Appl. Comput. Harmon. Anal. 28 (2010) 171–188

Contents lists available at ScienceDirect

Applied and Computational Harmonic Analysis

www.elsevier.com/locate/acha

3D curvelet transforms and astronomical data restoration

A. Woiselle [a,b,*], J.-L. Starck [a], J. Fadili [c]

[a] CEA, IRFU, SEDI-SAP, Laboratoire Astrophysique des Interactions Multi-échelles (UMR 7158), CEA/DSM-CNRS-Universite Paris Diderot, Centre de Saclay, F-91191 Gif-sur-Yvette, France
[b] Sagem Defense Securite, 95101 Argenteuil Cedex, France
[c] GREYC CNRS UMR 6072, Image Processing Group, ENSICAEN 14050 Caen Cedex, France

ARTICLE INFO

Article history:
Received 29 April 2009
Revised 21 December 2009
Accepted 26 December 2009
Available online 6 January 2010
Communicated by Ginette Saracco

Keywords:
3D transform
Curvelet
Wavelet
Multi-resolution representations
Astrophysics
Galaxy distribution

ABSTRACT

This paper describes two new 3D curvelet decompositions, which are built in a way similar to the first generation of curvelets (Starck et al., 2002 [35]). The first one, called BeamCurvelet transform, is well designed for representing 1D filaments in a 3D space, while the second one, the RidCurvelet transform, is designed to analyze 2D surfaces. We show that these constructions can be useful for different applications such as filament detection, denoising or inpainting. Hence, they could lead to alternative approaches for analyzing 3D cosmological data sets, such as catalogs of galaxies, λCDM simulation or weak lensing data.

1. Introduction

Sparse representations such as wavelets or curvelets have been very successful for 2D image processing. Impressive results were obtained for many applications such as compression (see [8] for an example of surflet compression; the new image standard JPEG2000 is based on wavelets rather than DCT like JPEG), denoising [39,35,19], contrast enhancement [37], inpainting [16,18] or deconvolution [41,42]. Curvelets [35,5], bandelets [25] and contourlets [11] were designed to well represent edges in an image while wavelets are especially efficient for isotropic feature analysis.

In 3D, the separable wavelet transform (decimated, undecimated, or any other kind) and the discrete cosine transform are certainly the most known decompositions [31,10,9]. The DCT is mainly used in video compression, but has also been used in denoising [30]. A lot of effort has been made in the last five years to build sparse 3D data representations, which represent better geometrical features contained in the data. The 3D beamlet transform [15] and the 3D ridgelet transform [40] were respectively designed for 1D and 2D features detection. Video denoising using the ridgelet transform was proposed in [6]. The 3D fast curvelet transform [45] consists in paving the Fourier domain with angular wedges in dyadic concentric squares using the parabolic scaling law to fix the number of angles depending on the scale, and has atoms designed for representing surfaces in 3D. The surflet transform [7] — a d-dimensional extension of the 2D wedgelets [13,27] — has recently been studied for compression purposes [8]. Surflets are an adaptive transform estimating each cube of a quad-tree decomposition of the data by two regions of constant value separated by a polynomial surface. Another possible representation uses the surfacelets developed by Do and Lu [20]. It relies on the combination of a Laplacian pyramid and a d-dimensional directional

* Corresponding author at: CEA, IRFU, SEDI-SAP, Laboratoire Astrophysique des Interactions Multi-échelles (UMR 7158), CEA/DSM-CNRS-Universite Paris Diderot, Centre de Saclay, F-91191 Gif-sur-Yvette, France.
E-mail addresses: arnaud.woiselle@cea.fr (A. Woiselle), jsarck@cea.fr (J.-L. Starck), jallal.fadili@greyc-ens.caen.fr (J. Fadili).

1063-5203/$ – see front matter
doi:10.1016/j.acha.2009.12.003

filter bank. Surfacelets produce a tiling of the Fourier space in angular wedges in a way close to the curvelet transform, and can be interpreted as a 3D adaptation of the 2D contourlet transform. This transformation has recently also been applied to video denoising [21].

1.1. 3D multiscale transforms and cosmological data set

Different statistical measures have been used in the cosmological literature to quantitatively describe the cosmic texture [23], i.e. the large-scale structure of the universe showing intricate patterns with filaments, clusters, and sheet-like arrangements of galaxies encompassing large nearly empty regions, the so-called voids. Wavelets have been used for many years [17,34,23], and it has been shown that denoising the galaxy distribution using wavelet instead of the standard Gaussian filtering, allows us to better preserve structure at different scales, and therefore better constrain our cosmological models [22].

Noise is also a problem of major concern for N-body simulations of structure formation in the early Universe and it has been shown that using wavelets for removing noise from N-body simulations is equivalent to simulations with two orders of magnitude more particles [28,29].

Finally, 3D wavelets, ridgelet and beamlet were also used in order to extract statistical information from galaxy catalogs [40] and compare our data set to simulations obtained from different cosmological models.

1.2. Why new transforms?

These 3D transforms all aim at representing the data using a minimal number of active coefficients, and by construction are better adapted to capture a specific kind of pattern. For the wavelet transform the pattern is smooth and isotropic, while for the DCT it is oscillating in all directions. All previously mentioned 3D transforms, except the beamlet transform, use plate-like functions, useful to represent surfaces in a 3D volume. The beamlet is therefore the only existing geometric decomposition allowing an efficient detection of filaments in 3D. Two relatively different implementations have been proposed. One [15] suffers the lack of any reconstruction algorithm, and the other one [40] is only optimal for the detection of filaments with a specific filament size. This has motivated the design of the FABT transform in biology [2], but this decomposition is also limited, since its optimality is only for filament of a given width. None of these transforms presents the nice scaling properties similar to those of 2D curvelet transform.

1.3. This paper

We propose in this paper a new transform, the BeamCurvelet transform, which is well adapted for the detection of filaments of different sizes and widths. A minor variation in its construction leads to another transform with plate-like elements, the RidCurvelet transform. Following the notations in [14], a more standard name for these representations would be "local-k plane ridgelet bases in n-D: LRB(k, n)", thus leading respectively to LRB$(1, 3)$ and LRB$(2, 3)$. Both constructions have interesting scaling properties, and offer exact reconstruction. They can therefore be used for different applications such as denoising or inpainting.

This paper is organized as follows: In Sections 2 and 3, we show how we can extend the 2D curvelet transform to 3D, leading to two new decompositions, the BeamCurvelet and the RidCurvelet. In Section 4, we investigate different approaches for BeamCurvelet/RidCurvelet denoising. The last section presents applications how these two transforms can be used for inpainting.

2. The BeamCurvelet transform

2.1. The 2D curvelet transform

In order to understand our construction for 3D curvelets, we first recall the simpler but very similar 2D case.

The first generation curvelets [35,4] were built using the isotropic undecimated wavelet transform [36] and the ridgelet transform [3].

The ridgelet transform is useful for representing global lines crossing an image on its full length the same way the wavelets represent isolated isotropic singularities and shapes. This property is obtained by the Radon transform, which transforms lines into points. A ridgelet function is indexed by a scale parameter a, a position b and an orientation θ. Let $\gamma = (a, b, \theta) \in \mathbb{R}_+^* \times \mathbb{R} \times [0, 2\pi[$. Given a wavelet function ψ, we define a ridgelet ψ_γ by

$$\psi_\gamma = a^{-1/2}\psi\big((x_1 \cos\theta + x_2 \sin\theta - b)/a\big).$$

The ridgelet transform is implemented as a line extraction in Fourier domain, using the projection-slice theorem. The process is shown in Fig. 1. The main drawback of this transform is that lines span the whole image and thus aren't well localized in space. The idea was then to make it local and multiscale by applying it blockwise on a multiscale isotropic wavelet transform. The key property is a law forcing the size of the blocks from one scale to the next, following a *parabolic scaling* which states that the number of blocks to get in a scale is downsized by a factor four each other scale (the size of the blocks is doubled each other scale on an isotropic undecimated wavelet transform, and divided by two on a decimated

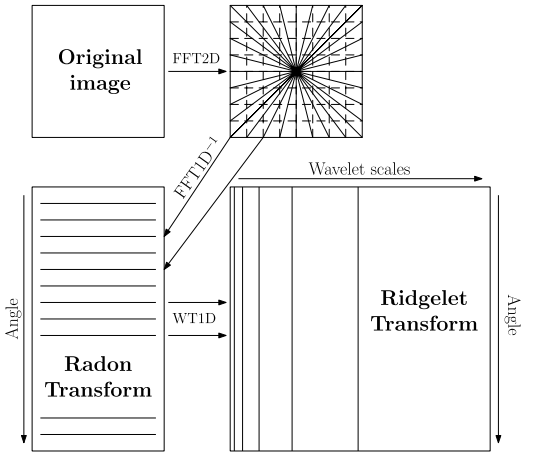

Fig. 1. Scheme showing the main steps of the ridgelet transform in 2D: lines passing through the origin are extracted from the Fourier transform of the image; a wavelet transform is applied on each of these lines.

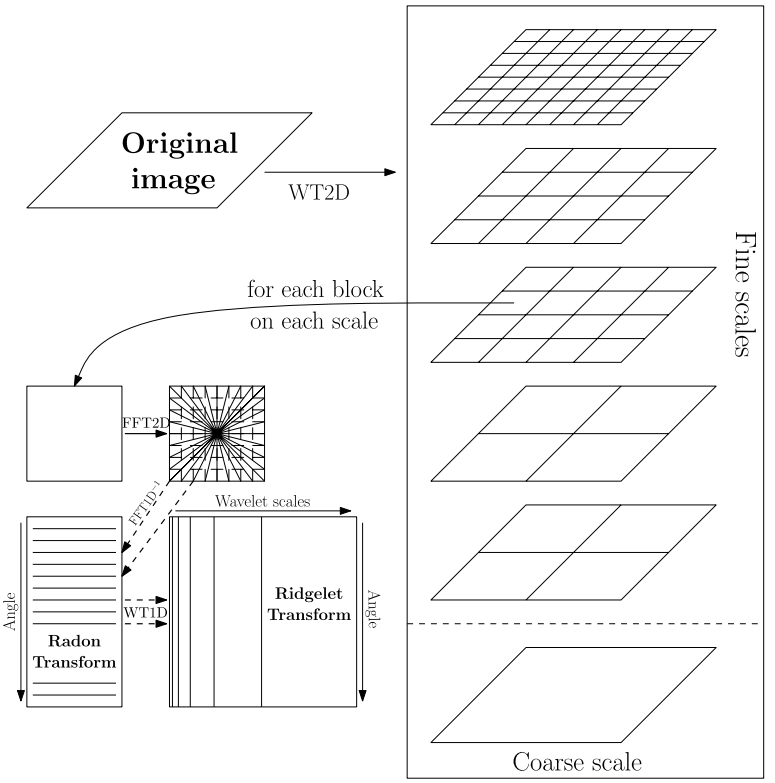

Fig. 2. Scheme showing the main steps of the curvelet-99 transform: the ridgelet transform is applied to each block of each scale of an isotropic 2D wavelet transform.

one). Fig. 2 shows the global process described above. The ridgelet transform is implemented through a Radon transform in Fourier domain, followed by a one-dimensional wavelet transform.

As in two dimensions, the 3D first generation curvelet transforms we develop in this paper are based on the Radon transform applied to localized blocks of a given size of a 3D spectral decomposition of the data. There are two ways of extending the Radon transform in three dimensions, which lead to the two transforms described below. The first one is obtained by projecting along 2D planes (3D partial Radon transform), which leads to the BeamCurvelets, and the second one by projecting *only* along 1D lines (3D Radon transform), which leads to the RidCurvelets.

2.2. The 3D continuous BeamCurvelet transform

In order to separate the signal into spectral bands, we use a filter-bank. Let $\mathbb{N}^*$ be the set of strictly positive integers. Given a smooth function $\psi \in L^2(\mathbb{R}^3)$: $\forall s \in \mathbb{N}^*$, $\psi_{2s} = 2^{6s}\psi(2^{2s}\cdot)$ extracting the frequencies around $|\xi| \in [2^{2s}, 2^{2s+2}]$, and a low-pass filter ψ_0 for $|\xi| \leqslant 1$. We get a partition of unity:

$$\left|\hat{\psi}_0(\xi)\right|^2 + \sum_{s>0}\left|\hat{\psi}_{2s}(\xi)\right|^2 = 1. \tag{1}$$

Let $P_0 f = \psi_0 * f$ and $\Delta_s f = \psi_{2s} * f$, where $*$ is the convolution product. We can represent any signal f as $(P_0 f, \Delta_1 f, \Delta_2 f, \ldots)$.

We tile the spatial domain of each scale s with a set $\mathcal{Q}_s$ of regions Q of size 2^s:

$$Q = Q(s, k_1, k_2, k_3) = \left[\frac{k_1}{2^s}, \frac{k_1+1}{2^s}\right[\times \left[\frac{k_2}{2^s}, \frac{k_2+1}{2^s}\right[\times \left[\frac{k_3}{2^s}, \frac{k_3+1}{2^s}\right[\subset [0,1]^3 \tag{2}$$

with smooth windows w_Q localized near Q, and verifying $\sum_{Q\in\mathcal{Q}_s} w_Q^2 = 1$, with

$$\mathcal{Q}_s = \left\{ Q(s, k_1, k_2, k_3) \;\middle|\; (k_1, k_2, k_3) \in \left[0, 2^s\right)^3 \right\}. \tag{3}$$

Each element of the frequency-space $w_Q \Delta_s$ is transported to $[0,1]^3$ by the transport operator $T_Q : L^2(Q) \to L^2([0,1]^3)$ applied to $f' = w_Q \Delta_s f$

$$\begin{aligned} &T_Q : L^2(Q) \to L^2\left([0,1]^3\right), \\ &\left(T_Q f'\right)(x_1, x_2, x_3) = 2^{-s} f'\left(\frac{k_1 + x_1}{2^s}, \frac{k_2 + x_2}{2^s}, \frac{k_3 + x_3}{2^s}\right). \end{aligned} \tag{4}$$

For each scale s, we have a space-frequency tiling operator g_Q, the output of which lives on $[0,1]^3$

$$g_Q = T_Q w_Q \Delta_s. \tag{5}$$

We can apply a 3D beamlet transform [15,12] on each block of each scale, by projecting on the beamlet functions:

$$\begin{aligned} \beta_{\sigma,\kappa_1,\kappa_2,\theta_1,\theta_2}(x_1, x_2, x_3) = \sigma^{-1/2}\phi\big(&(-x_1 \sin\theta_1 + x_2 \cos\theta_1 + \kappa_1)/\sigma, \\ &(x_1 \cos\theta_1 \cos\theta_2 + x_2 \sin\theta_1 \cos\theta_2 - x_3 \sin\theta_2 + \kappa_2)/\sigma\big), \end{aligned} \tag{6}$$

where σ is the beamlet scale parameter, (θ_1, θ_2) the orientation parameter and (κ_1, κ_2) the location parameter, which is two-dimensional because the beamlet transform integrates the data over one dimension through the partial Radon transform (see Section 2.3). $\phi \in L^2(\mathbb{R}^3)$ is a smooth function satisfying the following admissibility condition

$$\sum_{s\in\mathbb{Z}} \phi^2\left(2^s u\right) = 1, \quad \forall u \in \mathbb{R}^2. \tag{7}$$

Finally, the BeamCurvelet transform of a 3D function f is

$$\mathcal{BC} f = \left\{ \left\langle (T_Q w_Q \Delta_s) f, \beta_{\sigma,\kappa_1,\kappa_2,\theta_1,\theta_2} \right\rangle : s \in \mathbb{N}^*, Q \in \mathcal{Q}_s \right\}. \tag{8}$$

As we can see, a BeamCurvelet function is parametrized in scale (s, σ), position (Q, κ_1, κ_2), and orientation (θ_1, θ_2). The following sections describe the discretization and the effective implementation of such a transform.

2.3. Discretization

For convenience, and as opposed to the continuous notations, the scales are now numbered from 0 to J, from the finest to the coarsest. As seen in the continuous formulation, the transform operates in four main steps.

1. First the frequency decomposition is obtained by applying a 3D wavelet transform on the data with a wavelet compactly supported in Fourier space like the pyramidal Meyer wavelets with low redundancy [38], or using the 3D isotropic *à trou* wavelets.
2. Each wavelet scale is then decomposed in small cubes of a size following the parabolic scaling law, forcing the block size B_s with the scale size N_s according to the formula

$$\frac{B_s}{N_s} = 2^{s/2}\frac{B_0}{N_0}, \tag{9}$$

where N_0 and B_0 are the finest scale's dimension and block size.

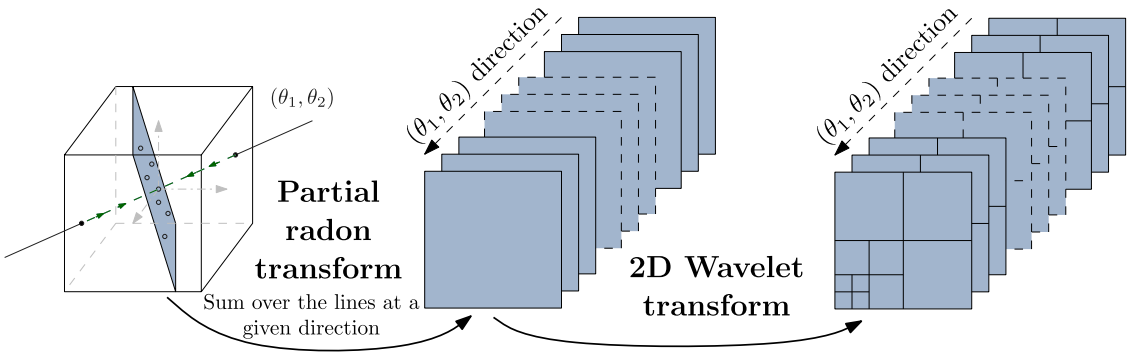

Fig. 3. Schematic view of a 3D beamlet transform. At a given direction, sum over the (θ_1, θ_2) line to get a $\circ$ point. Repeat over all its parallels to get the dark plane and apply a 2D wavelet transform within that plane. Repeat for all the directions to get the 3D beamlet transform. See the text (Section 3) for a detailed explanation and implementation clues.

Algorithm 1: The BeamCurvelet transform.

```
Data: A data cube x and an initial block size B
Result: BeamCurvelet transform of x
begin
    Apply a 3D isotropic wavelet transform
    for all scales from the finest to the second coarsest do
        Partition the scale into small cubes of size B
        for each block do
            Apply a 3D FFT
            Extract planes passing through the origin at every angle (θ1, θ2)
            for each plane (θ1, θ2) do
                apply an inverse 2D FFT
                apply a 2D wavelet transform to get the BeamCurvelet coefficients
        if the scale number is even then
            according to the parabolic scaling:
            B = 2B (in the undecimated wavelet case)
            B = B/2 (in the pyramidal wavelet case)
end
```

3. Then, we apply a partial 3D Radon transform on each block of each scale. This is accomplished by integrating the blocks along lines at every direction and position. For a fixed direction (θ_1, θ_2), the summation gives us a plane. Each point on this plane represents a line in the original cube. We obtain projections of the blocks on planes passing through the origin at every possible angle.
4. At last, we apply a two-dimensional wavelet transform on each partial Radon plane.

Steps 3 and 4 represent the beamlet transform of the blocks. The 3D beamlet atoms aim at representing filaments crossing the whole 3D space. They are constant along a line and oscillate like ϕ in the radial direction. Arranged blockwise on a 3D isotropic wavelet transform, and following the parabolic scaling, we obtain the BeamCurvelet transform.

Fig. 3 summarizes the beamlet transform, and Fig. 4 the global BeamCurvelet transform.

2.4. Algorithm summary

As for the 2D curvelets, the 3D BeamCurvelet transform is implemented effectively in the Fourier domain. Indeed, the integration along the lines (3D partial Radon transform) becomes a simple plane extraction in Fourier space, using the d-dimensional projection-slice theorem, which states that the Fourier transform of the projection of a d-dimensional function onto an m-dimensional linear submanifold is equal to an m-dimensional slice of the d-dimensional Fourier transform of that function through the origin in the Fourier space which is parallel to the projection submanifold. In our case, $d = 3$ and $m = 2$. Algorithm 1 summarizes the whole process.

2.5. Properties

As a composition of invertible operators, the BeamCurvelet transform is invertible. As the wavelet and Radon transform are both tight frames, so is the BeamCurvelet transform.

Given a Cube of size $N \times N \times N$, a cubic block of length B_s at scale s, and $J + 1$ scales, the redundancy can be calculated as follows:

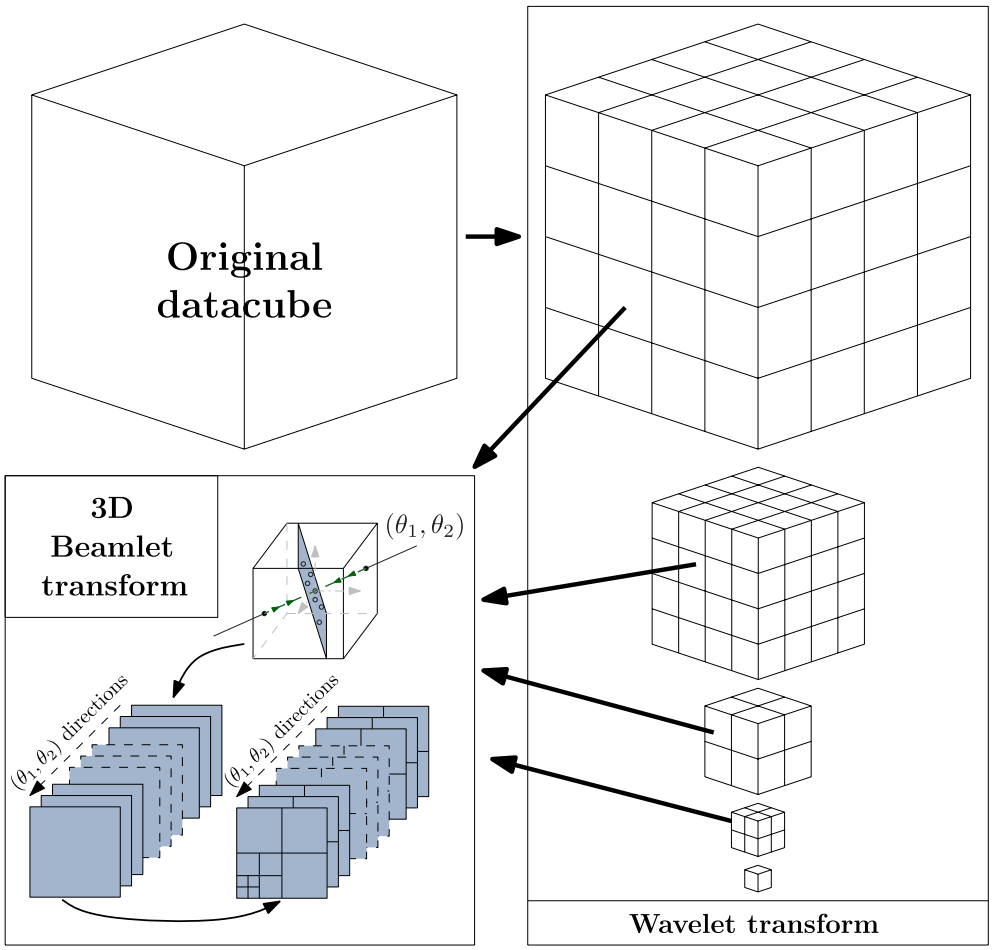

Fig. 4. Global flow graph of a 3D BeamCurvelet transform.

According to the parabolic scaling, $\forall s > 0$: $B_s/N_s = 2^{s/2}B_0/N_0$. The redundancy induced by the 3D wavelet transform is

$$R_w = \frac{1}{N^3}\sum_{s=0}^{J} N_s^3, \tag{10}$$

with $N_s = 2^{-s}N$ for pyramidal Meyer wavelets (the case used on experiments described in Sections 4 and 5), and thus $B_s = 2^{-s/2}B_0$ according to the parabolic scaling (see Eq. (9)).

The partial Radon transform of a cube of size B_s^3 has a size $3B_s^2 \times B_s^2$ to which we apply 2D decimated orthogonal wavelets with no redundancy. There are $(\rho N_s/B_s)^3$ blocks in each scale because of the overlap factor ($\rho \in [1, 2]$) in each direction. So the complete redundancy of the transform using the Meyer wavelets is

$$R = \frac{1}{N^3}\sum_{s=0}^{J-1}\left(\rho\frac{N_s}{B_s}\right)^3 3B_s^4 + \frac{N_J^3}{N^3} = 3\rho^3\sum_{i=0}^{J-1} B_s 2^{-3s} + 2^{-3J} \tag{11}$$

$$= 3\rho^3 B_0 \sum_{s=0}^{J-1} 2^{-7s/2} + 2^{-3J} \tag{12}$$

$$= \mathcal{O}\big(3\rho^3 B_0\big) \quad \text{when } J \to \infty, \tag{13}$$

$$R(J=1) = 3\rho^3 B_0 + \frac{1}{8}, \tag{14}$$

$$R(J=\infty) \approx 3.4\rho^3 B_0. \tag{15}$$

For a typical block size $B_0 = 17$, we get for $J \in [1, \infty[$:

Algorithm 2: The inverse BeamCurvelet transform.

Data: An initial block size B, and the BeamCurvelet coefficients: series of wavelet-space planes indexed by a scale, angles (θ_1, θ_2), and a 3D position $(\mathcal{B}_x, \mathcal{B}_y, \mathcal{B}_z)$
Result: The reconstructed data cube
begin
  for *all scales from the finest to the second coarsest* **do**
    Create a 3D cube the size of the current scale (according to the 3D wavelets used in the forward transform)
    for *each block position* $(\mathcal{B}_x, \mathcal{B}_y, \mathcal{B}_z)$ **do**
      Create a block $\mathcal{B}$ of size $B \times B \times B$
      for *each plane* (θ_1, θ_2) *indexed with this position* **do**
        – Apply an inverse 2D wavelet transform
        – Apply a 2D FFT
        – Put the obtained Fourier plane to the block, such that the plane passes through the origin of the block with normal angle (θ_1, θ_2)
      – Apply a 3D IFFT
      – Add the block to the wavelet scale at the position $(\mathcal{B}_x, \mathcal{B}_y, \mathcal{B}_z)$, using a weighted function if overlapping is involved
    if *the scale number is even* **then**
      according to the parabolic scaling:
      $B = 2B$ (in the undecimated wavelet case)
      $B = B/2$ (in the pyramidal wavelet case)
  Apply a 3D inverse isotropic wavelet transform
end

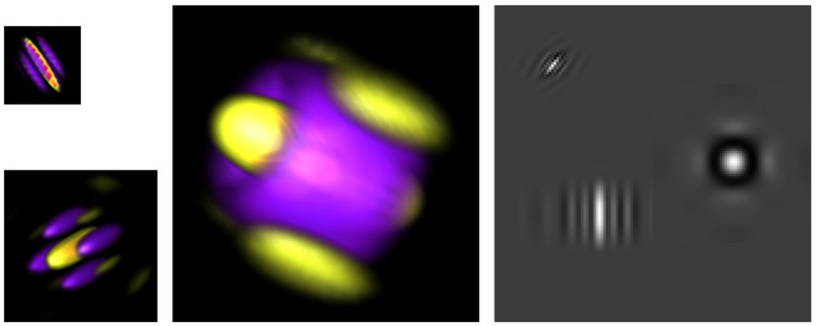

Fig. 5. Examples of a BeamCurvelet atoms at different scales and orientations. These are 3D density plots: the values near zero are transparent, and the opacity grows with the absolute value of the voxels. Positive values are red/yellow, and negative values are blue/purple. The right map is a slice of a cube containing these three atoms in the same position as on the left. The top left atom has an arbitrary direction, the bottom left is in the slice, and the right one is normal to the slice. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$$R \in [51.125, 57.8[\quad \text{without overlapping}, \tag{16}$$

$$R \in [408.125, 462.4[\quad \text{with 50\% overlapping } (\rho = 2). \tag{17}$$

2.6. Inverse BeamCurvelet transform

Because all its components are invertible, the BeamCurvelet transform is invertible and the reconstruction error is comparable to machine precision. Algorithm 2 details the reconstruction steps.

An example of a 3D BeamCurvelet atom is represented in Fig. 5. The BeamCurvelet atom is a collection of straight smooth segments well localized in space. Across the transverse plane, the BeamCurvelets exhibit a wavelet-like oscillating behavior.

3. The 3D RidCurvelet transform

3.1. The continuous transform

As referred to in Section 2.2, the second extension of the curvelet transform in 3D is obtained by using the 3D ridgelet transform [3] instead of the beamlets. A three-dimensional ridge function is given by:

$$\rho_{\sigma,\kappa,\theta_1,\theta_2}(x_1, x_2, x_3) = \sigma^{-1/2}\phi\left(\frac{1}{\sigma}(x_1 \cos\theta_1 \cos\theta_2 + x_2 \sin\theta_1 \cos\theta_2 + x_3 \sin\theta_2 - \kappa)\right), \tag{18}$$

where σ and κ are respectively the scale and position parameters, and $\phi \in L^2(\mathbb{R})$ satisfies Eq. (7). The global RidCurvelet transform of a 3D function f is then

$$\mathcal{RC} f = \left\{\left\langle (T_Q\, w_Q\, \Delta_s) f, \rho_{\sigma,\kappa,\theta_1,\theta_2}\right\rangle \colon s \in \mathbb{N}^*,\ Q \in \mathcal{Q}_s\right\}. \tag{19}$$

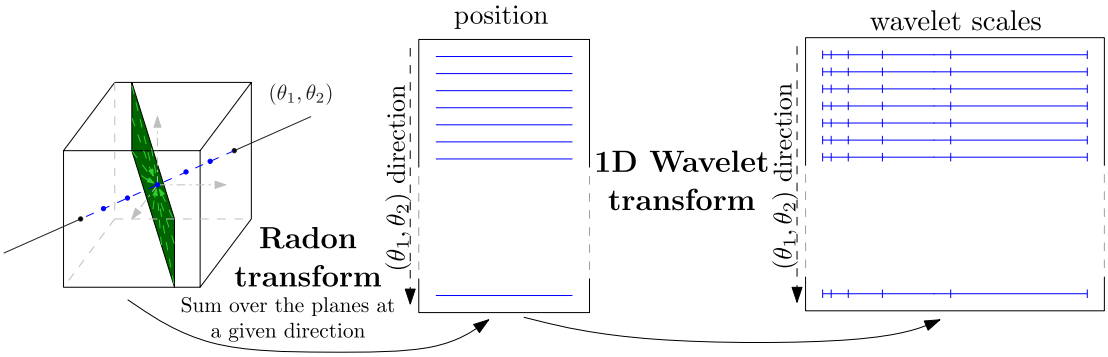

Fig. 6. Overview of the 3D ridgelet transform. At a given direction, sum over the normal plane to get a • point. Repeat over all its parallels to get the (θ_1, θ_2) line and apply a 1D wavelet transform on it. Repeat for all the directions to get the 3D Ridgelet transform. See the text (Section 3) for a detailed explanation and implementation clues.

3.2. Discretization

The discretization is made the same way, the sums over lines becoming sums over the planes of normal direction (θ_1, θ_2), which gives us a line for each direction. The 3D ridge function is useful for representing planes in a 3D space. It is constant along a plane and oscillates like ϕ in the normal direction. The main steps of the ridgelet transform are depicted in Fig. 6.

3.3. Implementation

The RidCurvelet transform is also implemented in Fourier domain, the integration along the planes becoming a line extraction in the Fourier domain. The overall process is shown in Fig. 7, and Algorithm 3 summarizes the implementation.

3.4. Properties

The RidCurvelet transform forms a tight frame. Additionally, given a 3D cube of size $N \times N \times N$, a block of size-length B_s at scale s, and $J+1$ scales, the redundancy is calculated as follows:

The Radon transform of a cube of size B_s^3 has a size $3B_s^2 \times B_s$, to which we apply a pyramidal 1D wavelet of redundancy 2, for a total size of $3B_s^2 \times 2B_s = 6B_s^3$. There are $(\rho N_s/B_s)^3$ blocks in each scale because of the overlap factor ($\rho \in [1, 2]$) in each direction. Therefore, the complete redundancy of the transform using many scales of 3D Meyer wavelets is

$$R = \sum_{s=0}^{J-1} 6B_s^3 \left(\rho \frac{N_s}{B_s}\right)^3 + 2^{-3J} = 6\rho^3 \sum_{s=0}^{J-1} 2^{-3s} + 2^{-3J}, \tag{20}$$

$$R = \mathcal{O}\left(6\rho^3\right) \quad \text{when } J \to \infty, \tag{21}$$

$$R(J=1) = 6\rho^3 + 1/8, \tag{22}$$

$$R(J=\infty) \approx 6.86\rho^3. \tag{23}$$

3.5. Inverse RidCurvelet transform

The RidCurvelet transform is invertible and the reconstruction error is comparable to machine precision. Algorithm 4 details the reconstruction steps.

An example of a 3D RidCurvelet atom is represented in Fig. 8. The RidCurvelet atom is composed of planes with values oscillating like a wavelet in the normal direction, and well localized due to the smooth function used to extract blocks on each wavelet scale.

4. Denoising

4.1. Introduction

In sparse representations, the simplest denoising methods are performed by a simple thresholding of the discrete curvelet coefficients. The threshold level is usually taken as three times the noise standard deviation, such that for an additive Gaussian noise, the thresholding operator kills all noise coefficients except a small percentage, keeping the big coefficients containing information. The threshold we use is often a simple $\kappa\sigma$, with $\kappa \in [3, 4]$, which corresponds respectively to 0.27%

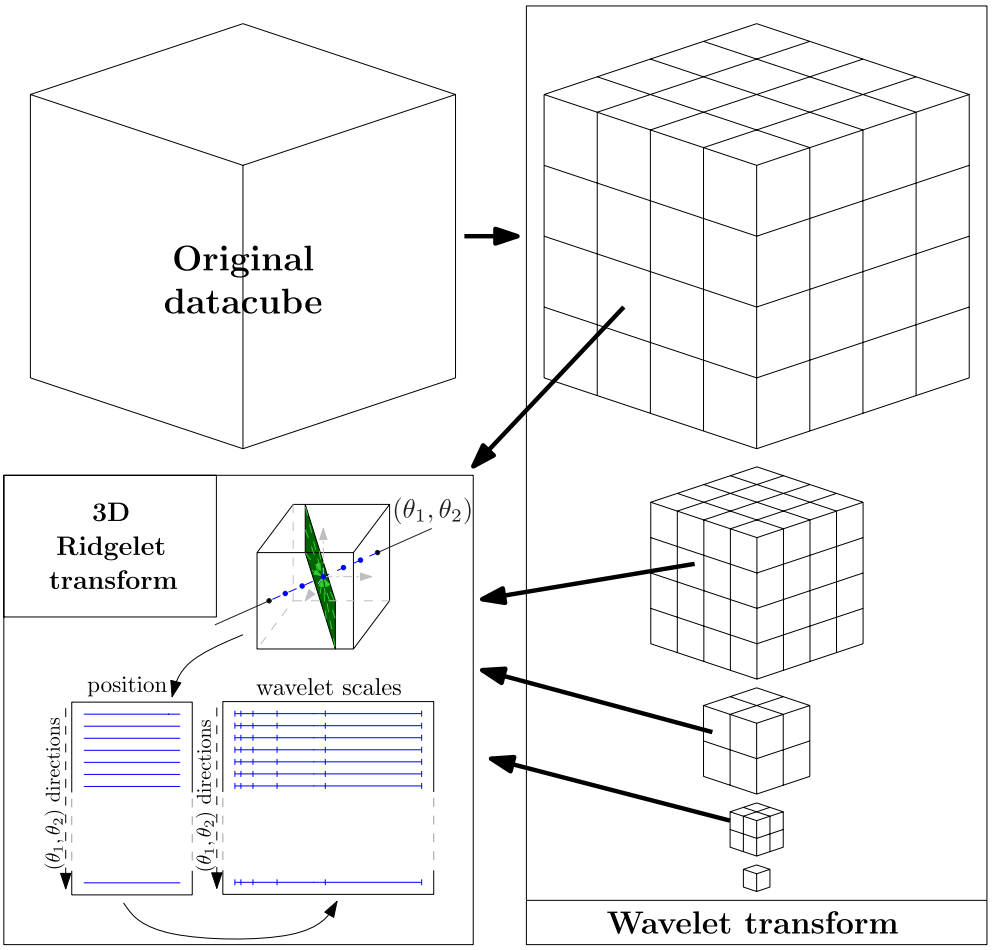

Fig. 7. Global flow graph of a 3D RidCurvelet transform.

Algorithm 3: The RidCurvelet transform.

Data: A data cube x and an initial block size B
Result: RidCurvelet transform of x
begin
  Apply a 3D isotropic wavelet transform
  for *all scales from the finest to the second coarsest* **do**
    Cut the scale into small cubes of size B
    for *each block* **do**
      Apply a 3D FFT
      Extract lines passing through the origin at every angle (θ_1, θ_2)
      for *each line* (θ_1, θ_2) **do**
        apply an inverse 1D FFT
        apply a 1D wavelet transform to get the RidCurvelet coefficients
    if *the scale number is even* **then**
      according to the parabolic scaling:
      $B = 2B$ (in the undecimated wavelet case)
      $B = B/2$ (in the pyramidal wavelet case)
end

and $6.3 \cdot 10^{-5}$ false detections. Sometimes we use a higher κ for the finest scale [35]. Other methods exist, that estimate automatically the threshold to use in each band like the false discovery rate (see [1,24]). The correlation between neighbor coefficients intra-band and/or inter-band may also be taken into account (see [33,32]). In this paper, in order to evaluate the different transforms, we use a $\kappa\sigma$ hard thresholding in our experiments.

Algorithm 4: The inverse RidCurvelet transform.

Data: An initial block size B, and the RidCurvelet coefficients: series of wavelet-space lines indexed by a scale, angles (θ_1, θ_2), and a 3D position $(\mathcal{B}_x, \mathcal{B}_y, \mathcal{B}_z)$
Result: The reconstructed data cube
begin
  for *all scales from the finest to the second coarsest* **do**
    Create a 3D cube the size of the current scale (according to the 3D wavelets used in the forward transform)
    for *each block position* $(\mathcal{B}_x, \mathcal{B}_y, \mathcal{B}_z)$ **do**
      Create a block $\mathcal{B}$ of size $B \times B \times B$
      for *each line* (θ_1, θ_2) *indexed with this position* **do**
        – Apply an inverse 1D wavelet transform
        – Apply a 1D FFT
        – Put the obtained Fourier line to the block, such that the line passes through the origin of the block with the angle (θ_1, θ_2)
      – Apply a 3D IFFT
      – Add the block to the wavelet scale at the position $(\mathcal{B}_x, \mathcal{B}_y, \mathcal{B}_z)$, using a weighted function if overlapping is involved
    if *the scale number is even* **then**
      according to the parabolic scaling:
      $B = 2B$ (in the undecimated wavelet case)
      $B = B/2$ (in the pyramidal wavelet case)
  Apply a 3D inverse isotropic wavelet transform
end

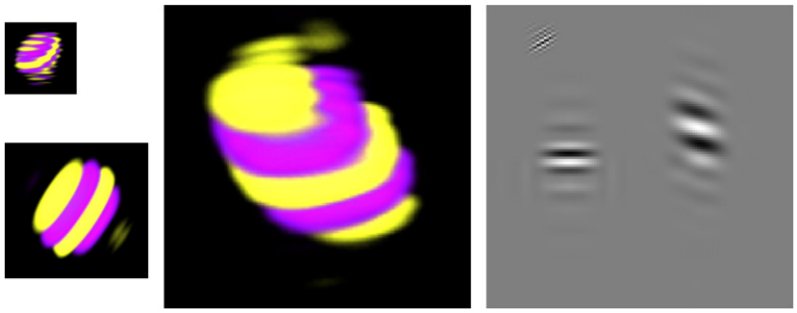

Fig. 8. Examples of RidCurvelet atoms at different scales and orientation. The rendering is similar to that of Fig. 5. The right plot is a slice from a cube containing the three atoms shown here.

4.2. Algorithm

Due to the high redundancy of the proposed transforms, and because of the huge size of 3D data, the transforms were implemented in a filtering-oriented way, in order to spare the machine resources, and enable easy multi-threading. The two new curvelet transforms operate blockwise, and when there is no intra-block correlation taken into account in the denoising process, each block can be treated independently and on a different processor. Therefore, we never have the full transform in memory, only the wavelet transform of the data, and the curvelet transform of one block (times the number of CPUs if working on a cluster). Algorithm 5 summarizes the denoising process. Using this algorithm, the memory used to filter a cube of any size using the curvelet transforms is about twice that of the isotropic Meyer wavelet transform, as the size of one block is usually negligible compared to the size of the data. As an example, a data cube of size $128 \times 128 \times 128$ (8 MB with 32 bit floating points) transformed with the BeamCurvelet and a block size of 17 with full 50% overlapping requires 3.3 GB of memory, while it runs under 70 MB with the low memory filtering algorithm. The transform and reconstruction with the RidCurvelet (BeamCurvelet) transform take respectively about 25 (260) and 35 (460) seconds without overlapping on a laptop (using a single core at 2.16 GHz). With full overlapping, it takes about six times more time, but as the code is parallelized with OpenMP, it can use all the CPUs of the computer to lower the execution time dramatically.

4.3. Experiments

We have performed a few denoising experiments to emphasize the applicability domain and specificity of each transform.

4.3.1. Toy model: Structure detection

First of all, in order to have a quantitative indicator of the efficiency of the two curvelet transforms to approximate and detect specific structures, we transform two cubes, one containing a plane and the second a filament. Table 1 contains the maximum coefficients of the transformed cubes, showing the level of detection of each structure in a given space. As we can see, the RidCurvelet has the greatest coefficient for the plane and the BeamCurvelet for the filament. In many applications (denoising, inpainting, morphological component analysis, etc.), the process is a thresholding of the coefficients at a level

Algorithm 5: Curvelet denoising algorithm.

Data: The observed data, the noise level σ and a threshold level $\kappa\sigma$.
Result: The denoised data. *// Memory usage*
begin
  Apply a 3D isotropic Meyer wavelet transform — 1 *Wavelet transform*
  Duplicate it and set it to zero — +1 *Wavelet transform*
  Separate the following loop on multiple CPUs
  for *each block* $(s, \mathcal{B}_x, \mathcal{B}_y, \mathcal{B}_z)$ *of all scales except the coarsest* **do**
    Extract the block — +1 *block*
    Apply a 3D beamlet or ridgelet transform to the block — +1 *transformed block*
    Hard threshold the coefficients at $\kappa\sigma$
    Apply a 3D beamlet or ridgelet inverse transform to the block
    Put the block into the new wavelet scale, or add it with a weighing window if overlapping is involved
  Apply an inverse 3D wavelet transform
end

Table 1
Maximum value of the coefficients of a transformed cube containing either a plane or a filament, using a given representation.

	RidCurvelets	BeamCurvelets	Undecimated wavelets	Decimated wavelets
Plane	**11.78**	8.11	5.06	3.13
Filament	1.02	**2.22**	1.30	1.16

fixed by the parameters of the algorithm and the noise level. Having many coefficients above a given level (or much energy), is an indication of effectiveness.

In order to see the gain when using one transform rather than another, we threshold the RidCurvelet transform of the cube containing the plane at the maximum BeamCurvelet detection level (which is also far above the wavelet one). This means that, had we used another transform besides the RidCurvelet at the same detection level, we would not have detected anything. The reconstruction is shown in Fig. 9 (top row). We have also performed the same experiment with the BeamCurvelet transform of the cube containing the filament with a threshold equal to the level where the three other tested transforms don't detect anything. As expected, there is much information kept by the most *adapted* transform, even at a very high level of thresholding.

4.3.2. Structure denoising

Another way to see the power of each transform when associated to the right structures is to denoise a synthetic cube containing plane- and filament-like structures. Fig. 10 shows a cut and a projection of the test cube containing parts of spherical shells and a spring-shaped filament. We denoise this cube using wavelets, RidCurvelets and BeamCurvelets. As shown in Fig. 11, the RidCurvelets denoise correctly the shells but poorly the filament, the BeamCurvelets restore the helix more properly while slightly underperforming for the shells, and wavelets are poor on the shell and give a dotted result and misses the faint parts of both structures. The PSNRs obtained with each transform are reported in Table 2. Here, the curvelet transforms did very well for a given kind of features, and the wavelets were better on the signal power. In Starck et al. [39], in the framework of 3D image denoising, it was advocated to combine several transforms in order to benefit from the advantages of each of them.

4.3.3. Combined filtering

Let x be the unknown data, and z a white Gaussian noise map of unitary variance. We observe $y = x + \sigma z$, where σ^2 is the noise variance. A combined filtering algorithm has been developed by Starck et al. [39], estimating x as $\tilde{x}$ as described below, using K dictionaries Φ_k, each having a fast transform and reconstruction. We denote by Φ_k^T the forward transform. Let $\alpha_k = (\alpha_{k,i})$ be the coefficient list indexed by i obtained by applying the k-th transform $\alpha_k = \Phi_k^T y$. Let Ω_k^y be the list of coefficients of y in Φ_k which have an absolute value greater than $\kappa\sigma$: $\Omega_k^y = \{i\colon\ |\alpha_{k,i}| > \kappa\sigma\}$, $\kappa\sigma$ being the final threshold level (usually 3σ). The combined filtering method consists in minimizing

$$\min_{\tilde{x}} \sum_{k=1}^{K} \left\| \Phi_k^T \tilde{x} \right\|_{\ell_1} \quad \text{s.t. } \forall k \in [1, K]\colon\ \left\| \left(\Phi_k^T \tilde{x} - \Phi_k^T y \right)_{\Omega_k} \right\|_{\ell_\infty} \leqslant \frac{\sigma}{2}, \tag{24}$$

where $(\alpha_k)_{\Omega_k}$ denotes the restriction to atoms in Ω_k. The algorithm proposed in [39] solves this minimization problem by iteratively transforming the solution in each dictionary, comparing the coefficients with the noisy data's coefficients, and soft-thresholding with a decreasing level. The main drawback of this algorithm is that it requires to have in memory all coefficients of all transforms during the iterations in order to evaluate the right-hand part of the functional. With the size of 3D data and the redundancy of the proposed transforms, this requires a huge memory size. Hence, using this method, we lose the advantages of the low-memory implementation that make our new transforms usable on a laptop computer with very low memory consumption. Therefore, we propose an alternative approach for the combined filtering.

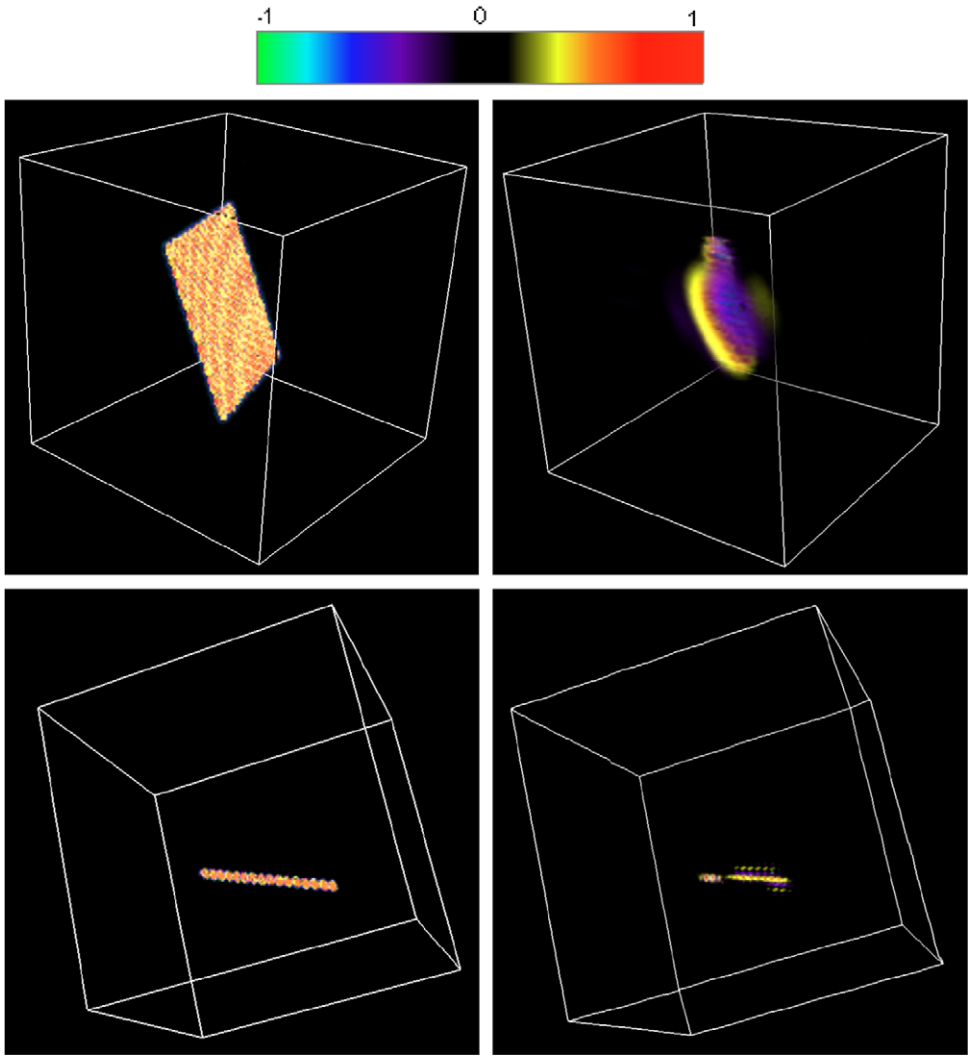

Fig. 9. Top row: a plane, and the reconstruction after thresholding the RidCurvelet coefficients at 8.2 (see Table 1). Bottom row: a filament, and its reconstruction after the thresholding of the BeamCurvelet coefficients at 1.31.

Fig. 10. From left to right: a 3D view of the cube containing pieces of shells and a spring-shaped filament, a slice of the previous cube, and finally a slice from the noisy cube.

Table 2
PSNR of the denoised synthetic cube using wavelets, RidCurvelets or BeamCurvelets.

	Wavelets	RidCurvelets	BeamCurvelets
Shells & spring	40.4 dB	40.3 dB	43.7 dB

Fig. 11. From left to right: a slice from the filtered test cube described in Section 4.3.2 (original in Fig. 10) by the wavelet transform (isotropic undecimated), the RidCurvelets and the BeamCurvelets.

Algorithm 6: The combined filtering.

Data: The observed data y, the noise level σ, an initial threshold level $k^{(0)}\sigma$, and a number of iterations N.
Result: The estimate $x^{(N)}$ of the unknown x.
Let $\mathcal{ST}_\lambda$ be the soft-thresholding operator with threshold λ.
Let $\Phi_k, k \in [0, K)$ be the K dictionaries to combine.
Let $\kappa = 3$ be the final threshold level (when multiplied by σ).
begin
  $x^{(0)} = 0$ *// Initial estimate*
  for $n = 0$ *to* $N-1$ **do**
    $u^{(0)} = x^{(n)}$
    $\lambda^{(n)} = (k^{(0)}\sigma - \kappa\sigma)(N-1-n)/(N-1) + \kappa\sigma$ *// Current threshold level*
    for $k = 0$ *to* $K-1$ **do**
      $u^{(k+1)} = u^{(k)} + \Phi_k \mathcal{ST}_{\lambda^{(n)}}(\Phi_k^T(y - u^{(k)}))$ *// New estimate*
    $x^{(n+1)} = u^{(K)}$
end

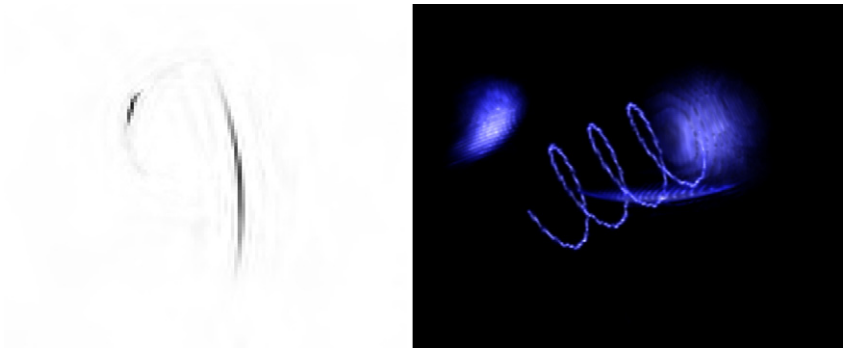

Fig. 12. A slice and a 3D view of the noisy cube in Fig. 10 after denoising with the presented combined filtering.

The idea is that the residual between the data and the filtered cube must not be sparse in any of the chosen transforms, and thus leads to small coefficients. It can be interpreted as "There is no structure (in the sense of detection with Φ_k) in the residual". We aim to solve the optimization problem

$$\min_{\tilde{x}} \|y - \tilde{x}\|_{\ell_2}^2 \quad \text{s.t. } \forall k \in [1, K]\text{: } \left\|\Phi_k^T(y - \tilde{x})\right\|_{\ell_\infty} < \kappa\sigma, \tag{25}$$

where σ is the noise estimated on the data (or the energy of the faintest detail we wish to recover). This problem can be solved using Algorithm 6. With this new algorithm, the memory required to treat data of size 128^3 with wavelets, BeamCurvelets and RidCurvelets with overlapping is about 110 MB, while it would need about 4 GB with the old algorithm if it were optimally coded (not using transforms as black boxes, but interacting with them). In our current implementation, it uses twice that.

We used this combined filtering method to denoise the synthetic data in Fig. 10, and compared to the single-transform denoising PSNRs (see Table 2), we get a PSNR of 45.4 dB, which is almost 2 dB higher than the best transform alone. Fig. 12 shows the result of the new CFA.

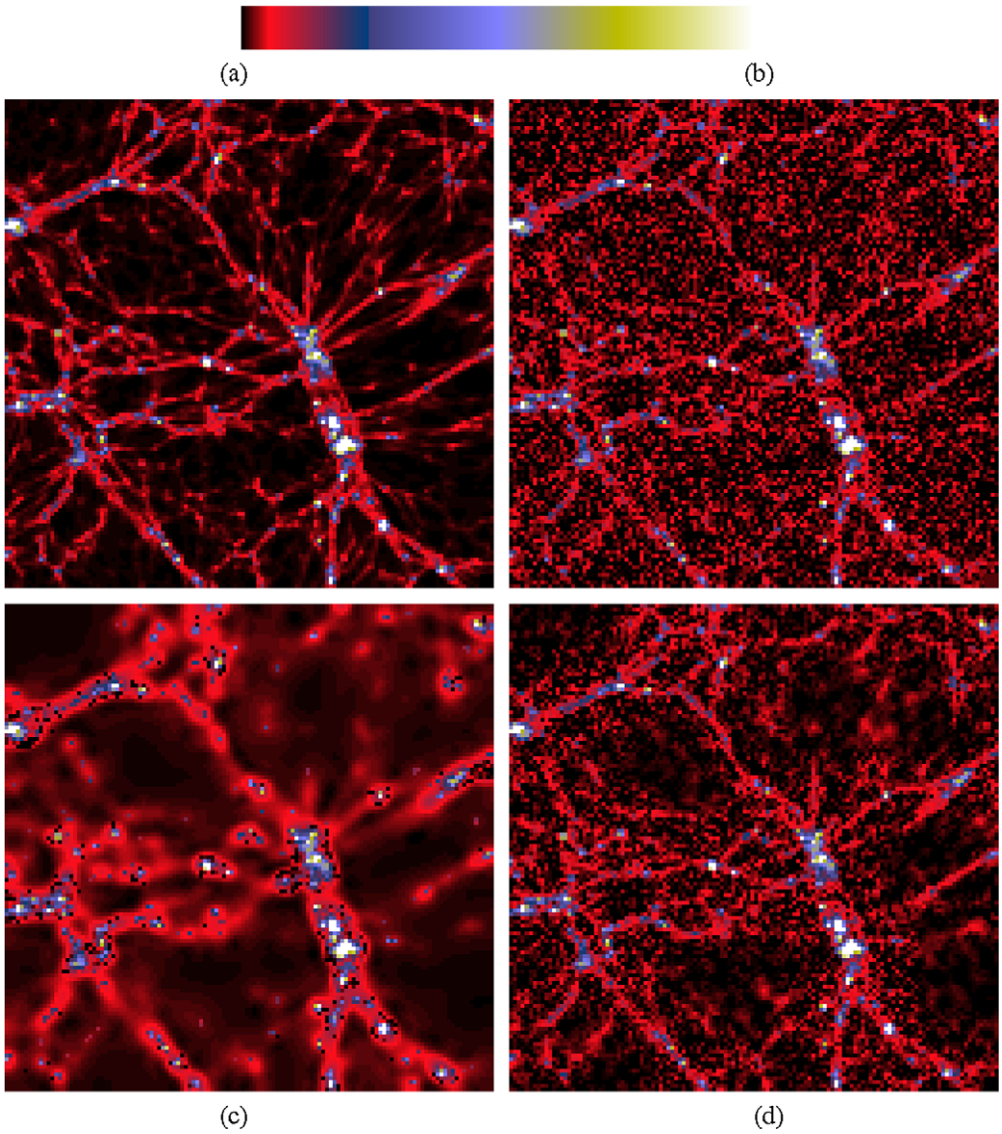

Fig. 13. The central slice of (a) the original λCDM data cube, (b) the noisy data, (c) the data recovered by wavelet hard thresholding, and (d) recovered by BeamCurvelet thresholding.

4.3.4. λCDM denoising

In astrophysics, we use N-body simulations when making an initial homogeneous universe evolve with gravity to form structures, and during this process, there is a noise present in the low density areas. Simulations of cold dark matter (λCDM) exhibit formation of clusters and filamentary, the density of the filaments being a thousand times lower than the clusters. As the noise is important at low density, because of material particles of a given mass, we have to apply a denoising between the time iterations of the simulation in order to improve it.

In the following example, we show a denoising of a simulated λCDM cube obtained by a simulation using the RAMSES code [44] with cosmological parameters as follows: $\Omega_m = 0.3$ (i.e. 30% of standard matter), $\Omega_\lambda = 0.7$ (70% of dark matter, which is interacting gravitationally only), the Hubble parameter $h = 0.7$ (the rate of expansion of the universe) and $\sigma_8 = 0.9$ (the dark matter density fluctuations at present time at scale 8 MPc).

The noise added to the data has a standard deviation ($\sigma = 0.025$) comparable to the filaments amplitude (0.02), which is very small compared to the total amplitude of the data (≈ 100), as the most dense clusters are five orders of magnitude higher than the filaments.

We first tried denoising the data using the undecimated wavelet transform, which correctly recovers the high-level clusters, but it fails to separate the filamentary structure from the noise. Facing this problem, we used the BeamCurvelet transform instead. The result, shown in Fig. 13, although recovering the filaments, reveals strong artifacts surrounding the high-density clusters, and therefore is useless. This phenomenon is not due to the noise, but to the form of the BeamCurvelet

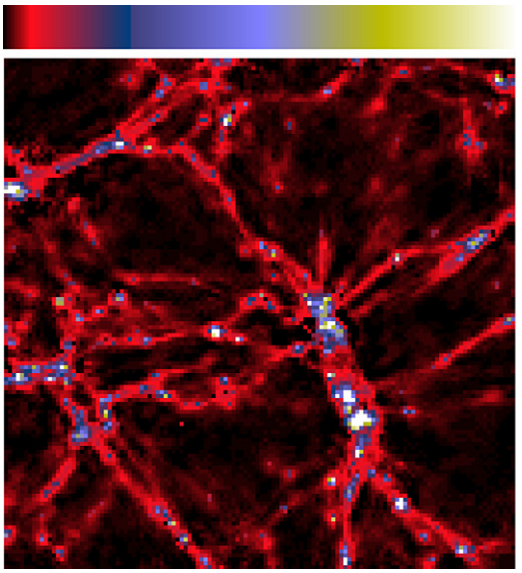

Fig. 14. Denoised λCDM data using the combined filtering (see Algorithm 6).

atoms. Fine-scale BeamCurvelet atoms have an oscillating behavior, and although these oscillations are very low (order $1e{-}4$ compared to the atom's amplitude), they become a serious problem when structures contained in the data have a huge dynamic range, as in our λCDM simulations where clusters are 10^5 larger than filaments. When the atom is used to approximate a high density object, and when we look round it at scales 10^5 smaller, the oscillations become dominant. In order to solve this problem, we must use conjointly the wavelets and the BeamCurvelets combined in the algorithm we presented in Section 4.3.3.

The results of denoising the λCDM data using the new combined filtering algorithm (Algorithm 6) are shown in Fig. 14. As expected, the result is far better than using one transform alone, and we restore the filaments very well. The oscillating structure has also naturally completely disappeared from the residual.

5. 3D inpainting

5.1. Introduction

When measuring quantities on sky observations, we often have to apply a mask to the data to cover inappropriate data, due to the stacking on the line of sight. For example, on the full-sky microwaves images, we have to mask out the whole galactic plane (roughly a cylinder on the sphere) because it is very polluted by the foreground, the Milky Way. In the case of 3D data, there can be a whole cone missing, a sphere of a given radius around the center, or any local area hidden by something. This problem has already been addressed with sparse representations in 2D, for example in weak lensing data [26]. We investigate, in this section, how the 2D sparse inpainting methods can be applied in 3D, using the new 3D curvelet transforms.

5.2. The inpainting problem

Inpainting is the process of recovering missing parts in altered data from the still available clues. Let x be our three-dimensional data cube with missing data indicated by the mask M. The available data is $y = Mx$. Given a dictionary Φ, we are trying to recover x from the observed y and the mask M. This is an ill posed inverse problem. To get a consistent solution, one must seek regularized solutions. One such regularization is to suppose that x is sparse in one dictionary of atoms Φ, which means that x can be represented by a few atoms from Φ.

Therefore, we want to solve the following inpainting problem:

$$\arg\min_x \left\| \Phi^T x \right\|_0 \quad \text{s.t. } \|y - Mx\|_2 < \epsilon(\sigma), \tag{26}$$

where Φ^+ stands for the analysis operator (forward transform). To solve this problem, Algorithm 7 is applied (see [16]).

5.3. Example 1: Random missing voxels

We tried to push the degradation process to see how the BeamCurvelet performed. We applied different masks to the λCDM data, with an increasing percentage of randomly missing voxels. The data are strictly positive (matter density); to

Algorithm 7: The inpainting process.

Data: A mask M, the observed data $x = My$, y being the unknown cube, the number of iterations N, an initial threshold level $k^{(0)}$.
Result: The estimate $x^{(N)}$ of y.
Let $\mathcal{HT}_{\lambda^{(n)}}$ be the hard-thresholding operator with threshold $\lambda^{(n)}$.
MAD stands for *median absolute deviation*.
begin
  $x^{(0)} = y$
  $\forall i \in [0, N)$: $k^{(i)} = k^{(0)}(N - 1 - i)/(N - 1)$
  for $n = 0$ *to* $N - 1$ **do**
    $\lambda^{(n)} = k^{(n)} \cdot MAD(\Phi^T x^{(n)})$
    $x^{(n+1)} = \Phi \mathcal{HT}_{\lambda^{(n)}}(\Phi^T[x^{(n)} + M(y - Mx^{(n)})])$
    $= \Phi \mathcal{HT}_{\lambda^{(n)}}(\Phi^T[(I - M)x^{(n)} + y])$
end

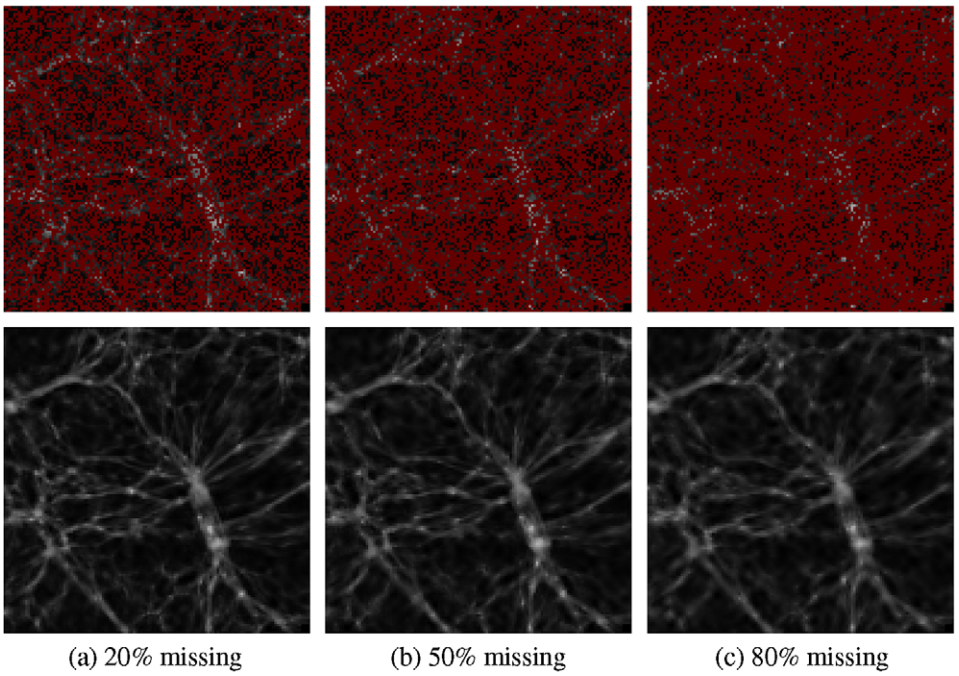

Fig. 15. Central slice of the masked λCDM data with 20, 50, and 80% missing voxels, and the inpainted maps. The missing voxels are dark red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

avoid high dynamic range issues, in the whole inpainting section we work on the logarithm of the data, but for more clarity, we display all figures with a linear scale, truncated at about 10 times the level of the filaments; the strong clusters are thus saturated. Fig. 15 presents slices of the original data with missing 20, 50 and 80% missing voxels, and the reconstructed data. The global structure is perfectly recovered in the three examples, and only the faintest filaments are lost in the 80% missing voxels data. The inpainting process and the transform used (the BeamCurvelet transform) are very well adapted to recover this kind of structure. Animations of the inpainting process and 3D representations of the results of this experiment and the following ones can be seen online at http://arnaud.woiselle.fr in the research panel.

5.4. Example 2: Missing blocks

In order to see the gain when using one transform rather than another in inpainting when the missing data is spatially coherent, we applied a mask to our λCDM cube, with 50% randomly missing voxels, and an additional 3D *checkerboard* array of missing boxes of 343 contiguous voxels each ($7 \times 7 \times 7$). We applied the inpainting algorithm to restore the data using either the RidCurvelet transform, or the BeamCurvelet transform. The results are shown in Fig. 16. The reconstruction is very good in both cases, but as the BeamCurvelets are more adapted to the structure of the data than the RidCurvelets, their reconstruction is better: the L^2 and L^1 norms of the errors are lower with the BeamCurvelets, and there is visually less residual noise in the reconstruction, which can be understood as the atoms (planes) of the RidCurvelet transform must spread around the data's filaments more than the atoms (filaments) of the BeamCurvelet transform to approximate the data.

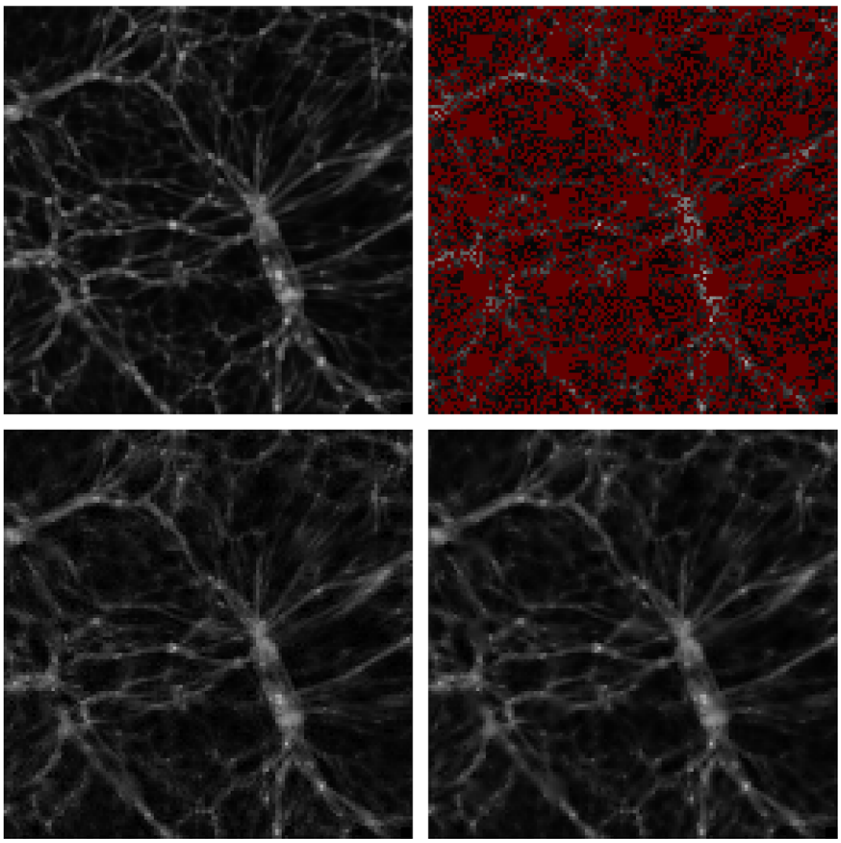

Fig. 16. First row: original central frame of the λCDM data cube, and degraded version with missing voxels in red. Bottom row: the filtered results using the RidCurvelets (left) and the BeamCurvelets (right). Missing voxels are dark red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

6. Conclusion

We have presented two novel multiscale geometric decompositions of a three-dimensional volume. The RidCurvelet represents well surfaces and the BeamCurvelet is well adapted for analyzing filamentary features. Preliminary results have shown that they produce interesting results for denoising and inpainting. An important aspect we did not treat is the noise property for denoising applications. Here we have considered only Gaussian noise. However, most 3D astronomical data sets require that we consider Poisson noise. A solution would be to consider nonlinear 3D decompositions, using both BeamCurvelet and the multiscale variance stabilization, in a similar way to what has been done for the 2D curvelet transform Poisson denoising [43].

Acknowledgments

We wish to thank Romain Teyssier for providing us the λCDM simulated data [44] used in Sections 5 and 4. This work is supported by Sagem DS.

References

[1] Y. Benjamini, Y. Hochberg, Controlling the false discovery rate: A practical and powerful approach to multiple testing, J. Roy. Statist. Soc. Ser. B Methodol. 57 (1) (1995) 289–300.
[2] S. Berlemont, A. Bensimon, J.-C. Olivo-Marin, Detection of curvilinear objects in biological noisy image using feature-adapted fast slant stack, in: SPIE Conference Wavelets XII, Special Session on Wavelet in Bioimaging, 2007.
[3] E.J. Candès, D.L. Donoho, Ridgelets: the key to high dimensional intermittency, Philos. Trans. R. Soc. Lond. Ser. A 357 (Sep. 1999) 2495–2509.
[4] E.J. Candès, D.L. Donoho, Curvelets, multiresolution representation, and scaling laws, in: SPIE Wavelet Applications in Signal and Image Processing, 2000.
[5] E.J. Candès, D.L. Donoho, New tight frames of curvelets and optimal representations of objects with C2 singularities, Comm. Pure Appl. Math. 57 (2) (2003) 219–266.
[6] P. Carre, D. Helbert, E. Andres, 3D fast ridgelet transform, in: International Conference on Image Processing, vol. 1, 2003, pp. 1021–1024.
[7] V. Chandrasekaran, M.B. Wakin, D. Baron, R.G. Baraniuk, Surflets: A sparse representation for multidimensional functions containing smooth discontinuities, in: Proceedings of the International Symposium on Information Theory, ISIT 2004, July 2004.

[8] V. Chandrasekaran, M.B. Wakin, D. Baron, R.G. Baraniuk, Representation and compression of multidimensional piecewise functions using surflets, IEEE Trans. Inform. Theory 55 (1) (Jan. 2009) 374–400.
[9] Z. Chen, R. Ning, Breast volume denoising and noise characterization by 3D wavelet transform, Comput. Med. Imaging Graph. 28 (5) (2004) 235–246.
[10] A. Dima, M. Scholz, K. Obermayer, Semiautomatic quality determination of 3D confocal microscope scans of neuronal cells denoised by 3D wavelet shrinkage, in: H.H. Szu (Ed.), Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, vol. 3723, March 1999, pp. 446–457.
[11] M.N. Do, M. Vetterli, The contourlet transform: an efficient directional multiresolution image representation, IEEE Trans. Image Process. 14 (12) (Dec. 2005) 2091–2106.
[12] D.L. Donoho, X. Huo, Beamlets and multiscale image analysis, in: Multiscale and Multiresolution Methods, in: Lecture Notes in Computational Science and Engineering, vol. 20, Springer, New York, 2001, pp. 149–196.
[13] D.L. Donoho, Wedgelets: nearly minimax estimation of edges, Ann. Statist. 27 (3) (1999) 859–897.
[14] D.L. Donoho, Tight frames of k-plane ridgelets and the problem of representing objects that are smooth away from d-dimensional singularities in $\mathbb{R}^n$, in: Applied Mathematics, Proc. Natl. Acad. Sci. USA 96 (1999) 1828–1833.
[15] D.L. Donoho, O. Levi, Fast X-ray and beamlet transforms for three-dimensional data, in: D. Rockmore, D. Healy (Eds.), Modern Signal Processing, 2002, pp. 79–116.
[16] M. Elad, J.-L. Starck, P. Querre, D.L. Donoho, Simultaneous cartoon and texture image inpainting using morphological component analysis, Appl. Comput. Harmon. Anal. 19 (2005) 340–358.
[17] E. Escalera, E. Slezak, A. Mazure, New evidence for subclustering in the Coma cluster using the wavelet analysis, Astronom. Astrophys. 264 (Oct. 1992) 379–384.
[18] M.J. Fadili, J.-L. Starck, F. Murtagh, Inpainting and zooming using sparse representations, Comput. J. 52 (1) (2007) 64–79.
[19] G. Hennenfent, F.J. Herrmann, Seismic denoising with nonuniformly sampled curvelets, IEEE Comput. Sci. Eng. 8 (3) (May 2006) 16–25.
[20] Y. Lu, M.N. Do, 3-D directional filter banks and surfacelets, in: Proc. of SPIE Conference on Wavelet Applications in Signal and Image Processing XI, San Diego, USA, 2005.
[21] Y.M. Lu, M.N. Do, Multidimensional directional filter banks and surfacelets, IEEE Trans. Image Process. 16 (4) (2007) 918–931.
[22] V.J. Martínez, J.-L. Starck, E. Saar, D.L. Donoho, S.C. Reynolds, P. de la Cruz, S. Paredes, Morphology of the galaxy distribution from wavelet denoising, Astrophys. J. 634 (Dec. 2005) 744–755.
[23] V.J. Martinez, S. Paredes, E. Saar, Wavelet analysis of the multifractal character of the galaxy distribution, MNRAS 260 (Jan. 1993) 365–375.
[24] C.J. Miller, C. Genovese, R.C. Nichol, L. Wasserman, A. Connolly, D. Reichart, A. Hopkins, J. Schneider, A. Moore, Controlling the false-discovery rate in astrophysical data analysis, Astronom. J. 122 (6) (Dec. 2001) 3492–3505.
[25] G. Peyre, S. Mallat, Discrete bandelets with geometric orthogonal filters, in: Proceedings of ICIP'05, vol. 1, Sept. 2005, pp. 65–68.
[26] S. Pires, J.-L. Starck, A. Amara, R. Teyssier, A. Refregier, J. Fadili, FASTLens (FAst STatistics for weak Lensing): Fast method for weak lensing statistics and map making, ArXiv e-print, April 2008.
[27] J.K. Romberg, M. Wakin, R. Baraniuk, Multiscale wedgelet image analysis: fast decompositions and modeling, in: IEEE Int. Conf. on Image Proc., ICIP'02, vol. 3, June 2002, pp. 585–588.
[28] A.B. Romeo, C. Horellou, J. Bergh, N-body simulations with two-orders-of-magnitude higher performance using wavelets, MNRAS 342 (June 2003) 337–344.
[29] A.B. Romeo, C. Horellou, J. Bergh, A wavelet add-on code for new-generation n-body simulations and data de-noising (jofiluren), MNRAS 354 (Nov. 2004) 1208–1222.
[30] D. Rusanovskky, K. Egiazarian, Video denoising algorithm in sliding 3D DCT domain, Lecture Notes in Comput. Sci. 37 (08) (2005) 618–625.
[31] I.W. Selesnick, K.Y. Li, Video denoising using 2D and 3D dual-tree complex wavelet transforms, in: Proc. of SPIE Conference on Wavelet Applications in Signal and Image Processing X, San Diego, USA, Aug. 2003.
[32] L. Sendur, I.W. Selesnick, Bivariate shrinkage functions for wavelet-based denoising exploiting interscale dependency, IEEE Trans. Signal Process. 50 (11) (Nov. 2002) 2744–2756.
[33] L. Sendur, I.W. Selesnick, Bivariate shrinkage with local variance estimation, IEEE Signal Process. Lett. 9 (12) (Dec. 2002) 438–441.
[34] E. Slezak, V. de Lapparent, A. Bijaoui, Objective detection of voids and high density structures in the first CfA redshift survey slice, Astrophys. J. 409 (1993) 517–529.
[35] J.-L. Starck, E.J. Candès, D.L. Donoho, The curvelet transform for image denoising, IEEE Trans. Image Process. 11 (6) (June 2002) 670–684.
[36] J.-L. Starck, F. Murtagh, Astronomical Image and Data Analysis, Astronomy and Astrophysics Library, Springer, Berlin, 2006.
[37] J.-L. Starck, F. Murtagh, E. Candès, D.L. Donoho, Gray and color image contrast enhancement by the curvelet transform, IEEE Trans. Image Process. 12 (6) (2003) 706–717.
[38] J.L. Starck, A. Bijaoui, B. Lopez, C. Perrier, Image reconstruction by the wavelet transform applied to aperture synthesis, Astronom. Astrophys. 283 (1999) 349–360.
[39] J.L. Starck, D.L. Donoho, E. Candès, Very high quality image restoration by combining wavelets and curvelets, in: A. Laine, M.A. Unser, A. Aldroubi (Eds.), SPIE Conference on Signal and Image Processing: Wavelet Applications in Signal and Image Processing IX, San Diego, 1–4 August 2001, SPIE, 2001.
[40] J.L. Starck, V.J. Martinez, D.L. Donoho, O. Levi, P. Querre, E. Saar, Analysis of the spatial distribution of galaxies by multiscale methods, EURASIP J. Appl. Signal Process. 15 (Nov. 2005) 2455–2469.
[41] J.L. Starck, M.K. Nguyen, F. Murtagh, Deconvolution based on the curvelet transform, in: International Conference on Image Processing, vol. II, 2003, pp. 993–996.
[42] J.L. Starck, M.K. Nguyen, F. Murtagh, Wavelets and curvelets for image deconvolution: a combined approach, Signal Process. 83 (2003) 2279–2283.
[43] B. Zhang, J. Fadili, J.L. Starck, Wavelets, ridgelets, and curvelets for Poisson noise removal, IEEE Trans. Image Process. 17 (7) (July 2008) 1093–1108.
[44] R. Teyssier, Cosmological hydrodynamics with adaptive mesh refinement. A new high resolution code called ramses, Astronom. Astrophys. 385 (April 2002) 337–364.
[45] L. Ying, L. Demanet, E. Candès, 3D discrete curvelet transform, Appl. Comput. Math. 217 (50) (2005).