolar tiling of the frequency plane with trapezoids, in Figure 2, is already well-established as a data-friendly alternative to the ideal polar tiling. It was perhaps first introduced in two articles that appeared as book chapters in the same book, *Beyond Wavelets*, Academic Press, 2003. The first construction is that of *contourlets* [15] and is based on a cascade of properly sheared directional filters. On the other hand, *ridgelet packets* [24] are defined directly in the frequency plane via interpolation onto a pseudopolar grid aligned with the trapezoids.

In the next two sections we explain in parallel the two versions of the transform, namely via USFFT and via Wrapping. In a nutshell, the two implementations differ in the way curvelets at a given scale and angle are translated with respect to each other. In the USFFT-based version the translation grid is tilted to be aligned with the orientation of the curvelet, yielding the most faithful discretization of the continuous definition. In the Wrapping version the grid is the same for every angle within each quadrant—-yet each curvelet is given the proper orientation. As a result, the wrapping-based transform may be simpler to understand and implement.

3.2 Digital Curvelet Transform via Unequispaced FFTs

In what follows, we choose to work with the windows as in (3.4) although one could easily adapt the discussion to the other type, namely, (3.3). The digital coronization suggests Cartesian curvelets of the form $\tilde{\varphi}_{j,\ell,k}(x) = 2^{3j/4}\tilde{\varphi}_j(S^T_{\theta_\ell}(x - S^{-T}_{\theta_\ell} b))$ where b takes on the discrete values $b := (k_1 \cdot 2^{-j}, k_2 \cdot 2^{-j/2})$. The goal is to find a digital analog of the coefficients now given by

$$c(j,\ell,k) = \int \hat{f}(\omega)\tilde{U}_j(S^{-1}_{\theta_\ell}\omega)e^{i\langle S^{-T}_{\theta_\ell} b,\omega\rangle}\, d\omega. \tag{3.6}$$

Suppose for simplicity that $\theta_\ell = 0$. To numerically evaluate (3.6) with discrete data, one would just (1) take the 2D FFT of the object f and obtain $\hat{f}$, (2) multiply $\hat{f}$ with the window $\tilde{U}_j$, and (3) take the inverse Fourier transform on the appropriate Cartesian grid $b = (k_1 \cdot 2^{-j}, k_2 \cdot 2^{-j/2})$. The difficulty here is that for $\theta_\ell \neq 0$, we are asked to evaluate the inverse discrete Fourier transform (DFT) on the nonstandard sheared grid $S^{-T}_{\theta_\ell}(k_1 \cdot 2^{-j}, k_2 \cdot 2^{-j/2})$ and unfortunately, the classical FFT algorithm does not apply. To recover the convenient rectangular grid, however, one can pass the shearing operation to $\hat{f}$ and rewrite (3.6) as

$$c(j,\ell,k) = \int \hat{f}(\omega)\tilde{U}_j(S^{-1}_{\theta_\ell}\omega)e^{i\langle b, S^{-1}_{\theta_\ell}\omega\rangle}\, d\omega = \int \hat{f}(S_{\theta_\ell}\omega)\tilde{U}_j(\omega)e^{i\langle b,\omega\rangle}\, d\omega. \tag{3.7}$$

Suppose now that we are given a Cartesian array $f[t_1,t_2]$, $0 \le t_1, t_2 < n$ and let $\hat{f}[n_1,n_2]$ denote its 2D discrete Fourier transform

$$\hat{f}[n_1,n_2] = \sum_{t_1,t_2=0}^{n-1} f[t_1,t_2]e^{-i2\pi(n_1t_1+n_2t_2)/n}, \qquad -n/2 \le n_1, n_2 < n/2.$$

which here and below, we shall view as samples[2]

$$\hat{f}[n_1,n_2] = \hat{f}(2\pi n_1, 2\pi n_2)$$

from the interpolating trigonometric polynomial, also denoted $\hat{f}$, and defined by

$$\hat{f}(\omega_1,\omega_2) = \sum_{0\le t_1,t_2<n} f[t_1,t_2]e^{-i(\omega_1t_1+\omega_2t_2)/n}. \tag{3.8}$$

Assume next that $\tilde{U}_j[n_1,n_2]$ is supported on some rectangle of length $L_{1,j}$ and width $L_{2,j}$

$$\mathcal{P}_j = \{(n_1,n_2) : n_{1,0} \le n_1 < n_{1,0} + L_{1,j},\ n_{2,0} \le n_2 < n_{2,0} + L_{2,j}\}, \tag{3.9}$$

[2] Notice the notational difference between brackets $[\cdot,\cdot]$ for array indices, and parentheses $(\cdot,\cdot)$ for function evaluations, which holds throughout this paper. Non-integer arguments n_1, n_2 in $[n_1,n_2]$ are allowed and point to the fact that some interpolation is necessary.

(where $(n_{1,0}, n_{2,0})$ is the index of the pixel at the bottom-left of the rectangle.) Because of the parabolic scaling, $L_{1,j}$ is about 2^j and $L_{2,j}$ is about $2^{j/2}$. With these notations, the FDCT via USFFT simply evaluates

$$c^D(j, \ell, k) = \sum_{n_1, n_2 \in \mathcal{P}_j} \hat{f}[n_1, n_2 - n_1 \tan\theta_\ell] \, \tilde{U}_j[n_1, n_2] \, e^{i2\pi(k_1 n_1/L_{1,j} + k_2 n_2/L_{2,j})}, \tag{3.10}$$

$(\hat{f}[n_1, n_2 - n_1 \tan\theta_\ell] = \hat{f}(2\pi n_1, 2\pi(n_2 - n_1 \tan\theta_\ell)))$ and is therefore faithful to the original mathematical transformation.

This point of view suggests a first implementation we shall refer to as the FDCT via USFFT, and whose architecture is then roughly as follows:

1. Apply the 2D FFT and obtain Fourier samples $\hat{f}[n_1, n_2]$, $-n/2 \le n_1, n_2 < n/2$.
2. For each scale/angle pair (j, ℓ), resample (or interpolate) $\hat{f}[n_1, n_2]$ to obtain sampled values $\hat{f}[n_1, n_2 - n_1 \tan\theta_\ell]$ for $(n_1, n_2) \in \mathcal{P}_j$.
3. Multiply the interpolated (or sheared) object $\hat{f}$ with the parabolic window $\tilde{U}_j$, effectively localizing $\hat{f}$ near the parallelogram with orientation θ_ℓ, and obtain
$$\tilde{f}_{j,\ell}[n_1, n_2] = \hat{f}[n_1, n_2 - n_1 \tan\theta_\ell] \, \tilde{U}_j[n_1, n_2].$$
4. Apply the inverse 2D FFT to each $\tilde{f}_{j,\ell}$, hence collecting the discrete coefficients $c^D(j, \ell, k)$.

Of all the steps, the interpolation step is the less standard and is discussed in details in Section 4; we shall see that it is possible to design an algorithm which, for practical purposes, is exact and takes $O(n^2 \log n)$ flops for computation, and requires $O(n^2)$ storage, where n^2 is the number of pixels.

3.3 Digital Curvelet Transform via Wrapping

The "wrapping" approach assumes the same digital coronization as in Section 3.1, but makes a different, somewhat simpler choice of spatial grid to translate curvelets at each scale and angle. Instead of a tilted grid, we assume a regular rectangular grid and define "Cartesian" curvelets in essentially the same way as before,

$$c(j, \ell, k) = \int \hat{f}(\omega) \tilde{U}_j(S_{\theta_\ell}^{-1} \omega) e^{i\langle b, \omega \rangle} \, d\omega. \tag{3.11}$$

Notice that the $S_{\theta_\ell}^{-T} b$ of formula (3.6) has been replaced by $b \simeq (k_1 2^{-j}, k_2 2^{-j/2})$, taking on values on a rectangular grid. As before, this formula for b is understood when $\theta \in (-\frac{\pi}{4}, \frac{\pi}{4})$ or $(\frac{3\pi}{4}, \frac{5\pi}{4})$, otherwise the roles of $L_{1,j}$ and $L_{2,j}$ are to be exchanged.

The difficulty behind this approach is that, in the frequency plane, the window $\tilde{U}_{j,\ell}[n_1, n_2]$ does not fit in a rectangle of size $\sim 2^j \times 2^{j/2}$, aligned with the axes, in which the 2D IFFT could be applied to compute (3.11). After discretization, the integral over ω becomes a sum over n_1, n_2 which would extend beyond the bounds allowed by the 2D IFFT. The resemblance of (3.11) with a standard 2D inverse FFT is in that respect only formal.

To understand why respecting rectangle sizes is a concern, we recall that $\tilde{U}_{j,\ell}$ is supported in the parallelepipedal region

$$\mathcal{P}_{j,\ell} = S_{\theta_\ell}\,\mathcal{P}_j.$$

For most values of the angular variable θ_ℓ, $\mathcal{P}_{j,\ell}$ is supported inside a rectangle $\mathcal{R}_{j,\ell}$ aligned with the axes, and with sidelengths both on the order of 2^j. One could in principle use the 2D inverse FFT on this larger rectangle instead. This is close in spirit to the discretization of the continuous directional wavelet transform proposed by Vandergheynst and Gobbers in [41]. This seems ideal, but there is an apparent downside to this approach: dramatic oversampling of the coefficients. In other words, whereas the previous approach showed that it was possible to design curvelets with anisotropic spatial spacing of about $n/2^j$ in one direction and $n/2^{j/2}$ in the other, this approach would seem to require a naive regular rectangular grid with sidelength about $n/2^j$ in both directions. In other words, one would need to compute on the order of 2^{2j} coefficients per scale and angle as opposed, to only about $2^{3j/2}$ in the USFFT-based implementation. By looking at fine scale curvelets such that $2^j \asymp n$, this approach would require $O(n^{2.5})$ storage versus $O(n^2)$ for the USFFT version.

It is possible, however, to downsample the naive grid, and obtain for each scale and angle a subgrid which has the same cardinality as that in use in the USFFT implementation. The idea is to periodize the frequency samples as we now explain.

As before, we let $\mathcal{P}_{j,\ell}$ be a parallelogram containing the support of the discrete localizing window $\tilde{U}_{j,\ell}[n_1, n_2]$. We suppose that at each scale j, there exist two constants $L_{1,j} \sim 2^j$ and $L_{2,j} \sim 2^{j/2}$ such that, for every orientation θ_ℓ, one can tile the two-dimensional plane with translates of $\mathcal{P}_{j,\ell}$ by multiples of $L_{1,j}$ in the horizontal direction and $L_{2,j}$ in the vertical direction. The corresponding periodization of the windowed data $d[n_1, n_2] = \tilde{U}_{j,\ell}[n_1, n_2]\hat{f}[n_1, n_2]$ reads

$$Wd[n_1, n_2] = \sum_{m_1 \in \mathbf{Z}} \sum_{m_2 \in Z} d[n_1 + m_1 L_{1,j}, n_2 + m_2 L_{2,j}]$$

The *wrapped* windowed data, around the origin, is then defined as the restriction of $Wd[n_1, n_2]$ to indices n_1, n_2 inside a rectangle with sides of length $L_{1,j} \times L_{2,j}$ near the origin:

$$0 \le n_1 < L_{1,j}, \qquad 0 \le n_2 < L_{2,j}.$$

Given indices (n_1, n_2) originally inside $\mathcal{P}_{j,\ell}$ (possibly much larger than $L_{1,j}$, $L_{2,j}$), the correspondence between the wrapped and the original indices is one-to-one. Hence, the wrapping transformation is a simple reindexing of the data. It is possible to express the wrapping of the array $d[n_1, n_2]$ around the origin even more simply by using the "modulo" function:

$$Wd[n_1 \bmod L_{1,j},\, n_2 \bmod L_{2,j}] = d[n_1, n_2], \tag{3.12}$$

with $(n_1, n_2) \in \mathcal{P}_{j,\ell}$. Intuitively, the modulo operation maps the original (n_1, n_2) into their new position near the origin.

For those angles in the range $\theta \in (\pi/4, 3\pi/4)$, the wrapping is similar, after exchanging the role of the coordinate axes. This is the situation shown in figure 3.

Equipped with this definition, the architecture of the FDCT via wrapping is as follows:

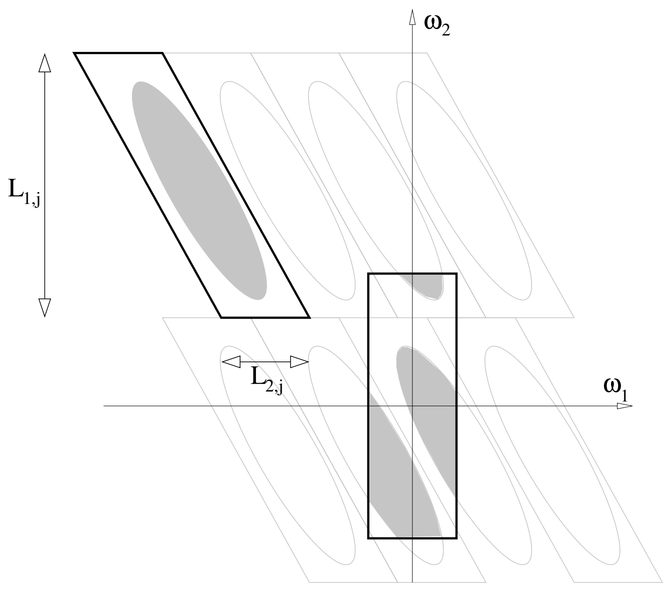


Figure 3: Wrapping data, intially inside a parallelogram, into a rectangle by periodicity. The angle θ is here in the range $(\pi/4, 3\pi/4)$. The black parallelogram is the tile $\mathcal{P}_{j,\ell}$ which contains the frequency support of the curvelet, whereas the gray parallelograms are the replicas resulting from periodization. The rectangle is centered at the origin. The wrapped ellipse appears "broken into pieces" but as we shall see, this is not an issue in the *periodic* rectangle, where the opposite edges are identified.

1. Apply the 2D FFT and obtain Fourier samples $\hat{f}[n_1, n_2]$, $-n/2 \leq n_1, n_2 < n/2$.
2. For each scale j and angle ℓ, form the product $\tilde{U}_{j,\ell}[n_1, n_2]\hat{f}[n_1, n_2]$.
3. Wrap this product around the origin and obtain
$$\tilde{f}_{j,\ell}[n_1, n_2] = W(\tilde{U}_{j,\ell}\hat{f})[n_1, n_2],$$
where the range for n_1 and n_2 is now $0 \leq n_1 < L_{1,j}$ and $0 \leq n_2 < L_{2,j}$ (for θ in the range $(-\pi/4, \pi/4)$).
4. Apply the inverse 2D FFT to each $\tilde{f}_{j,\ell}$, hence collecting the discrete coefficients $c^D(j, \ell, k)$.

It is clear that this algorithm has computational complexity $O(n^2 \log n)$ and in practice, its computational cost does not exceed that of 6 to 10 two-dimensional FFTs, see Section 8 for typical values of CPU times. In Section 6, we will detail some of the properties of this transform, namely, (1) it is an isometry, hence the inverse transform can simply be computed as the adjoint, and (2) it is faithful to the continuous transform.

3.4 FDCT Architecture

We finally close this section by listing those elements which are common to to both implementations

1. Frequency space is divided into dyadic annuli based on concentric squares.
2. Each annulus is subdivided into trapezoidal regions.
3. In the USFFT version, the discrete Fourier transform, viewed as a trigonometric polynomial, is sampled within each parallelepipedal region according an equispaced grid aligned with the axes of the parallelogram. Hence, there is a different sampling grid for each scale/orientation combination. The wrapping version, instead of interpolation, uses periodization to localize the Fourier samples in a rectangular region in which the IFFT can be applied. For a given scale, this corresponds only to two Cartesian sampling grids, one for all angles in the east-west quadrants, and one for the north-south quadrants.
4. Both forward transforms are specified in closed form, and are invertible (with inverse in closed form for the wrapping version).
5. The design of appropriate digital curvelets at the finest scale, or outermost dyadic corona, is not straightforward because of boundary/periodicity issues. Possible solutions at the finest scale are discussed in Section 7.
6. The transforms are cache-aware: all component steps involve processing n items in the array in sequence, e.g., there is frequent use of 1D FFTs to compute n intermediate results simultaneously.

They are other similarities such as similar running time complexities that shall be discussed in later sections.

4 FDCT via USFFTs

4.1 Interpolation

As explained earlier, we need to evaluate the DFT of $f[t_1, t_2]$ on the irregular grid $(n_1, n_2 - n_1 \tan\theta_\ell)$ where the parameters range as follows: $(n_1, n_2) \in \mathcal{P}_j$ and ℓ indexes all the angles $\theta_\ell \in (-\pi/4, \pi/4)$, say; Figure 4 shows the structure of this grid at a fixed scale and for orientations in the "east" quadrant. Fix n_1 or equivalently $\omega_1 = 2\pi n_1$, and consider the restriction g of the trigonometric polynomial F (3.8) to this (vertical) line; g is a 1-dimensional trigonometric polynomial of degree n which we express as

$$g(\omega) = \sum_{-n/2 \le u < n/2} c_u \, e^{-iu\omega/n}, \tag{4.1}$$

with $c_u = \sum_{t_1} f[t_1, u] e^{-i\omega_1 t_1/n}$. Now, (3.10) asks to evaluate g on the family of meshes (ω_m^ℓ)

$$\omega_m^\ell = 2\pi \cdot (m + n_1 \ \tan\theta_\ell), \quad m = -L_{2,j}/2, -L_{2,j}/2 + 1, \ldots, L_{2,j}/2 - 1$$

($L_{2,j}$ is the width of $\mathcal{P}_j$). For each ℓ, the mesh (ω_m^ℓ) with running point indexed by m is a regularly spaced grid, and there are as many meshes as discrete angles. This family of interleaved meshes is shown in Figure 5.

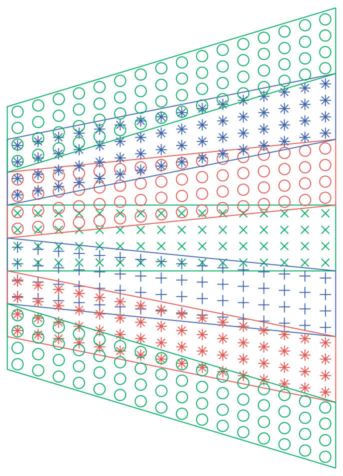


Figure 4: This figure illustrates the sampling within each parallelepipedal region according to an equispaced grid aligned with the axes of the parallelogram. There as many parallelograms as they are angles $\theta_\ell \in [-\pi/4, \pi/4)$.

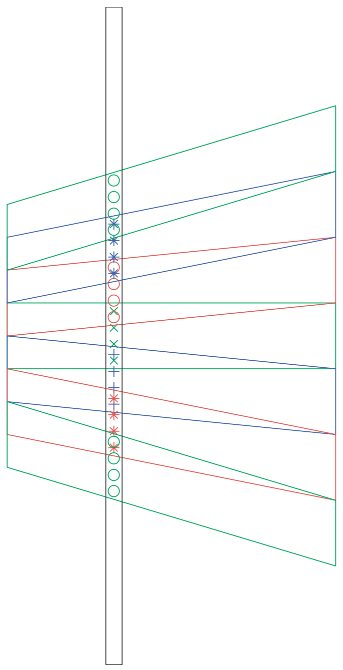


Figure 5: This figure illustrates a key property of the USFFT version. The interpolation step is organized so that it is obtained by solving a sequence of one-dimensional problems. For a fixed column, we need to resample a one-dimensional trigonometric polynomial on the mesh shown here.

The problem of evaluating the sum $g(\omega)$ (4.1) on the irregular grid is equivalent to that of resampling the polynomial g, which is known on the regular Nyquist grid $2\pi n_2$, $-n/2 \le n_2 < n/2$ by means of trigonometric interpolation

$$g(\omega) = \sum_{-n/2 \le n_2 < n/2} D\left(\frac{\omega - 2\pi n_2}{n}\right) g(2\pi n_2),$$

where D is the Dirichlet kernel

$$D(\omega) = \frac{\sin(n\omega/2)}{n\sin(\omega/2)}. \tag{4.2}$$

For each ℓ, it is well known that one can evaluate all the sampled values $g(\omega_m^\ell)$ using two 1D FFTs of length n. We omit the standard details.

We would like to emphasize that viewing $\hat{f}[n_1, n_2 - n_1 \tan\theta_\ell]$ as samples from the trigonometric polynomial (3.8) imposes trigonometric interpolation of the Fourier samples $\hat{f}[n_1, n_2]$. Naturally, one might employ other models which would lead to other interpolation schemes.

It is possible to compute (4.1) on the irregular grid by using as many one-dimensional FFTs as there are distinct angles. Since the curvelet pyramid exhibits about $\sqrt{n}$ orientations at fine scales, the complexity of column interpolation would be at most of the order $O(n^{3/2}\log n)$. Clearly, the interpolation step is computationally the most expensive component of the digital transform (see Section 3); because each column is only touched at most twice, the algorithm just described would take $O(n^{5/2}\log n)$ for exact computation for an image of size n by n. However, the algorithm can also be implemented in an approximate manner in $O(n^2 \log n)$ flops. For practical purposes, this approximation is exact.

The reason for the speed-up is that the fast approximate transform is applied using the 1-dimensional USFFT (unequally spaced fast Fourier transform). This step is organized so that many related sampling problems, i.e., problems for unrelated meshes, are done simultaneously. In effect, the USFFT rapidly computes *all* the irregularly spaced samples we need with high accuracy. We postpone the presentation of this algorithm to Section 5.

4.2 Riesz Representers and the Dual Grid

How do digital curvelets look like? To answer this question, let S_θ^D be the digital shear shifting each column of $\hat{f}$ as in (3.10), namely,

$$(S_\theta^D \hat{f})[n_1, n_2] = \hat{f}[n_1, n_2 - n_1 \tan\theta], \quad (n_1, n_2) \in \mathcal{P}_j.$$

Now define $\hat{\varphi}_{j,0,k}^D$ by

$$\hat{\varphi}_{j,0,k}^D[n_1, n_2] = \tilde{U}_j[n_1, n_2]\, e^{-i2\pi(k_1 n_1/L_{1,j} + k_2 n_2/L_{2,j})}. \tag{4.3}$$

With these notations, we have

$$c^D(j, \ell, k) = \langle S_{\theta_\ell}^D \hat{f}, \hat{\varphi}_{j,0,k}^D \rangle = \langle \hat{f}, (S_{\theta_\ell}^D)^* \hat{\varphi}_{j,0,k}^D \rangle.$$

In other words and for $\theta_\ell = 0$, $\hat{\varphi}^D_{j,0,k}$ is the frequency domain definition of our digital curvelets since $c^D(j,0,k) = \langle S^D_{\theta_\ell}\hat{f}, \hat{\varphi}^D_{j,0,k}\rangle$. In addition, for arbitrary angles, the discrete Fourier transform of a digital curvelet is given by the expression

$$\hat{\varphi}^D_{j,\ell,k} = (S^D_{\theta_\ell})^* \hat{\varphi}^D_{j,0,k},$$

and therefore $\hat{\varphi}^D_{j,\ell,k}$ is obtained from the reference $\hat{\varphi}^D_{j,0,k}$ by a digital shear. We elaborate on this point and argue that the digital shear nearly acts like an exact resampling operation since

$$\hat{\varphi}^D_{j,\ell,k}[n_1,n_2] \approx \hat{\varphi}^D_{j,0,k}[S^{-1}_{\theta_\ell}(n_1,n_2)] \tag{4.4}$$

where the shear operator is as before, and where $\approx$ means that both sides are equal to within high accuracy. This last relation says that at a given scale, curvelets at arbitrary angles are basically obtained by shearing corresponding horizontal and vertical elements.

To justify (4.4), recall that S^D_θ is a sequence of 1-dimensional trigonometric interpolation shifting each column by $\tau = n_1 \tan\theta$ (n_1 is fixed). For convenience, let L_τ be the one-dimensional shift operator acting on vectors of size n, $h = L_\tau f$, and represented by the convolution

$$h(t) = \sum_{-n/2 \le t' < n/2} D\left(\frac{2\pi}{n}(t - \tau - t')\right) f(t'),$$

where D is the Dirichlet kernel (4.2). The interpolation is of course exact on trigonometric exponentials, i.e., $(L_\tau f)(t) = f(t-\tau)$ for $f(t) = e^{i2\pi ut/n}$, $-n/2 \le u < n/2$. The same property applies to its adjoint since L^*_τ is the same operator—only shifting by $-\tau$, instead.

To see how the interpolation acts on $\hat{\varphi}^D_{j,0,k}$, we recall the definition of the basic window $\tilde{U}_j[n_1,n_2] = \psi_j(2\pi n_1)V_j(n_2/n_1)$ as in (3.4). For a fixed column n_1, we will argue that

$$L_\tau V_j(n_2/n_1) \approx V_j((n_2-\tau)/n_1). \tag{4.5}$$

To see why this is true, observe that for a fixed scale j and abscissa n_1, $V_j(n_2/n_1)$ are sampled values of the function $V_\alpha(t) = V(\alpha t)$ on the grid $n_2/n \in [-1/2, 1/2]$ with $\alpha = 2^{\lfloor j/2 \rfloor} n/n_1$. Now, one can approximate V_α by means of its Fourier series

$$V_\alpha(t) \approx \sum_{u=-n/2}^{n/2-1} \hat{V}_\alpha(u)\, e^{i2\pi ut},$$

where $\hat{V}_\alpha(u)$ are the Fourier coefficients of the continuous time function V_α. The near-equality derives from the fact that for α substantially smaller than n, V_α is a smooth window with many derivatives, and is consequently well approximated by its Fourier series. Now because L_τ is exact on complex exponentials,

$$(L_\tau V_j[n_1,\cdot])(n_2) \approx \sum_{u=-n/2}^{n/2-1} \hat{V}_\alpha(u) e^{i\frac{2\pi u(n_2-\tau)}{n}} \approx V_j((n_2-\tau)/n_1).$$

as claimed. Therefore, letting $\hat{\varphi}^D_{j,0,k}$ be the basic curvelet as in (4.3),

$$\hat{\varphi}^D_{j,0,k}[n_1,n_2] = \psi_j(2\pi n_1)\, V_j(n_2/n_1)\, e^{-i\frac{2\pi k_1 n_1}{L_{1,j}}}\, e^{-i\frac{2\pi k_2 n_2}{L_{2,j}}},$$

and assuming that $L_{2,j}$ divides n, we proved that for each column

$$\begin{aligned}(L_\tau \hat{\varphi}^D_{j,0,k}[n_1,\cdot])(n_2) &\approx \psi_j(2\pi n_1) V_j((n_2-\tau)/n_1)\, e^{-i\frac{2\pi k_1 n_1}{L_{1,j}}}\, e^{-i\frac{2\pi k_2 (n_2-\tau)}{L_{2,j}}} \\ &= \hat{\varphi}^D_{j,0,k}[n_1, n_2-\tau].\end{aligned}$$

In conclusion, $L^*_{n_1 \tan\theta} \hat{\varphi}^D_{j,0,k}[n_1,n_2] \approx \tilde{\varphi}_{jk}[n_1, n_2 + n_1\tan\theta]$; that is (4.4).

We have just seen that we were entitled to think about curvelets at scale 2^{-j} and orientation θ_ℓ as elements of the form

$$\hat{\varphi}^D_{j,\ell,k}[n] \approx \tilde{U}_j[S^{-1}_{\theta_\ell} n] e^{i\langle S^{-T}_{\theta_\ell} b^D, n\rangle}, \quad b^D = (2\pi k_1/L_{1,j}, 2\pi k_2/L_{2,j}).$$

Let $\varphi^D_j[t_1,t_2]$, $-n/2 \le t_1, t_2 < n/2$, be the inverse discrete Fourier transform of $\tilde{U}_j[n_1,n_2]$. Then

$$\varphi^D_{j,\ell,k}[t] \approx \varphi^D_j[S^T_{\theta_\ell}(t - S^{-T}_{\theta_\ell} b^D)].$$

In other words, all the digital curvelets sharing that orientation and scale have support tiling the space according to a dual tilted lattice. In summary, at a given scale, all digital curvelets are essentially obtained by shearing and translating *a single reference element.*

4.3 The Adjoint Transformation

Each step of the curvelet transform via USFFT has an evident adjoint, and the overall adjoint transformation is computed by taking the adjoint of each step and applying them in reverse order.

1. For each pair (j,ℓ), apply the 2D FFT to the array $c^D(j,\ell;k)$ (j and ℓ are fixed) and obtain Fourier samples $\tilde{g}_{j,\ell}[n_1,n_2]$, $n_1,n_2 \in \mathcal{P}_j$.
2. For each pair (j,ℓ), form the product $\tilde{g}_{j,\ell}[n_1,n_2]\tilde{U}_j[n_1,n_2]$.
3. For each pair (j,ℓ), view the product $\tilde{g}_{j,\ell}[n_1,n_2]\tilde{U}_j[n_1,n_2]$ as samples on the sheared grid $(n_1, n_2 - n_1\tan\theta_\ell)$, and use trigonometric interpolation to resample this function on the standard Nyquist grid. Sum the contributions from all different scales and angles, and obtain $\hat{g}[n_1,n_2]$.
4. Apply the 2D IFFT and obtain the Cartesian array $g[t_1,t_2]$.

Clearly, the adjoint transformation shares all the basic properties of the forward transform. In particular, the cost of applying the adjoint is $O(n^2 \log n)$, with n^2 the number of pixels.

4.4 The Inverse Transformation

The transformation is invertible. Looking at the flow of the algorithm (Section 3), we see that the first and the last step are easily invertible by means of FFTs. We use conjugate gradients to invert the combination of step 2 and 3 (which in practice is applied scale by scale). Each CG iteration is implemented via a series of 1D processes which, thanks to the special structure of the Gram matrix, can be accelerated as we will see in the next section. In practice, 20 CG iterations (at each scale) give about 5 digit accuracy. The practical cost of this approximate inverse is about ten times that of the forward transform, see Section 8 for actual CPU times.

5 Unequispaced Fast Fourier Transforms

Suppose we are given a vector $(f[t])_{-n/2\le t<n/2}$ of size n, and a set of points (ω_k), $1 \le k \le m$. We wish to evaluate the Fourier transform of the vector f at each point ω_k

$$y[k] = F(\omega_k) = \sum_{t=-n/2}^{n/2} f[t]\, e^{-i\omega_k t}. \tag{5.1}$$

A closely related problem of interest as well is the evaluation of the adjoint transform which takes the form

$$g[t] = \sum_{k=1}^{m} y[k]\, e^{i\omega_k t}, \tag{5.2}$$

with t still in the range $t \in \{-n/2, -n/2+1, \dots, n/2-1\}$. For arbitrary nodes ω_k, direct evaluation of (5.1) takes $O(mn)$ operations which is often too much for practical purposes. For equispaced nodes on the Nyquist grid $\omega_k = 2\pi k/n$, the values can be computed via the FFT in $O(n \log n)$. However, in many applications, data are irregularly sampled or do not require sampling on an equispaced grid which seriously limits the applicability of the FFT. Application fields such as geophysics, geography, or astronomy all come to mind. As a consequence, it is critical to develop rapid and accurate algorithms that would evaluate sums such as (5.1). In the last decade or so, this problem received a large amount of attention.

Perhaps the most important references on this subject date back to the work of Dutt and Rokhlin [22] and Beylkin [3]. The basic idea is best explained when considering (5.2). First express the function $g(t)$ as the Fourier transform of the spike series

$$P(\omega) = \sum_{k=1}^{m} y[k]\, \delta(\omega - \omega_k).$$

The strategy is then to convolve $P(\omega)$ with a short filter $H(\omega)$ to make it approximately band-limited, sample the result on a regular grid and apply the FFT, and deconvolve the output to correct for the convolution with $H(\omega)$. This idea is further refined in [21] where the authors also report on error estimates.

5.1 The Algorithm

In this paper, we develop a different strategy for computing (5.1). Our approach is essentially the same as that of Anderson and Dahleh [1]. The idea is to compute intermediate Fourier samples on a finer grid and use Taylor approximations to compute approximate values of $F(\omega_k)$ at each node ω_k. The algorithm operates as follows:

1. Pad the vector f with zeros and create the vector $(f^D[t])$ of size Dn with index t obeying $-Dn/2 \le t < Dn/2$

$$f^D[t] = \begin{cases} f[t] & -n/2 \le t < n/2, \\ 0 & \text{otherwise.} \end{cases}$$

   This is schematically illustrated in Figure 6.

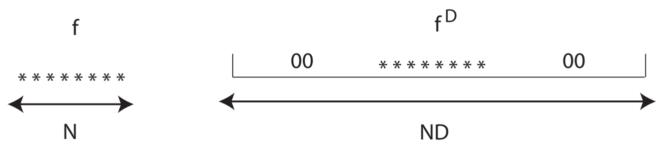


Figure 6: Zero-padding

2. Make L copies of f^D and multiply each copy by $(-it)^\ell$ obtaining
$$f^{D,\ell}[t] = (-it)^\ell f^D[t], \qquad \ell = 0, 1, \ldots, L-1.$$
3. Take the FFT of each $f^{D,\ell}$, and thereby obtain the values of F together with those of F^ℓ on the finer grid with spacing $2\pi/nD$, namely,
$$F^{(\ell)}\left(\frac{2\pi k}{nD}\right)$$
In short, the $(L-1)$-th order Taylor polynomial at each point on the finer grid is known.
4. Given an arbitrary point ω, evaluate an approximation to $F(\omega)$ by
$$F(\omega) \approx P(\omega_0) := F(\omega_0) + F'(\omega_0)(\omega - \omega_0) + \ldots F^{(L-1)}(\omega_0)\frac{(\omega-\omega_0)^{L-1}}{(L-1)!},$$
where ω_0 is the closest fine grid point to ω.

What is the cost of this algorithm? We need to compute L FFTs of length Dn followed by m evaluations of the Taylor polynomial. The complexity is therefore of $O(n \log n + m)$.

5.2 Error Analysis

What is the accuracy of this algorithm? Obviously, the error obeys
$$\|F(\omega) - P(\omega_0)\| \le \|F^L\|_\infty \cdot \frac{|\omega - \omega_0|^L}{L!}, \quad \|F^L\|_\infty = \sup_{[-\pi,\pi]} |F^{(L)}(\omega)|.$$
Now F is a trigonometric polynomial of degree n (with frequencies ranging from $-n/2$ to $n/2$) and obeys the Bernstein inequality [42] which states that
$$\|F^{(L)}\|_\infty \le (n/2)^L \|F\|_\infty.$$
Since by definition, the nearest point on the finer lattice obeys $|\omega - \omega_0| \le \pi/nD$, we have that for all $\omega \in [-\pi, \pi)$ the relative error is bounded by
$$\frac{|F(\omega) - P(\omega_0)|}{\|F\|_\infty} \le \left(\frac{\pi}{2D}\right)^L \cdot \frac{1}{L!}. \tag{5.3}$$
Table 1 below presents some numerical values of the upper bound in (5.3) for typical values of the oversampling factor D and of the number of derivatives. As one can see, we get quite a few number of digits of accuracy with relatively small values of both D and L; e.g., $L = 6$ and $D = 16$ guarantees 9 digits.

	$L = 4$	$L = 6$
$D = 8$	6.19(-5)	7.96(-8)
$D = 16$	3.87(-6)	1.24(-9)

Table 1: Numerical values for the relative error (5.3).

5.3 The Adjoint USFFT

Suppose now that we are interested in computing the adjoint transformation (5.2). A possible strategy is to take the adjoint of each step of the forward algorithm and apply them in reverse order. Equivalently, observe that

$$e^{i\omega t} = e^{i\omega_0 t} e^{i(\omega - \omega_0)t} \approx e^{i\omega_0 t} \sum_{\ell=0}^{L-1} \frac{[it(\omega - \omega_0)]^\ell}{\ell!},$$

where ω_0 is again the closest point to ω on the finer grid. This suggests the following strategy for computing (5.2):

1. For each point ω_0 on the finer lattice, compute
$$Z^\ell(\omega_0) = \sum_{\omega_k \in \mathcal{N}(\omega_0)} (\omega_k - \omega_0)^\ell y_k,$$
where $\omega_k \in \mathcal{N}(\omega_0)$ if and only if ω_0 is the nearest neighbor to ω_k.
2. Take the inverse Fourier transform of each vector Z^ℓ and obtain L vectors $(G^{D,\ell}[t])$ with $-Dn/2 \le t < Dn/2$.
3. Evaluate
$$G^D[t] = \sum_{\ell=0}^{L-1} \frac{(it)^\ell}{\ell!} G^{D,\ell}[t].$$
4. Finally, extract g on the subdomain of interest, namely, $-n/2 \le t < n/2$.

Clearly, the complexity and error estimates developed for the forward algorithm apply here as well.

5.4 The Gram Matrix

The inverse mapping from an equispaced sampling to an unequally spaced sampling does not have an analytical inverse, and one could think about applying preconditioned Conjugate Gradients or other iterative algorithms to go in the other direction. Let A be the unequally spaced Fourier transform (5.1) and A^* its adjoint (5.2). Many iterative algorithms—e.g., Conjugate Gradients—would actually require applying A^*A to an arbitrary vector a repeated number of times. As is well known, the linear transformation A^*A exhibits a Toeplitz structure which is here particularly useful. Set $g = A^*Af$ or

$$g[t] = \sum_{t'=-n/2}^{n/2-1} \sum_{k=1}^{m} e^{i\omega_k(t-t')} f[t'] = \sum_{t'=-n/2}^{n/2-1} c[t-t'] f[t'], \tag{5.4}$$

where

$$c[u] = \sum_{k=1}^{m} e^{i\omega_k u}, \quad \text{i.e., } c = A^*(1, 1, \ldots, 1).$$

The advantage is that we can apply a Toeplitz matrix to a vector of length n using essentially 2 FFTs of length $2n$. The idea is to embed the Toeplitz in a larger circulant matrix of size $2n-1$ which can then be applied efficiently by means of the FFT [40, 13].

6 FDCT via Frequency Wrapping

6.1 Riesz Representers

The naive technique suggested in Section 3 to obtain oversampled curvelet coefficients consists of a simple 2D inverse FFT, which reads

$$c^{D,O}(j,\ell,k) = \frac{1}{n^2} \sum_{n_1,n_2 \in \mathcal{R}_{j,\ell}} \hat{f}[n_1,n_2] \tilde{U}_{j,\ell}[n_1,n_2] e^{2\pi i (k_1 n_1/R_{1,j} + k_2 n_2/R_{2,j})}. \tag{6.1}$$

The superscripts D, O stand for Digital, Oversampled. As before, $\mathcal{R}_{j,\ell}$ is a rectangle of size $R_{1,j} \times R_{2,j}$, aligned with the Cartesian axes, and containing the parallelogram $\mathcal{P}_{j,\ell}$. Assume that $R_{1,j}$, $R_{2,j}$ divide the image size n. Then it is not hard to see that the coefficients $c^{D,O}(j,\ell,k)$ come from the discrete convolution of a curvelet with the signal $f(t_1,t_2)$, downsampled regularly in the sense that one selects only one out of every $n/R_{1,j} \times n/R_{2,j}$ pixel.

In general the dimensions $R_{1,j}$, $R_{2,j}$ of the rectangle are too large, as explained earlier. Equivalently, one wishes to downsample the convolution further. The idea of the wrapping approach is to replace $R_{1,j}$ and $R_{2,j}$ in equation (6.1) by $L_{1,j}$ and $L_{2,j}$, the original dimensions of the parallelogram $\mathcal{P}_{j,\ell}$. In order to fit $\mathcal{P}_{j,\ell}$ into a rectangle with the same dimensions, we need to copy the data by periodicity, or wrap-around, as illustrated in Figure 3. This is just a relabeling of the frequency samples, of the form

$$n_1' = n_1 + m_1 L_{1,j}, \qquad n_2' = n_2 + m_2 L_{2,j},$$

for some adequate integers m_1 and m_2 themselves depending on n_1 and n_2.

The 2D inverse FFT of the wrapped array therefore reads

$$c^{D}(j,\ell,k) = \frac{1}{n^2} \sum_{n_1=0}^{L_{1,j}-1} \sum_{n_2=0}^{L_{2,j}-1} W(\tilde{U}_{j,\ell}\hat{f})[n_1,n_2] e^{2\pi i (k_1 n_1/L_{1,j} + k_2 n_2/L_{2,j})}. \tag{6.2}$$

Notice that the wrapping relabeling leaves the phase factors unchanged in the above formula, so we can also write it as[3]

$$c^D(j,\ell,k) = \frac{1}{n^2} \sum_{n_1=-n/2}^{n/2-1} \sum_{n_2=-n/2}^{n/2-1} \tilde{U}_{j,\ell}[n_1,n_2] \hat{f}[n_1,n_2] e^{2\pi i (k_1 n_1/L_{1,j} + k_2 n_2/L_{2,j})}.$$

It is then easy to conclude that we have correctly downsampled the convolution of f with the discrete curvelet, this time at every other $n/L_{1,j} \times n/L_{2,j}$ pixels. The following statement establishes precisely this fact, i.e., that the curvelet transform computed by wrapping is as geometrically faithful to the continuous transform as the sampling on the grid allows.

Proposition 6.1. *Let $\varphi^D_{j,\ell}$ be the "mother curvelet" at scale j and angle ℓ,*

$$\varphi^D_{j,\ell}(x) = \frac{1}{(2\pi)^2} \int e^{i\langle x,\omega\rangle} \tilde{U}_{j,\ell}(\omega)\, d\omega,$$

and $\varphi^\sharp_{j,\ell}$ denote its periodization over the unit square $[0,1]^2$,

$$\varphi^\sharp_{j,\ell}(x_1,x_2) = \sum_{m_1 \in \mathbf{Z}} \sum_{m_2 \in \mathbf{Z}} \varphi^D_{j,\ell}(x_1+m_1, x_2+m_2).$$

In exact arithmetic, the coefficients in the East and West quadrants are given by

$$c^D(j,\ell,k) = \frac{1}{n^2} \sum_{t_1=0}^{n-1} \sum_{t_2=0}^{n-1} f[t_1,t_2]\, \overline{\varphi^\sharp_{j,\ell}}\left(\frac{t_1}{n} - \frac{k_1}{L_{1,j}}, \frac{t_2}{n} - \frac{k_2}{L_{2,j}}\right). \qquad (6.3)$$

This is a discrete circular convolution if and only if $L_{1,j}$ and $L_{2,j}$ both divide n. For angles in the North and South quadrants, reverse the roles of $L_{1,j}$ and $L_{2,j}$.

Proof. See appendix 10. □

Notice that the actual value of x_μ, the center of $\varphi_\mu(x)$ in physical space, is implicit in formula (6.3). If φ_μ is centered at the origin when $k_1 = k_2 = 0$, then

$$x_\mu = \left(\frac{k_1}{L_{1,j}}, \frac{k_2}{L_{2,j}}\right)$$

when the angle is $-\pi/4 \le \theta_\ell < \pi/4$, and

$$x_\mu = \left(\frac{k_1}{L_{2,j}}, \frac{k_2}{L_{1,j}}\right)$$

for angles $\pi/4 \le \theta_\ell < 3\pi/4$.

[3]The leading factor $\frac{1}{n^2}$ is not the standard one for the inverse FFT (that would be $\frac{1}{L_{1,j}L_{2,j}}$), but this choice of normalization is useful in the formulation of proposition 6.1. Yet another choice of normalization will be made later to make the transform an isometry.

6.2 Isometry and Inversion

In practice the curvelet coefficients are normalized as follows,

$$c^{D,N}(j,\ell,k) = \frac{n}{\sqrt{L_{1,j}L_{2,j}}} c^D(j,\ell,k),$$

where $L_{1,j}, L_{2,j}$ are the sidelengths of the parallelogram $\mathcal{P}_{j,\ell}$. Equipped with this normalization, we have the Plancherel relation

$$\sum_{t_1,t_2} |f[t_1,t_2]|^2 = \sum_{j,\ell,k} |c^{D,N}(j,\ell,k)|^2.$$

This is easily proved by noticing that every step of the transform is isometric.

- The discrete Fourier transform, properly normalized,
$$f[t_1,t_2] \to \frac{1}{n}\hat{f}[n_1,n_2]$$
is an isometry (and unitary).
- The decomposition into different scale-angle subbands,
$$\hat{f}[n_1,n_2] \to \{\tilde{U}_{j,\ell}[n_1,n_2]\hat{f}[n_1,n_2]\}_{j,\ell}$$
is an isometry because the windows $\tilde{U}_{j,\ell}$ are constructed to obey $\sum_{j=0}^{J}\sum_\ell \tilde{U}_{j,\ell}(\omega)^2 = 1$.
- The wrapping transformation is only a relabeling of the frequency samples, thereby, preserving ℓ^2 norms.
- The local inverse Fourier transform (6.2) is an isometry when properly normalized by $\frac{1}{\sqrt{L_{1,j}L_{2,j}}}$.

Owing to this isometry property, the inverse curvelet transform is simply computed as the adjoint of the forward transform. Adjoints can typically be computed by "reversing" all the operations of the direct transform. In our case,

1. For each scale/angle pair (j,ℓ), perform a (properly normalized) 2D FFT of each array $c^{D,N}(j,\ell,k)$, and obtain $W(\tilde{U}_{j,\ell}\hat{f})[n_1,n_2]$.
2. For each scale/angle pair (j,ℓ), multiply the array $W(\tilde{U}_{j,\ell}\hat{f})[n_1,n_2]$ by the corresponding wrapped curvelet $W(\tilde{U}_{j,\ell})[n_1,n_2]$ which gives
$$W(|\tilde{U}_{j,\ell}|^2\hat{f})[n_1,n_2].$$
3. Unwrap each array $W(|\tilde{U}_{j,\ell}|^2\hat{f})[n_1,n_2]$ on the frequency grid and add them all together. This recovers $\hat{f}[n_1,n_2]$.
4. Finally, take a 2D inverse FFT to get $f[t_1,t_2]$.

In the wrapping approach, both the forward and inverse transform are computed in $O(n^2 \log n)$ operations, and require $O(n^2)$ storage.

7 Extensions

7.1 Curvelets at the Finest Scale

The design of appropriate basis functions at the finest scale, or outermost dyadic corona, is not as straightforward for directional transforms like curvelets as it is for 1D or 2D tensor-based wavelets. This is a *sampling* issue. If a fine-scale curvelet is sampled too coarsely, the pixelization will make it look like a checkerboard and it will not be clear in which direction it oscillates anymore. In the frequency domain, the wedge-shaped support does not fit in the fundamental cell and its periodization introduces energy at unwanted angles.

The problem can be solved by assigning wavelets to the finest level. When $j = J$, the unique sampled window $\tilde{U}_J[n_1, n_2]$ is so constructed that its square forms a partition of unity, together with the curvelet windows. A full 2D inverse FFT can then be performed to obtain the wavelet coefficients. This highpass filtering is very simple but goes against the philosophy of directional basis elements at fine scale. Wavelets at the finest scale are illustrated in Figure 11 (top row).

In this section, we present the next simplest solution to the design of faithful curvelets at the finest scale. For simplicity let us adopt the sampling scheme of the wrapping implementation, but a parallel discussion can be made for the USFFT-based transform. As above, denote by J the finest level. By construction, the standard curvelet window $\tilde{U}_{j,\ell}[n_1, n_2]$ is obtained by sampling a continuous profile $\tilde{U}_{j,\ell}(\omega_1, \omega_2)$ at $\omega_1 = 2\pi n_1$, $\omega_2 = 2\pi n_2$. When $j = J$, the profile $\tilde{U}_{j,\ell}$ overlaps the border of the fundamental cell but can still be sampled according to the formula

$$\tilde{U}_{J,\ell}[(n_1 + \frac{n}{2}) \bmod n - \frac{n}{2}, (n_2 + \frac{n}{2}) \bmod n - \frac{n}{2}] = \tilde{U}_{J,\ell}(2\pi n_1, 2\pi n_2). \quad (7.1)$$

The indices n_1, n_2 are still chosen such that $\tilde{U}_{J,\ell}$ is evaluated on its support. The latter is by construction sufficiently small so that no confusion occurs when taking modulos. In effect we have just copied $\tilde{U}_{J,\ell}$ by periodicity inside the fundamental cell. The windows $\tilde{U}_{J,\ell}(\omega_1, \omega_2)$ must be chosen adequately so that the discrete arrays $\tilde{U}_{J,\ell}[n_1, n_2]$, now with $n_1, n_2 = -n/2 \ldots n/2 - 1$, obey the isometry property together with the other windows,

$$\sum_{j=0}^{J} \sum_{\ell} |\tilde{U}_{j,\ell}[n_1, n_2]|^2 = 1.$$

In fact, this is the case if $\tilde{U}_{J,\ell}$ is chosen as in Section 3 (after an appropriate rescaling).

Periodization in frequency amounts to sampling in space, so finest-scale curvelets are just undersampled standard curvelets. This is illustrated in Figure 11 (middle row). What do we loose in terms of aliasing? Spilling over by periodicity is inevitable, but here the aliased tail consists of essentially only one-third of the frequency support. Observe in Figure 11 (middle right) that a large fraction of the energy of the discrete curvelet still lies within the fundamental cell. Numerically, the non-aliased part amounts to about 92.4% of the total squared ℓ^2 norm $\|\varphi^D_{j,\ell,k}\|^2_{\ell^2}$. The "checkerboard" look of undersampled curvelets, mentioned above, is shown in Figure 11 (bottom right).

Accordingly, the definition of wrapping of an array $d[n_1, n_2]$, in the presence of undersampled curvelets, is modified to read:

$$Wd[n_1 \bmod L_{1,j}, n_2 \bmod L_{2,j}] = d[(n_1 + \frac{n}{2}) \bmod n - \frac{n}{2}, (n_2 + \frac{n}{2}) \bmod n - \frac{n}{2}] \qquad (7.2)$$

The new modulo that appears in the above equation (compare with (3.12)) prevents data queries outside $[0, n]^2$, which would otherwise happen if equation (3.12) were used naively. Instead, data is folded back by periodicity onto the fundamental cell, ultimately resulting in aliased basis functions.

The definitions of forward and inverse curvelet transforms, as well as their properties, otherwise go unchanged. Proposition 6.1 and its proof do not have to be changed either: they are already compatible with equation (7.2).

7.2 Windows over Junctions between Quadrants

The construction of windows $\tilde{U}_{j,\ell}$ explained in Section 3.1 make up an orthonormal partitioning of unity as long as the window is supported near wedges that do not touch neither of the two diagonals. There are 8 "corner" wedges per scale calling for a special treatment, and corresponding to angles near $\pm\pi/4$ and $\pm 3\pi/4$, see Figure 7 on the left. In these exceptional cases, creating a partition of unity is not as straightforward. This is the topic of this section.

It is best to follow Figure 7 while reading this paragraph. Consider a trapezoid in the top quadrant and corresponding to an angle near $3\pi/4$ as in the figure. The grey trapezoid is the corner wedge near which the curvelet is supported, but the actual support of the curvelet is the nonconvex hexagon bounded by the dash-dotted line. As before, the corner curvelet window is given as a product of the radial window W_j and of the angular window $V_{j,\ell}$,

$$\hat{\varphi}^D_{j,\ell}(\omega) = W_j(\omega) V_{j,\ell}(\omega).$$

We decompose the corner window $V_{j,\ell}$ into a left-half and a right-half. The right-half is given by the standard construction presented earlier. It is a function of $\frac{\omega_1}{\omega_2}$. The left-half of the window is constructed as a member of a square-root of a partition of unity designed in a frame rotated by 45 degrees with respect to the Cartesian axes. The left-half of the window is a function of $\frac{\omega_1+\omega_2}{\omega_1-\omega_2}$. The left and right-halves agree on the line where they are stitched together (on the figure, it is the tilted line, first to the right of the diagonal $\omega_1 = -\omega_2$). Along the border line, they are both equal to one and they have at least a couple of vanishing derivatives in all directions. Again, the partition of unity can be designed so that all these derivatives are zero. By construction, our set of windows obeys the partition of unity property, equation (3.2).

7.3 Other Frequency Tilings

The construction of curvelets is based on a polar dyadic-parabolic partition of the frequency plane, also called FIO tiling, as explained in Section 2. However, the approach is flexible, and can be used with a variety of choices of parallelepipedal tilings, for example, including based on principles besides parabolic scaling. For example:

- A *directional wavelet* transform is obtained if, instead of dividing each dyadic corona into $C \cdot 2^{\lfloor j/2 \rfloor}$ angles, we divide it into a constant number, say 8 or 16 angles, regardless of scale

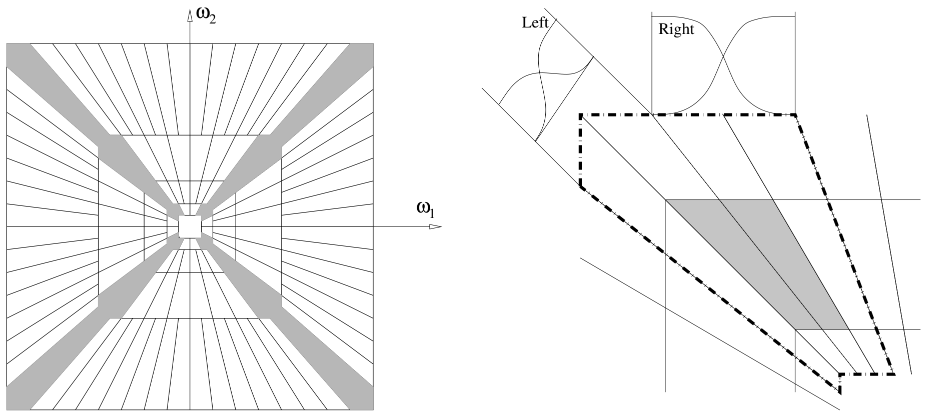


Figure 7: Left: The corner wedges appear in grey. Right: Detail of the construction of a partition of unity over the junction between quadrants.

as in [35]. This can be realized by dropping the requirement that wedges be split as scale increases.

- A *ridgelet* transform is obtained by subdividing each dyadic corona into $C \cdot 2^j$ angles. This can be achieved by subdividing every angular wedge every time the scale index j increases (not just every other time, as for curvelets.)
- A *Gabor* analysis is obtained if, instead of considering bandpass concentric annuli of thickness increasing like a power of two, we consider the thickness to be the same for all annuli. In other words, coronae with fixed width are substituted for dyadic coronae. The number of wedges into which an annulus should be divided is proportional to its length, or equivalently, its distance to the origin.
- More generally, one can create an adaptive partitioning of the frequency plane that best matches the features of the analyzed image. This is the construction of *ridgelet packets* as explained in [24]. A best basis strategy can then be overlaid on the packet construction to find the optimal partitioning in the sense that it minimizes an additive measure of "entropy," or sparsity.

In all these cases both the USFFT and wrapping strategies carry over without essential modifications and yield tight or nearly tight frames. The design problem is reduced to the construction of a smooth partition of unity that indicates the desired frequency tiling.

7.4 Higher Dimensions

Curvelets exist in any dimension [5]. In 3 dimensions for example, curvelets are little plates of side-length about $2^{-j/2}$ in two directions and thickness about 2^{-j} in the orthonormal direction. They vary smoothly in the two long directions and oscillate in the short one (the 3D parabolic scaling matrix is of the form $\mathrm{diag}(2^{-j/2}, 2^{-j/2}, 2^{-j})$). Just as 2D curvelets provide optimally efficient representations of 2D objects with singularities along smooth curves, 3D curvelets would provide efficient representations of 3D objects with singularities along smooth 2D surfaces, and more generally, of objects with singularities along smooth manifolds of codimension 1 in higher dimensions.

The algorithms for 3D discrete curvelet transforms are similar to their 2D analogs. We first decompose the object into dyadic annuli based on concentric cubes. Each annulus is subdivided into trapezoidal regions obeying the usual frequency parabolic scaling (one long and two short directions), see Figure 8. (Note that they are now 6 components corresponding to the 6 faces of the cube.)

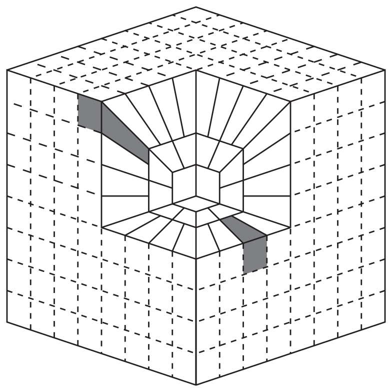


Figure 8: The dyadic-parabolic frequency tiling in 3D. Curvelets are supported near the gray regions.

Both transforms carry over to 3 dimensions and we only rehearse the minor modifications.

1. The 3D FDCT via wrapping just wraps the 3D parallelepipeds instead of their 2D analogs.
2. In the 3D FDCT via USFFT, we need to resample $\hat{f}[n_1, n_2, n_3]$ on 2D planes which are orthogonal to the coordinate axes. Fix a scale, and an abscissa n_1 as before. The problem is to evaluate $\hat{f}$ on the 2D irregular grid $(n_1, m_2 + n_1 \tan\theta_{\ell_2}, m_3 + n_1 \tan\theta_{\ell_3})$ where $-L_{2,j} \leq m_2, m_3 < L_{2,j}$. Set $c[u_2, u_3] = \sum_{t_1} f[t_1, u_2, u_3] e^{-i2\pi n_1 t_1/n}$. We need to sample

$$g(\omega_2, \omega_3) = \sum_{-n/2 \leq u_2, u_3 < n/2} c[u_2, u_3]\, e^{-i(u_2\omega_2 + u_3\omega_3)/n}, \tag{7.3}$$

   on the family of meshes $(\omega^{\ell_2}_{m_2}, \omega^{\ell_3}_{m_3})$ where $\omega^{\ell_2}_{m_2} := 2\pi(m_2 + n_1 \tan\theta_{\ell_2})$ and likewise for $\omega^{\ell_3}_{m_3}$. The key point is that one can compute (7.3) with about $2n$ one-dimensional USFFTs; first, one applies the USFFT along the columns by holding u_3 constant and thereby obtains the partial sums $\sum_{-n/2 \leq u_2 < n/2} c[u_2, u_3] \exp(-i(u_2\omega^{\ell_2}_{m_2} + u_3\omega^{\ell_3}_{m_3})/n)$ for all the $\omega^{\ell_2}_{m_2}$; second, the values of (7.3) on the grid of interest are obtained by applying the USFFT along the rows—holding $\omega^{\ell_2}_{m_2}$ constant.

   To summarize, the 3D FDCT via USFFT operates by applying a sequence of 1D USFFTs and, therefore, the 3D FDCT never needs to hold large chunks of data in memory. For an n by n by n Cartesian array, a simple operation count shows that the resampling of $\hat{f}$ on the 2D grid $(n_1, m_2 + n_1 \tan\theta_{\ell_2}, m_3 + n_1 \tan\theta_{\ell_3})$ can be implemented accurately in $O(n^2 \log n)$ flops. Since each 2D plane is touched at most twice, the 3D FDCT via USFFT runs in $O(n^3 \log n)$ flops.

3. The construction of junction windows (described in Section 7.2 for 2D FDCTs) is a little more delicate since one needs to consider more cases. One possible solution is to develop a partition of unity over the unit sphere which is then mapped onto the cube. The detailed algorithm and numerical results of the 3D transform will be presented in a future report.

In short, 3D FDCTs follow exactly the same architecture as 2D FDCTs, and the forward, adjoint, and inverse transforms all run in $O(N \log N)$ for Cartesian arrays of size $N = n^3$ voxels.

7.5 Nonperiodic Image Boundaries

An (unfortunate) consequence of using the DFT to define our transform is that the image is implicitly considered as a periodic array. The leftmost and rightmost pixels in a given row, or the top and bottom pixels in a given column, are considered immediate neighbors as much as ordinary adjacent pixels are. By construction, a substantial number of basis functions appear to be supported on two (or more) very distant regions of the image, because they overlap the image boundary and get copied by periodicity. Let us call them "boundary curvelets."

Periodization may result in unwanted curvelet-looking artifacts near the image boundary, for example in image denoising experiments. The reason for the presence of these artifacts, however, is not the same for curvelets and for wavelets. In order to understand this phenomenon, we need to sort curvelets according to their orientation.

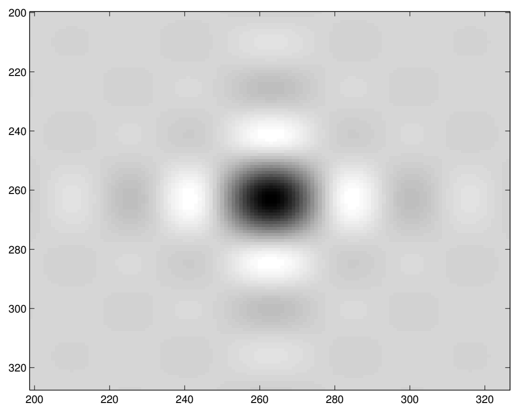
1. Boundary curvelets that are *aligned* with a boundary edge mostly respond to the artificial discontinuity created by periodization. Since the basis elements very closely follow the boundary, the visual effect of a big coefficient is minor.

2. Boundary curvelets *misaligned* with respect to the boundary edge are assigned big coefficients when they respond to geometrical structure on the *opposite side* of the image, across the edge. This causes the most severe visual artifacts.

In the remainder of this section, we present a few (somewhat naive) solutions to artifacts of type 2, when boundary curvelets are misaligned.

The most obvious remedy is to pad the image with zeros to make it twice larger in both directions. The curvelet transform is then applied to the extended image, increasing the redundancy by a factor 4. The blank surrounding region is large enough to prevent boundary curvelets from wrapping around. The inverse or adjoint transform would then would have an extra step, clipping off the extra pixels.

If we postulate that artifacts of type 2 are caused by boundary curvelets forming an angle greater than 45 degrees with the edge, then it is not necessary to zeropad in all directions. The image should only be extended horizontally for mostly horizontal curvelets, and vertically for mostly vertical curvelets. The zeropadding will make the image twice larger in only one direction, depending on


(a)

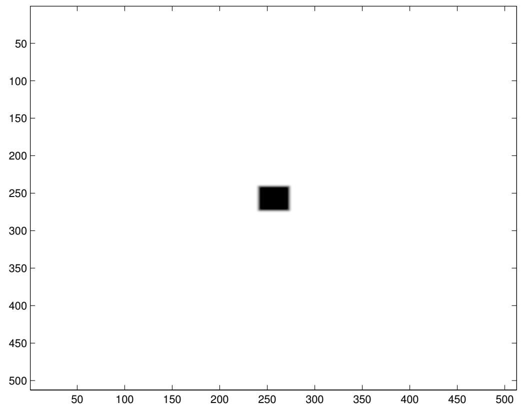

(b)

Figure 9: At the coarsest level, curvelets are nondirectional and are Meyer scaling functions. (a) Spatial-side. The color map is as follows: white is most negative, zero corresponds to some tone of grey, and black is most positive. (b) Frequency-side (modulus of the Fourier transform). The level of grey indicates values from zero (white) to one (black).

the orientation of the subband considered. In this case, the increase in redundancy is only of a factor 2.

In principle it would be advantageous to make the width of the zeropadding not only angle-dependent, but also scale-dependent. More precisely, the width of the padding does not have to be bigger than a factor times the length of misaligned curvelets, i.e., $C \cdot 2^{-\lfloor j/2 \rfloor}$. The gain in redundancy would be obvious. There is a complication, however, in considering scale-dependent or even angle-dependent paddings. Different subbands will correspond to different grids and extra care will be needed to properly re-design the transform to make it an isometry. It will be necessary to rethink the notion of discrete partition of unity to accommodate interpolation between different grids.

We have not pursued this issue much further, but a better handling of image boundaries would improve the current architecture of the curvelet transform for image processing applications.

8 Numerical Examples

We start this section by displaying a few curvelets in both the spatial and the frequency domain, see Figures 9 (coarsest scale curvelets), 10 and 11 (curvelets at the finest level where one can choose between wavelets and curvelets). Localization in both space and frequency is apparent. The digital curvelets seem faithful to their continuous analog. In the spatial domain, they are smooth along and oscillatory across the ridge. In the frequency domain, they are sharply localized.

Next, Tables 2 and 3 report the running time of both FDCTs on a sequence of arrays of increasing size. T_{Fwd}, T_{Inv} and T_{Adj} are running times of the forward, inverse and adjoint transforms respectively (we only give T_{Inv} for the FDCT via wrapping since the inverse is the same as the adjoint). The column T_{Fwd}/T_{FFT} gives the ratio between the running time of the FDCT and that of the FFT

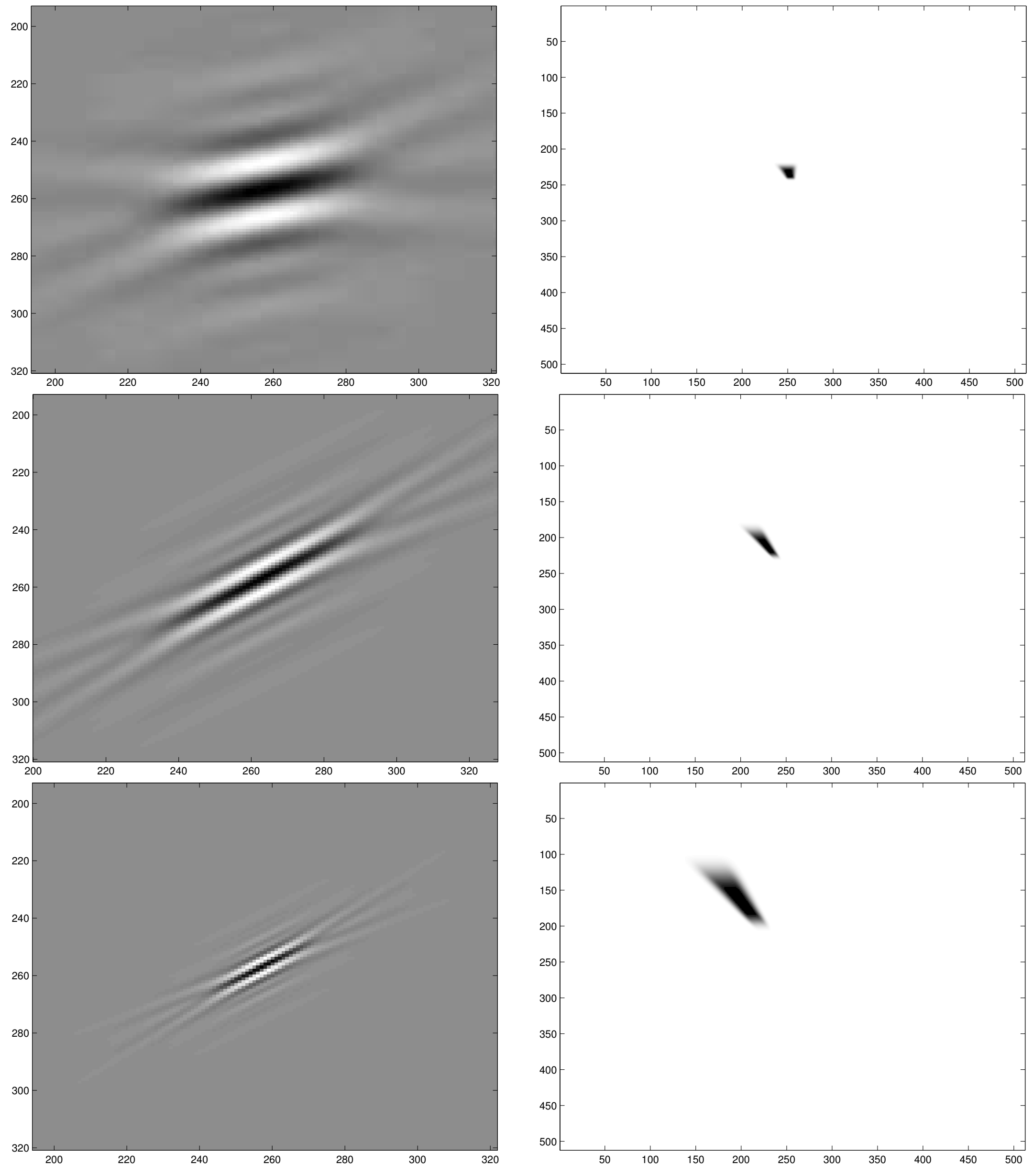
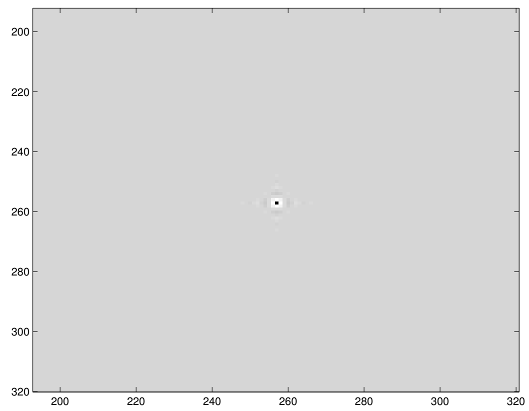
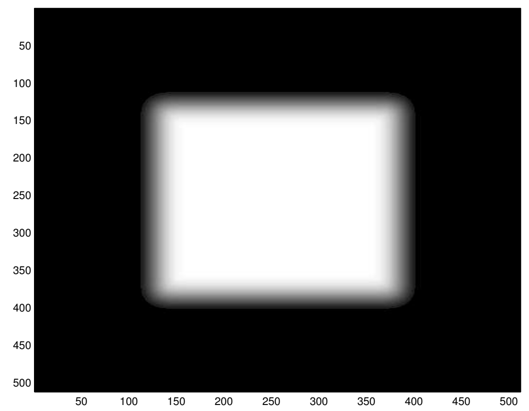
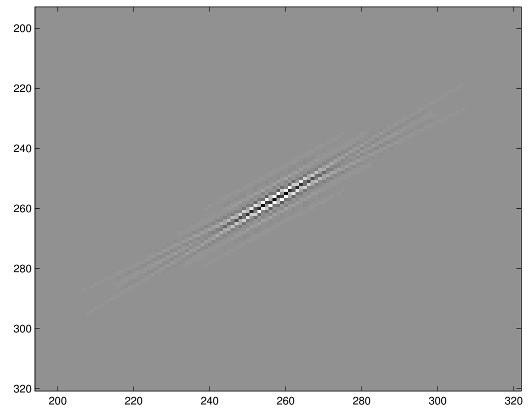


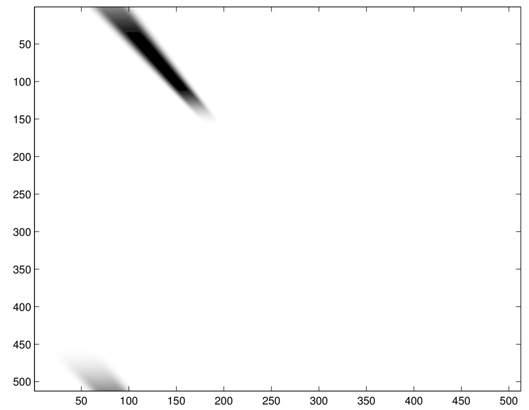
Figure 10: Curvelets at increasingly fine scales. The left panels represent curvelets (real part) in the spatial domain (as functions of the spatial variable x). The right panels show the modulus of the Fourier transform (as functions of the frequency variable ω). The color map is the same as in Figure 9.

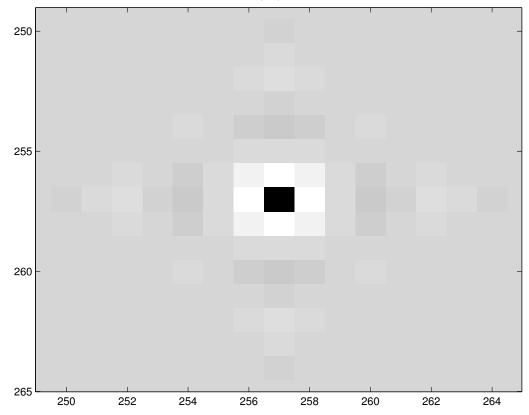
on an array of the same size. The accuracy or ℓ^2-error is computed as $\|f - C_{Inv}C_{Fwd}f\|_{\ell^2}/\|f\|_{\ell^2}$ where C_{Inv} and C_{Fwd} are the the forward and inverse FDCTs. The FDCT via wrapping achieves machine accuracy because of the exact numerical tightness of the digital transform. The FDCT via USFFT also achieves high accuracy, i.e., of the order of 10^{-6}. Although both transforms have


(a)


(b)


(c)


(d)


(e)

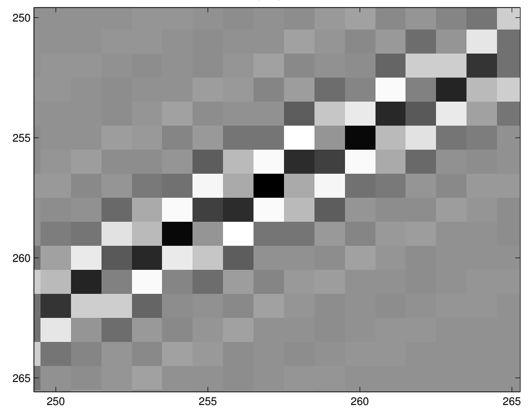

(f)

Figure 11: Wavelets and curvelets at the finest scale. Meyer wavelet in space (a) and frequency (b). Undersampled curvelet in space (c) and frequency (d). (e) Zoom of (a). (f) Zoom of (c).

low running times, the USFFT transform is somewhat slower; this is due to the interpolation step in the forward transform and to the CG iterations in the inverse transform.

We then illustrate the potential of FDCTs with several examples. The wrapping-based implementa-

Image size	T_{Fwd}(s)	T_{Inv}(s)	T_{Fwd}/T_{FFT}	ℓ^2 error
128×128	0.040458	0.039520	11.2383	4.5450e-16
256×256	0.174807	0.176519	8.8286	4.8230e-16
512×512	0.829820	0.868141	6.0793	4.8908e-16
1024×1024	4.394066	4.482452	7.7224	5.6303e-16
2048×2048	20.01692	23.02144	7.7567	6.3018e-16

Table 2: Running time and error for the wrapping-based transform.

Image size	T_{Fwd}(s)	T_{Adj}(s)	T_{Inv}(s)	T_{Fwd}/T_{FFT}	ℓ^2 error
128×128	0.088832	0.091578	1.006522	24.6756	1.4430e-06
256×256	0.376838	0.390533	4.002353	19.0322	8.8154e-07
512×512	2.487052	2.579102	35.09599	18.2202	5.3195e-07
1024×1024	16.47702	16.87764	129.3631	28.9579	3.2390e-07
2048×2048	62.42980	65.09365	566.1732	24.1920	3.4305e-06

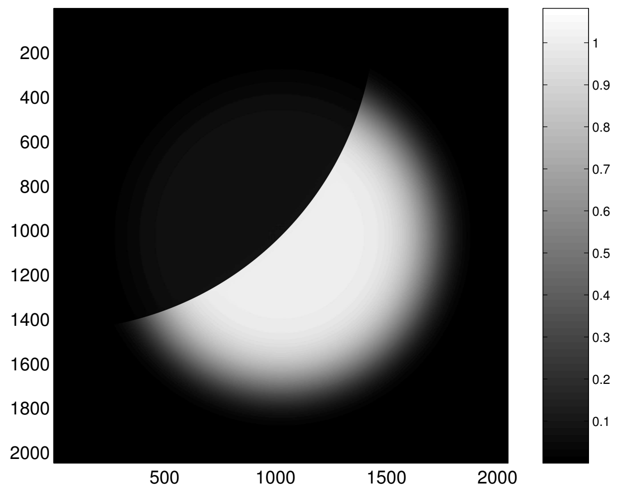
Table 3: Running time and error for the USFFT-based transform.

tion has been used for all experiments. In the first example, we compare the decay of the coefficients of the curvelet and various wavelet representations on images with curve-like singularities. Our first input image—shown in Figure 12 (a)—is singular along a smooth curve and is otherwise perfectly smooth (this image is de-aliased to remove the artifacts due to pixelization). To compensate for the redundancy of the curvelet transform and to display a meaningful comparison, we extract a fraction of the entries of the curvelet coefficient table so that the number of curvelet and wavelet coefficients is identical. The extracted curvelet entries are renormalized to preserve the overall ℓ^2 norm. Figure 12 (b) shows the values of the coefficients sorted in decreasing order of magnitude. The faster the decay, the better. The sparsity analysis is complemented by the quantitative study of partial reconstructions of f, where we have again used redundancy compensation as explained above. Figure 12 (c) shows the PSNR of best m-term approximation,

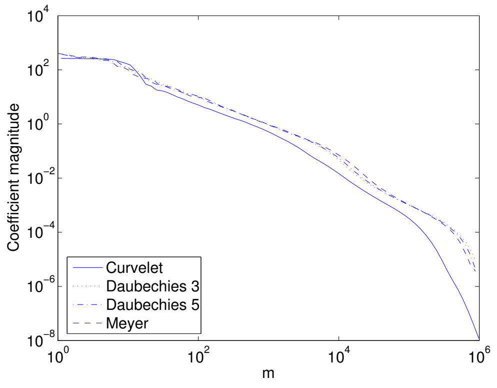
$$PSNR = 20 \log_{10} \left(\frac{\max(f(x)) - \min(f(x))}{\|f - f_m\|_2} \right) \qquad (dB)$$

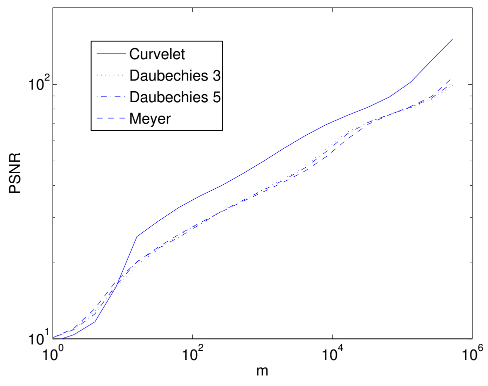
where f_m is the partial reconstruction of f using the m largest coefficients in magnitude, in the curvelet (or wavelet) expansion (note that because of the redundancy of the FDCT, there are better ways of obtaining partial reconstructions).

The second input image—shown in Figure 13 (a)—is a synthetic seismogram corresponding to the acoustic response of a one-dimensional layered medium to a point source. The decay of the coefficients and the partial reconstruction error for this image are shown in Figure 13 (b) and (c) respectively. Our experiments suggest that FDCTs outperform, by a significant margin, traditional wavelet representations on these types of image data. Synthetic seismic images seem to be the ideal setting for curvelets because they are prepared as solutions to a wave equation in simple layered media, with a bandlimited point excitation. The solution itself is therefore very close to being bandlimited. We are in the setting of proposition 6.1: when the data are oscillatory yet properly sampled, curvelets are expected to be completely faithful to the continuous transform, explaining the good denoising performance.



(a)

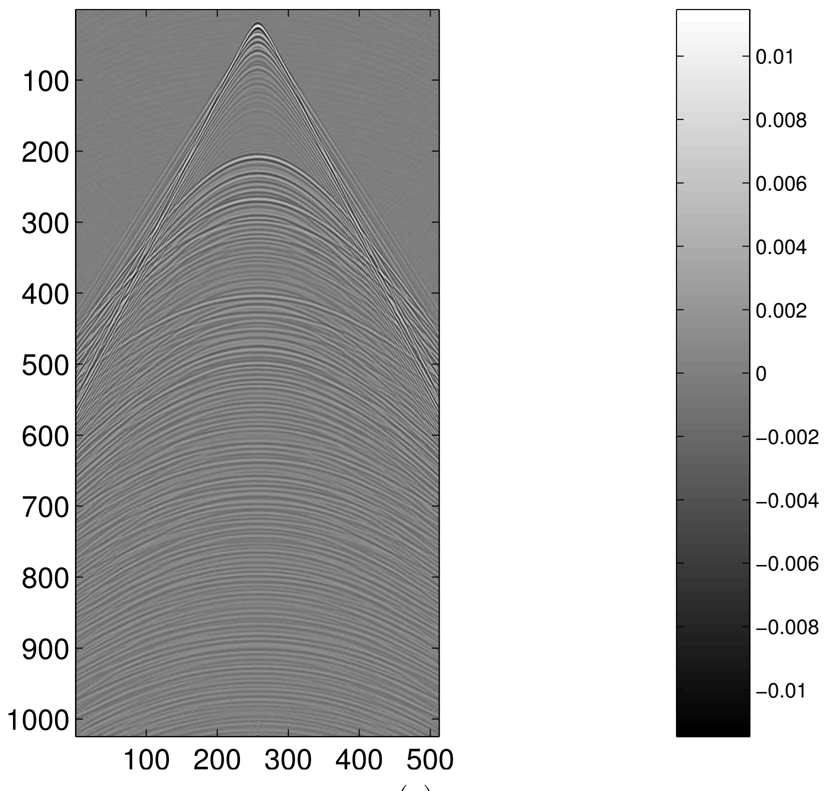




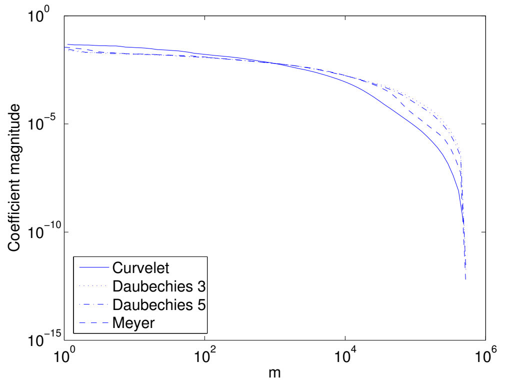
(b) (c)

Figure 12: Sparsity analysis of the curvelet and wavelet representations of a singular object. (a) Input image. (b) Magnitude of the coefficients sorted in descending order. (c) PSNR for partial reconstruction with the m largest coefficients in magnitude. The horizontal line at 40 dB indicates a typical "visually acceptable" level of reconstruction.

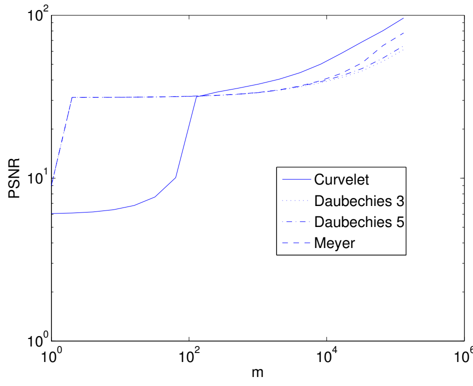
The second example is denoising. The original image is the seismogram used in the previous example (see Figure 13 (a)). The noise-to-signal ratio is set to 10%, which corresponds to PSNR = 20.0 dB. A denoising algorithm based on our curvelet transform results in an image with PSNR = 37.6 dB. (see Figure 14 (c)) while a traditional wavelet denosing algorithm (Symmlet 8 in WaveLab, shift-invariant hard thresholding at 2.5σ) gives PSNR = 30.8 dB. (see Figure 14 (d)). The curvelet denoising algorithm used above is a simple shift-invariant block-thresholding of the wrapping-based curvelet transform (with curvelets at the finest scale) and is available as Matlab code in CurveLab. (For an image of size 1024×512, the whole procedure runs in less than 90 seconds on a standard desktop.)



(a)



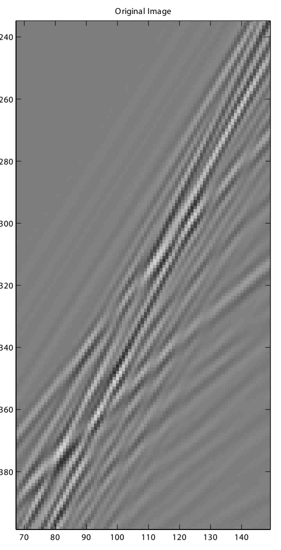
(b)



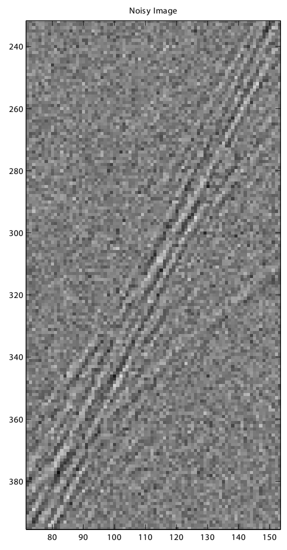
(c)

Figure 13: Sparsity analysis of the curvelet and wavelet representations of a seismogram. (a) Synthetic seismogram corresponding to the acoustic response of a one-dimensional layered medium to a point source, courtesy of Eric Verschuur and Felix Herrmann. The x-axis is the offset from the source and the y-axis is time. (b) Decay of the coefficients. (c) Partial reconstruction error, measured in PSNR.

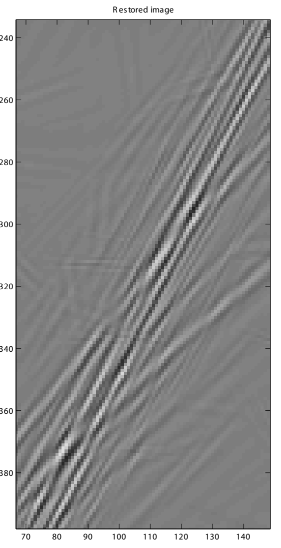
In the introduction section, we pointed out that curvelets were especially well adapted to simultaneously represent the solution operators to large classes of wave equations and the wavefields that are solutions to those equations. In our third example, we consider the constant coefficient



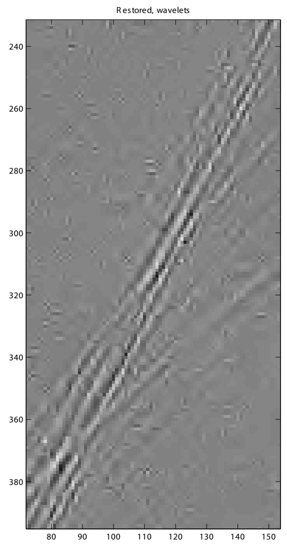
(a)



(b)



(c)



(d)

Figure 14: Image denoising using curvelets. (a) The Original image (zoom). (b) Noisy image (Gaussian white noise with $\sigma = 10\%$ of the maximum intensity), PSNR = 20.0 dB. (c) Denoised image using curvelets, PSNR = 37.6 dB. (d) Denoised image using wavelets, PSNR = 30.8 dB.

second-order wave equation with periodic boundary condition

$$u_{tt} - \Delta u = 0 \quad x \in [0, 1) \times [0, 1).$$

We discretize the domain with a 512-by-512 Cartesian grid, and take as initial wavefield a delta function located at the center of the domain, see Figure 15 (a). The solution at a later time is known analytically, and may therefore be computed exactly. We use the FDCT to compress the wavefield at time $t = 0.25$ and $t = 0.75$. Figures 15 (b) and (c) show the approximate wavefields reconstructed from only 1.25% of the curvelet coefficients. In both cases, the relative ℓ^2 error is about 10^{-5}.

We have seen that the wavefield is well approximated by just a few curvelets and now study the compressibility of the wave propagator E_t. For simplicity, assume E_t acts on scalar wavefields. From a theoretical point of view, it is known that the entries of $E_t(\mu, \mu') = \langle \varphi_\mu, E_t \varphi_{\mu'} \rangle$ taken from an arbitrary row (fixed μ) or column (fixed μ') decay faster than any negative power law. Figure 15 (d) plots the decay of the matrix coefficients (sorted by decreasing magnitude) for several columns of the propagator matrix E_t at $t = 0.75$ while (e) plots the relative truncation error for those same columns. "Scale" in the legend refers to the scale j' corresponding to μ', the index of the column. Observe that for every column, we achieve a relative error of order 10^{-5} by using about 1% of the largest curvelet coefficients. The data are shown as is; no compensation for redundandy has been made in this experiment.

9 Discussion

The two transforms introduced in this paper were designed with the goal of being as faithful to continuous curvelets as possible. In both cases the main step of the transform is to window the data in frequency with prescribed windows, sampled on the same grid as the data. This sampling in frequency is the *only* distortion that curvelets incur in the digital transforms. This issue is inevitable but minor, since it is equivalent to periodization in space where curvelets decay fast. Recall that the other potential source of error, spatial sampling, is a nonissue here since curvelets are nearly bandlimited.

Both transforms are fast and the wrapping variant is to our knowledge the fastest curvelet transform currently available. Computing a direct or inverse transform in C++ takes about the same time as 6 to 10 FFTs using FFTW (available at http://www.fftw.org), which can hardly be improved upon.

9.1 Open Problems

In addition to removing periodicity, the curvelet transform can be made more useful or attractive in a number of ways and we discuss a few opportunities.

- Firstly, the redundancy of our transform is about 2.8 when wavelets are chosen at the finest scale, and 7.2 otherwise. For certain image processing tasks, redundant transformations may be of benefit, but for others, digital transforms with low redundancy might be more desirable. It is not immediate how one could adapt our ideas to reduce the redundancy while keeping the isometry property and remaining faithful to the continuous transform. In particular, it is not known whether one can construct orthonormal bases of curvelets. We regard this problem as very significant and extremely challenging.
- Secondly, compactly-supported (or at least exponentially-decaying) curvelets would have the

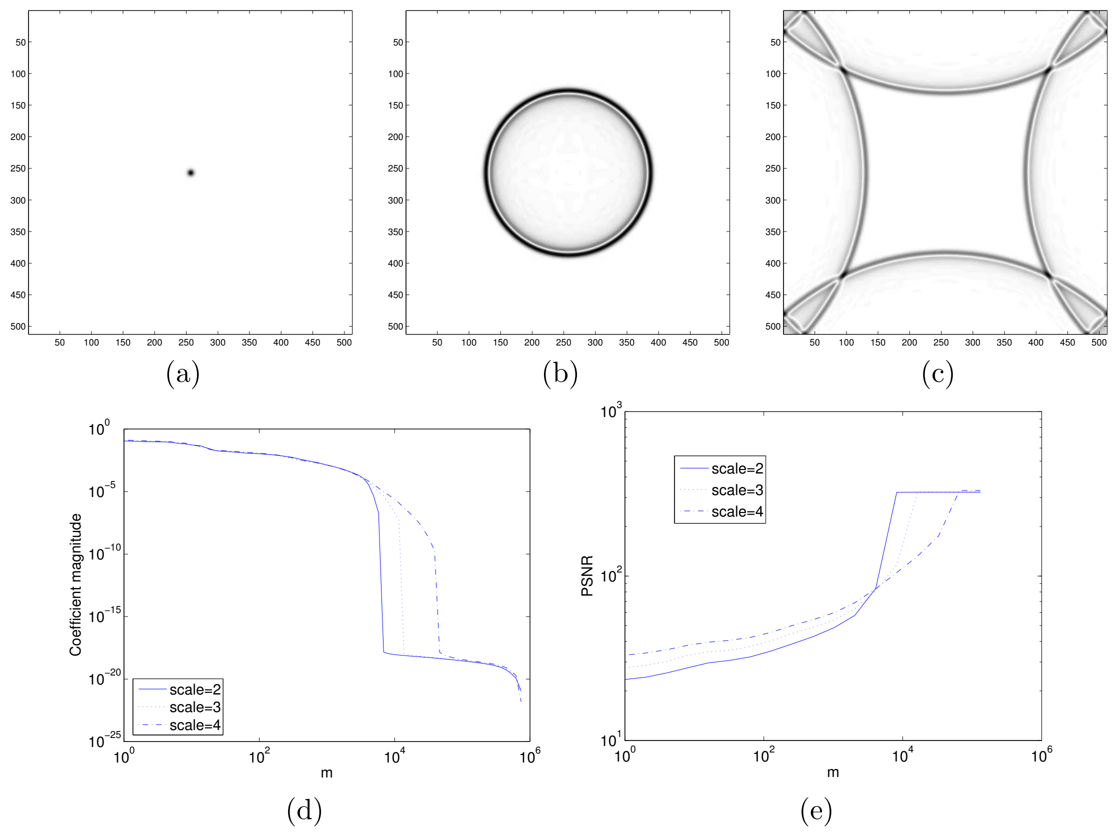


Figure 15: Compression of the wavefield and of the solution operator to the wave equation with periodic boundary conditions. (a) The initial condition is a delta function located at the center of the domain. (b) Approximate solution at $t = 0.25$. (c) Approximate solution at $t = 0.75$. Both approximations only use 1.25% of nonzero curvelet coefficients. (d) Magnitude of the matrix entries (rearranged in descending order) of the solution operator E_t at $t = 0.75$ taken from three columns corresponding to three curvelets at various scales. (e) For the same three columns, truncation error obtained by keeping the m largest entries, measured in PSNR.

potential to yield sparser expansions of images with geometrical regularity. We consider the design of compactly-supported curvelet tight frames as another interesting open problem.

- Thirdly, although proposition 6.1 settles the accuracy question when data is bandlimited, it remains to be studied how faithful the curvelet transform can be in the presence of *aliased data*. Aliasing occurs when, for example, a function with a discontinuity is discretized by pointwise evaluation. In image processing this typically happens in the neighborhood of an edge. Yet not all hope is lost, because of geometric regularity along the edge. A complete theory of approximation for curvelets (or wavelets for that matter) needs to solve this sampling issue.

9.2 Relationships with Other Works

The notion of directional multiscale transform originated independently in different fields in the early nineties. Without the claim of being exhaustive, let us only mention continuous wavelet theory [32] and steerable pyramids in the field of computer vision [35, 34]. The latter approach was the first practical, data-friendly strategy to extract information at different scales and angles.

A more recent, very interesting attempt at implementing low-redundancy curvelets, was introduced by Minh Do and Martin Vetterli, in [16]. The construction is based on a filterbank decomposition of the image in both scale and angle. The resulting basis functions are called "contourlets," and form a tight frame with redundancy 4/3. The contourlet transform has a very fast $O(n^2 \log n)$ implementation as well, at least when contourlets are selected to be compactly supported. The only problem with this construction is that it is not faithful to the idea of the curvelet transform in the sense that for most choices of filters in the angular filterbank, contourlets are not sharply localized in frequency. On the practical side, this means that contourlets lack smoothness along the ridge in the spatial domain and exhibit spurious oscillations which may be of source of numerous problems, especially if one wants to use these transforms for scientific computing. On the theoretical side and to the best of our knowledge, contourlets do not allow to formulate as strong theorems in approximation and operator theory as in [5, 10].

The idea of using concentric squares and shears is also central to the construction of tight frames of "shearlets", by Guo, Kutyniok, Labate, Lim, Weiss and Wilson in a recent series of papers [28, 29, 30] starting with [28]. In these papers, they show how to built wavelets or multiwavelets from *composite dilations* and translations. The architecture is similar to that of curvelets, except that the tiling of the frequency plane induced by dilation and pure shearing has a preferred direction—vertical or horizontal.

9.3 Possible Applications

Just as the wavelet transform has been deployed a countless number of times in many fields of science and technology, we expect fast digital curvelet transforms to be widely applicable—especially in the field of image processing and scientific computing.

In image analysis for example, the curvelet transform may be used for the compression of image data, for the enhancement and restoration of images as acquired by many common data acquisition devices (e.g., CT scanners), and for postprocessing applications such as extracting patterns from large digital images, detecting features embedded in very noisy images, enhancing low contrast images, or registering a series of images acquired with very different types of sensors.

Curvelet-based seismic imaging already is already a very active field of research, see for example the recent papers [25, 27] as well as several expanded abstracts of Felix Herrmann and his collaborators, currently available at http://slim.eos.ubc.ca/.

In scientific computing, curvelets may be used for speeding up fundamental computations; numerical propagation of waves in inhomogeneous media is of special interest. Promising applications include seismic migration and computational geophysics.

10 Appendix

Proof of proposition 6.1. By definition, the East and West coefficients are given by the formula

$$c^D(j,\ell,k) = \frac{1}{n^2} \sum_{n_1=0}^{L_{1,j}-1} \sum_{n_2=0}^{L_{2,j}-1} e^{2\pi i k_1 n_1/L_{1,j}} e^{2\pi i k_2 n_2/L_{2,j}} W(\tilde{U}_{j,\ell}\hat{f})[n_1,n_2].$$

Let us change n_1 and n_2 to $n_1' = n_1 + m_1 L_{1,j}$, $n_2' = n_2 + m_2 L_{2,j}$, for appropriate integers m_1, m_2 (themselves depending on n_1 and n_2) so that $(2\pi n_1', 2\pi n_2') \in \mathcal{P}_{j,\ell}$, or more concisely, "$n_1', n_2'$ in tile." This is the unwrapping transformation, and leaves the phase factors unchanged. Notice that $n_1 = n_1' \bmod L_{1,j}$ and $n_2 = n_2' \bmod L_{2,j}$. We can then use the definition of wrapping in equation (3.12) to rewrite

$$c^D(j,\ell,k) = \frac{1}{n^2} \sum_{n_1,n_2 \text{ in tile}} e^{2\pi i k_1 n_1/L_{1,j}} e^{2\pi i k_2 n_2/L_{2,j}}\, \tilde{U}_{j,\ell}[n_1,n_2]\, \hat{f}[n_1,n_2].$$

We recall that the index-to-sample correspondence in the frequency plane is just

$$\tilde{U}_{j,\ell}[n_1,n_2] = \tilde{U}_{j,\ell}(2\pi n_1, 2\pi n_2).$$

It is also valid for $\hat{f}$, if we introduce $\hat{f}(\omega_1,\omega_2)$ as the trigonometric interpolant of the array $\hat{f}[n_1,n_2]$ (3.8). Notice in passing that $\hat{f}(\omega_1,\omega_2)$ is periodic in ω outside of the fundamental cell, so we actually have

$$\hat{f}(2\pi n_1, 2\pi n_2) = \hat{f}[(n_1 + \frac{n}{2}) \bmod n - \frac{n}{2}, (n_2 + \frac{n}{2}) \bmod n - \frac{n}{2}] \qquad (10.1)$$

for every $(n_1,n_2) \in \mathbf{Z}^2$. With this convention the data $f[t_1,t_2]$ itself can be viewed as samples $f(\frac{t_1}{n}, \frac{t_2}{n})$ of f, the inverse (continuous) Fourier transform of $\hat{f}$ restricted to the fundamental cell.

Using this continuous representation of the data, along with equation (7.1) in the case when the modulo is triggered in equation (10.1), $c^D(j,\ell,k)$ obeys

$$c^D(j,\ell,k) = \frac{1}{n^2} \sum_{n_1,n_2 \text{ in tile}} e^{i2\pi(k_1 n_1/L_{1,j} + k_2 n_2/L_{2,j})}\, \hat{\varphi}^D_{j,\ell}(2\pi n_1, 2\pi n_2)\, \hat{f}(2\pi n_1, 2\pi n_2)$$

and since $\hat{\varphi}_{j,\ell}$ is compactly supported, one can extend the sum above to $(n_1,n_2) \in \mathbf{Z}^2$. Introduce the Dirac comb

$$c(\omega_1,\omega_2) = \sum_{n_1 \in \mathbf{Z}} \sum_{n_2 \in \mathbf{Z}} \delta(\omega_1 - 2\pi n_1)\delta(\omega_2 - 2\pi n_2).$$

and rewrite $c^D(j,\ell,k)$ as

$$c^D(j,\ell,k) = \frac{1}{n^2} \int_{\mathbf{R}^2} e^{i\omega_1 \frac{k_1}{L_{1,j}}} e^{i\omega_2 \frac{k_2}{L_{2,j}}} c(\omega) \hat{\varphi}^D_{j,\ell}(\omega) \hat{f}(\omega)\, d\omega.$$

Our claim follows from Parseval's identity which states that $\int \hat{u}\bar{\hat{v}} = (2\pi)^2 \int u\bar{v}$. Indeed, the inverse Fourier transform of $\hat{f}$ is given by

$$\mathcal{F}^{-1}(\hat{f}(\omega))(x) = \sum_{t_1=0}^{n-1} \sum_{t_2=0}^{n-1} \delta(x_1 - \frac{t_1}{n})\delta(x_2 - \frac{t_2}{n}) f[t_1,t_2],$$

while for the other

$$\mathcal{F}^{-1}(e^{-i\omega_1 \frac{k_1}{L_{1,j}}} e^{-i\omega_2 \frac{k_2}{L_{2,j}}} c(\omega)\hat{\varphi}^D_{j,\ell}(\omega))(x) = \frac{1}{(2\pi)^2}\varphi^\sharp_{j,\ell}(x_1 - \frac{k_1}{L_{1,j}}, x_2 - \frac{k_2}{L_{2,j}}).$$

The Parseval formula then gives (6.3). For the North and South quadrants, the proof is identical after swapping $L_{1,j}$ and $L_{2,j}$.

□

References

[1] C. R. Anderson and M. D. Dahleh. Rapid computation of the discrete Fourier transform. *SIAM J. Sci. Comput.* **17** (1996), 913–919.

[2] G. Beylkin, R. Coifman and V. Rokhlin. Fast wavelet transforms and numerical algorithms. *Comm. on Pure and Appl. Math.* **44** (1991), 141–183.

[3] G. Beylkin. On the fast Fourier transform of functions with singularities. *Appl. Comput. Harmon. Anal.*, **2-4** (1995), 363–381.

[4] E. J. Candès. Harmonic analysis of neural networks. *Applied and Computational Harmonic Analysis* **6** (1999), 197–218.

[5] E. J. Candès and L. Demanet, The curvelet representation of wave propagators is optimally sparse, *Comm. Pure Appl. Math.*, **58-11** (2005) 1472–1528.

[6] E. J. Candès and L. Demanet. Curvelets and fast wave equation solvers. Technical report, California Institute of Technology, 2005. In preparation.

[7] E. J. Candes and D. L. Donoho. Ridgelets: the key to higher-dimensional intermittency? *Phil. Trans. R. Soc. Lond. A.* **357** (1999), 2495–2509.

[8] E. J. Candès and D. L. Donoho. Curvelets – a surprisingly effective nonadaptive representation for objects with edges. In C. Rabut A. Cohen and L. L. Schumaker, editors, *Curves and Surfaces*, pages 105–120, Vanderbilt University Press, 2000. Nashville, TN.

[9] E. J. Candès and D. L. Donoho. Recovering edges in ill-posed inverse problems: Optimality of curvelet frames. *Ann. Statist.* **30** (2002), 784 –842.

[10] E. J. Candès and D. L. Donoho. New tight frames of curvelets and optimal representations of objects with piecewise-C^2 singularities. *Comm. on Pure and Appl. Math.* **57** (2004), 219–266.

[11] E. J. Candès and D. L. Donoho. Curvelets: new tools for limited-angle tomography, Manuscript, 2004.

[12] E. J. Candès and F. Guo. New multiscale transforms, minimum total variation synthesis: application to edge-preserving image reconstruction. *Sig. Process., special issue on Image and Video Coding Beyond Standards* **82** (2002), 1519–1543.

[13] R. H. Chan and M. K. Ng. Conjugate gradient methods for Toeplitz systems. *SIAM Rev.* **38** (1996), 427–482.

[14] M. N. Do. *Directional Multiresolution Image Representations.* PhD thesis, Swiss Federal Institute of Technology, Lausanne, November 2001.

[15] M. N. Do and M. Vetterli, Contourlets, in *Beyond Wavelets*, G. V. Welland ed., Academic Press, 2003.

[16] M. N. Do and M. Vetterli, The contourlet transform: an efficient directional multiresolution image representation, *IEEE Trans. Im. Proc.*, to appear, 2005.

[17] D. L. Donoho. Wedgelets: nearly-minimax estimation of edges. *Ann. Statist.* **27** (1999), 859–897.

[18] D. L. Donoho and M. R. Duncan. Digital Curvelet Transform: Strategy, Implementation, Experiments. Technical Report, Stanford University, 1999.

[19] D. L. Donoho and X. Huo. *Beamlets and Multiscale Image Analysis.* Springer, Lecture Notes in Computational Science and Engineering: Multiscale and Multiresolution Methods, 2001.

[20] H. Douma and M. V. de Hoop. Wave-character preserving prestack map migration using curvelets. Presentation at the *Society of Exploration Geophysicists*, Denver, CO, 2004.

[21] A. J. W. Duijndam and M. A. Schonewille, Nonuniform fast Fourier transform. *Geophys.* **64-2** (1999), 539–551.

[22] A. Dutt and V. Rokhlin. Fast Fourier transforms for nonequispaced data. *SIAM J. Sci. Stat. Comput.* **14-6** (1993), 1368–1393.

[23] A. Dutt and V. Rokhlin, Fast Fourier transforms for nonequispaced data II. *Appl. Comput. Harmon. Anal.*, **2** (1995), 85–100.

[24] A. G. Flesia, H. Hel-Or, A. Averbuch, E. J. Candès, R. R. Coifman and D. L. Donoho. Digital implementation of ridgelet packets, *Beyond Wavelets*, J. Stoeckler and G. V. Welland eds., Academic Press, 2003.

[25] G. Hennenfent and F. J. Herrmann. Seismic denoising with unstructured curvelets, *Comput. in Sci. Eng.*, to appear, 2006.

[26] F. J. Herrmann and E. Verschuur. Separation of primaries and multiples by non-linear estimation in the curvelet domain. In *EAGE 66th Conference & Exhibition Proceedings*, 2004.

[27] F. J. Herrmann, P. P. Moghaddam, C. C Stolk, Sparsity- and continuity-promoting seismic image recovery with curvelet frames, submitted, 2006.

[28] K. Guo, D. Labate, W. Lim, G. Weiss, and E. Wilson, Wavelets with Composite Dilations, *Electr. Res. Ann. AMS* **10** (2004), 78–87.

[29] K. Guo, D. Labate, W. Lim, G. Weiss, and E. Wilson, Wavelets with Composite Dilations and their MRA Properties, to appear in *Appl. Comput. Harmon. Anal.* (2005)

[30] D. Labate, W. Lim, G. Kutyniok and G. Weiss, "Sparse Multidimensional Representation using Shearlets, *SPIE conf. Wavelets XI*, San Diego, USA, 2005

[31] E. Le Pennec and S. Mallat. Sparse geometric image representations with bandelets. *IEEE Trans. Image Process.* **14** (2005), 423–438.

[32] R. Murenzi, *Ondelettes multidimensionelles et applications a l'analyse d'images,* Thèse, Université catholique de Louvain, Louvain-la-Neuve, 1990.

[33] F. Natterer. *The mathematics of computerized tomography.* B. G. Teubner; John Wiley & Sons, 1986.

[34] E. P. Simoncelli and W T Freeman. The Steerable Pyramid: A Flexible Architecture for Multi-Scale Derivative Computation. *IEEE Second Int'l Conf on Image Processing.* Washington DC, October 1995.

[35] E. P. Simoncelli, W. T. Freeman, E. H. Adelson, and D. J. Heeger. Shiftable multi-scale transforms [or what's wrong with orthonormal wavelets]. *IEEE Trans. Information Theory, Special Issue on Wavelets* **38** (1992), 587–607.

[36] H. A. Smith. A parametrix construction for wave equations with $C^{1,1}$ coefficients. *Ann. Inst. Fourier (Grenoble)* **48** (1998), 797–835.

[37] J. L. Starck, E. J. Candès, and D. L. Donoho. The curvelet transform for image denoising. *IEEE Trans. Im. Proc.*, **11-6** (2002), 670–684.

[38] J.-L. Starck, N. Aghanim and O. Forni, Detecting cosmological non-Gaussian signatures by multi-scale methods. *Astronomy and Astrophysics* **416** (2004), 9–17.

[39] J. L. Starck, M. Elad, and D. L. Donoho. Redundant multiscale transforms and their application for morphological component analysis. *Advances in Imaging and Electron Physics* **132** (2004).

[40] G. Strang. A proposal for Toeplitz matrix calculations. *Stud. Appl. Math.* **74** (1986), 171-176.

[41] P. Vandergheynst and J. F. Gobbers, Directional dyadic wavelet transforms: design and algorithms. *IEEE Trans. Im. Proc.* **11-4** (2002), 363–372.

[42] A. Zygmund. *Trigonometric series*, Cambridge University Press, 1964.