3-D Data Denoising and Inpainting with the low-redundancy Fast Curvelet Transform

A. Woiselle[1,2], J-L. Starck[1], J. Fadili[3]

[1] 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.

[2] Sagem groupe SAFRAN, 72-74 rue de la Tour Billy, 95101 Argenteuil CEDEX, France.

[3] GREYC CNRS UMR 6072, Image Processing Group, ENSICAEN 14050 Caen Cedex, France.

October 5, 2010

Abstract

In this paper, we first present a new implementation of the 3-D fast curvelet transform, which is nearly 2.5 less redundant than the Curvelab (wrapping-based) implementation as originally proposed in [1, 2], which makes it more suitable to applications including massive data sets. We report an extensive comparison in denoising with the Curvelab implementation as well as other 3-D multi-scale transforms with and without directional selectivity. The proposed implementation proves to be a very good compromise between redundancy, rapidity and performance. Secondly, we exemplify its usefulness on a variety of applications including denoising, inpainting, video de-interlacing and sparse component separation. The obtained results are good with very simple algorithms and virtually no parameter to tune.

Keywords: 3-D Curvelets, sparsity, denoising, inpainting, Morphological Component Analysis, video deinterlacing.

Contents

1 Introduction

Since the introduction of the multiscale point of view in image processing, an important gap has been crossed with the development of directional transforms such as the curvelet transform [3].

For instance, the latter provide a non-adaptive (near-)optimally sparse representation of smooth curvilinear structures in images, while their wavelet predecessors obtained as tensor-product of 1-D wavelets fail to efficiently represent these anisotropic structures. In fact, the curvelet transform provides atoms that are well localized in space and frequency and exhibit a strong directional selectivity. This transform has found a wide spectrum of applications including denoising [4, 5, 6], contrast enhancement [7], inpainting [8, 9] or deconvolution [10, 11].

With the increasing computing power and memory storage capabilities of computers, it becomes feasible to analyze 3-D data as a volume and not only slice-by-slice, which would mistakingly miss the 3-D geometrical nature of the data. In order to process such massive data, there is an urgent need to design new 3-D multiscale directional transforms. In the literature, some 3-D representations have already been developed, in addition to the natural extension of separable wavelets to 3-D. For example, 2-D wedgelets [12, 13] have been extended to d-D Surflets for compression purposes [14, 15]. d-dimensional wedgelets correspond to an adaptive transform based on a 2^d-tree spatial partitioning of the data, where each leaf of the 2^d-tree has two constant regions separated by a smooth polynomial manifold of codimension 1. Surfacelets have been proposed in [16] as a non-adaptive transform to efficiently represent singularities along 2-D smooth surfaces. They correspond to a 3-D extension of contourlets, and similarly to contourlets, their implementation relies on a filterbank decomposition of the 3-D data in both scale and angle. They produce a tiling of the 3-D frequency domain quite similar to the one of curvelets although not strictly faithful to the sharp frequency localization of curvelets. The latter were first designed and characterized in a 2-D continuous setting [3, 17], then discretized and extended to higher dimensions [1, 2]. The 2-D curvelet representation was formally proved to yield a (near) optimal sparse representation of piecewise smooth (C^2) images away from singularities along C^2 edges. More generally, the curvelet transform would provide efficient representation of d-D smooth objects with singularities along smooth manifolds of codimension 1. The 3-D implementation of the fast curvelets was proposed in [1, 2] with a public code distributed (including the 2-D version) in Curvelab, a C++/Matlab toolbox available at `www.curvelet.org`. This 3-D fast curvelet transform has found applications mainly in seismic imaging, for instance for denoising [18] and inpainting [19]. However, a major drawback of this transform is its high redundancy factor, of approximately 25. As a straightforward and somewhat naive remedy to this problem, the authors in [1, 2] suggest to use wavelets at the finest scale instead of curvelets, which indeed reduces the redundancy dramatically to about 5.4 (see Section 2.3 for details). However, this comes at the price of the loss of directional selectivity of fine details. On the practical side, this entails poorer performance in restoration problems compared to the full curvelet version. Note that directional selectivity was one of the main reasons curvelets were built at the first place.

In this paper, we present a new implementation of the fast curvelet transform (FCT) with a special emphasis on the 3-D case that has a reduced redundancy, of only 10, without sacrificing the directional selectivity property at the finest scale. Our implementation provides a very good tradeoff between the full FCT and its less-redundant counterpart using only wavelets at the finest scale to reduce the redundancy, as suggested in [1, 2]. In Section 2, we present our new implementation of the FCT, and compare it to the one implemented in Curvelab. In Section 3, we show how this new curvelet construction can be wisely used for many different applications including denoising, inpainting, video deinterlacing and sparse component separation.

2 Low redundancy 3-D fast curvelets

2.1 Background

The FCT of a 3-D object consists of a low-pass approximation subband, and a family of curvelet subbands carrying the curvelet coefficients indexed by their scale, position and orientation in 3-D. These 3-D FCT coefficients are formed by a proper tiling of the frequency domain following two steps (see Figure 1):

- Cartesian coronization or multiscale separation: first decompose the object into (Cartesian) dyadic coronae based on concentric cubes;

- Angular separation: each corona is separated into anisotropic wedges of trapezoidal shape obeying the so-called parabolic scaling law (to be defined shortly). The 3-D FCT coefficients are obtained by an inverse Fourier transform of applied to each wedge appropriately wrapped to fit into a 3-D rectangular parallelepipeds.

In the rest of this section, we will rehearse the main details of these two steps of the 3-D FCT, and describe the modifications we propose in the implementation to reduce its redundancy.

Symmetry The FCT isolates, in the frequency domain, oriented and localized 3-D wedges. There is a symmetry on a 3-D Cartesian grid : the cube has six faces which can be processed in a similar way. Let $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3) \in [-1/2, 1/2]^3$ be a frequency in the 3-D Shannon band. Exploiting the above symmetry, we will only focus on the subspace $\{\omega_1 > 0, |\omega_2/\omega_1| < 1, |\omega_3/\omega_1| < 1\}$ which is a squared-based pyramid. The five other parts can be dealt with exactly in the same way by symmetry around the origin and exchange of axes.

Cartesian coronization The multiscale separation is achieved using a 3-D Meyer wavelet transform [20, 21], where the Meyer wavelet and scaling functions are defined in Fourier domain with compactly supported Fourier transforms.

Let's denote ψ_j as the Meyer wavelet at scale $j \in \{0, \cdots, J-1\}$, and ϕ_{J-1} the scaling function at the coarsest scale. Denote $M_j = \hat{\psi}_j = 2^{-3j/2}\hat{\psi}(2^{-j}\cdot)$ and $M_J = \hat{\phi}_{J-1} = 2^{-3(J-1)/2}\hat{\phi}(2^{-(J-1)}\cdot)$ their Fourier transforms (see Section 2.2.1 for a formal definition). M_J is a lowpass and the wavelet functions $\{M_j\}_{0 \le j < J}$ is a family of bandpass frequency localized windows that form a uniform partition of the unity. Applied to a 3-D object, the family $\{M_j\}_{0 \le j < J}$ separates it into Cartesian coronae (annuli), and M_J selects its low frequency content. This coarsest subband is kept unaltered, and after an inverse Fourier transform, provides us with the first curvelet coefficients, which corresponds to coarse scale isotropic atoms. Only the next detail scales, i.e. those corresponding to $\{M_j\}_{0 \le j < J}$, have to be processed further.

Angular separation Let N_a be the number of angles on one edge of one face of the finest scale, for a total of N_a^2 angles on each face, and thus six times more bands for the entire considered scale (see Figure 1). This number varies with the scale because the number of angles decreases with the scales becoming coarser to obey the parabolic scaling law of curvelets [2], with a parabolic scaling matrix $\mathrm{diag}(2^{-j}, 2^{-j/2}, 2^{-j/2})$ (one short direction and two long ones). This property is essential to ensure that the 3-D curvelets are a basis for sparsely representing smooth trivariate functions with 2-D smooth surface-like singularities.

The vector indexing the angular locations on a face at the j^{th} scale may be expressed with $\mathbf{l} = (l, l') \in \{0, \cdots, 2^{\lfloor -j/2 \rfloor} N_a - 1\}^2$, where $\lfloor \cdot \rfloor$ is the integer part of its argument[1]. Recall that a wedge is the trapezoidal region sharply localized along a given angle at a given scale, see the dark gray area in Figure 1. The center of the wedge is on the line going from the origin to the point $(1, \theta_l, \theta'_{l'})$, with

$$\theta_l = \left(-1 + \frac{2l+1}{2^{\lfloor -j/2 \rfloor} N_a}\right), \theta'_{l'} = \left(-1 + \frac{2l'+1}{2^{\lfloor -j/2 \rfloor} N_a}\right). \tag{1}$$

We can now define the angular separation by multiplying the dyadic annuli corresponding to the wavelet detail subbands by the smooth angular windows $V_{j,\mathbf{l}}$ in the Fourier domain. The angular windows are built from a smooth real-valued function V supported on $[-1, 1]$ and satisfying the partition property

$$\sum_{l=-\infty}^{\infty} V^2(t - 2l) = 1 \quad \forall t \in \mathbb{R}. \tag{2}$$

The angular window at scale j and orientation $\mathbf{l} = (l, l')$ is then constructed as

$$V_{j,\mathbf{l}}(\boldsymbol{\omega}) \quad = \quad V\left(2^{\lfloor -j/2 \rfloor} N_a \frac{\omega_2 - \theta_l \omega_1}{\omega_1}\right) \cdot V\left(2^{\lfloor -j/2 \rfloor} N_a \frac{\omega_3 - \theta'_{l'} \omega_1}{\omega_1}\right), \tag{3}$$

[1] On a Cartesian grid, the slopes are equispaced not the angles.

where θ_l and $\theta'_{l'}$ are defined in (1). Note the scaling factor $2^{-j/2}$ as dictated by the parabolic scaling. Owing to (2), the family of angular windows $\{V_{j,\mathbf{l}}\}_{\mathbf{l}}$ makes a uniform partition of the dyadic annulus at scale j, i.e.

$$\sum_{\mathbf{l}} V^2_{j,\mathbf{l}}(\boldsymbol{\omega}) = 1 \ . \tag{4}$$

However, because of the support constraint on V, this relation does not hold for all $\boldsymbol{\omega}$, and a special care should be taken at the corners where only three out of usually four windows overlap. We thus need to redefine them for (4) to hold for any $\boldsymbol{\omega}$. Here is a simple remedy to this problem. Let $\mathbf{l}_a, \mathbf{l}_b, \mathbf{l}_c$ be the indices of the three corner windows, we redefine them on their overlapping domain Ω as

$$\begin{aligned} &\forall \boldsymbol{\omega} \in \Omega, \forall \mathbf{l} \in \{\mathbf{l}_a, \mathbf{l}_b, \mathbf{l}_c\} \\ &V_{j,\mathbf{l}}(\boldsymbol{\omega}) \leftarrow \frac{V_{j,\mathbf{l}}(\boldsymbol{\omega})}{\sqrt{V^2_{j,\mathbf{l}_a}(\boldsymbol{\omega}) + V^2_{j,\mathbf{l}_b}(\boldsymbol{\omega}) + V^2_{j,\mathbf{l}_c}(\boldsymbol{\omega})}}. \end{aligned} \tag{5}$$

Piecing all ingredients together, the *scale-angular* wedge at scale-orientation $(j, \mathbf{l})$ is extracted by the frequency window

$$W_{j,\mathbf{l}}(\boldsymbol{\omega}) = M_j(\boldsymbol{\omega}) \cdot V_{j,\mathbf{l}}(\boldsymbol{\omega}) \ , \tag{6}$$

which is sharply localized near the trapezoid

$$\left\{(\omega_1, \omega_2, \omega_3) : 2^{j+1} < \omega_1 < 2^j, \left|\frac{\omega_2}{\omega_1} - \theta_l\right| < 2^{\lfloor j/2 \rfloor}/N_a, \left|\frac{\omega_3}{\omega_1} - \theta'_{l'}\right| < 2^{\lfloor j/2 \rfloor}/N_a\right\}. \tag{7}$$

Once a wedge is extracted, an inverse Fourier transform must be applied in order to get the curvelet coefficients at the corresponding scale and orientation. Prior to this, the trapezoidal wedge has to be transformed to a convenient form for which the 3-D FFT algorithm applies. As it can be seen from Figure 1, the wedge can be inscribed inside a 3-D parallelepiped which is $\sim 2^{j/2}$ long on the (ω_2, ω_3) coordinates (i.e. tangentially), and 2^j on ω_1, i.e. radially. Although this expands the area including the wedge, we can still wrap it inside a rectangular parallelepiped of dimensions $\sim (2^j, 2^{j/2}, 2^{j/2})$ centered at the origin aligned with the axes of the grid (see Section 2.2.2 for further details about wrapping). With appropriate choice of the size of the rectangular parallelepiped, the data does not overlap with itself after wrapping. With the wrapping trick, an inverse 3-D FFT can be readily applied to the rectangular parallelepiped to obtain the curvelet coefficients at the selected scale and orientation.

Algorithm 1 summarizes our implementation of the 3-D FCT and outlines its main steps.

Algorithm 1: The 3-D Fast Curvelet Transform

Data: A 3-D data object f of size $\mathbf{N} = (N_x, N_y, N_z)$.
Input: Number of scales J, number of angles N_a on each face at the finest scale.
begin

1. **Multiscale separation:** apply the 3-D Meyer wavelet transform in Fourier domain, get cubes of sizes $\mathbf{N}$, $\mathbf{N}/2$, ..., $\mathbf{N}/2^J$.

2. **Angular separation:**

foreach ***scale*** $j = 0$ ***to*** J **do**
  foreach ***orientation*** $\mathbf{l} = (l, l')$ **do**
    Multiply the wavelet cube at scale j with the angular window $V_{j,\mathbf{l}}$ in Fourier;
    Wrapping: wrap the result in a rectangular parallelepiped centered at the origin of minimal size $(2^j 3/8 \times 2^{j/2+1}/N_a \times 2^{j/2+1}/N_a)$;
    Apply a 3-D inverse FFT to the rectangular parallelepiped to collect the curvelet coefficients ;

end
Result: 3-D FCT of f.

We are now ready to explain the differences we introduced in the 3-D FCT implementation to reduce its redundancy.

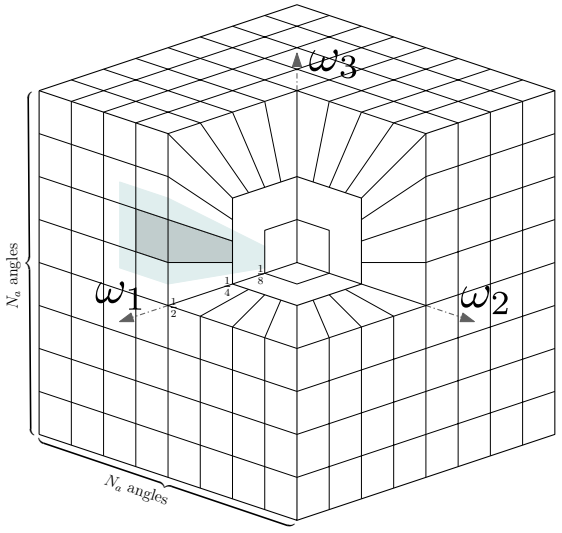


Figure 1: Example in 3-D of how the number of directions N_a is defined along an edge, (eight in this example), and showing the overlapping region around one wedge. The dark gray area is the main part of the wedge, and the light one represents the overlapping region.

2.2 A new implementation

The overall implementation of this new FCT differs from the original one in several points. The main difference lays in the way to apply the Meyer wavelet transform to the data. Other changes have been introduced as described hereafter.

2.2.1 The multiscale separation

The extra redundancy of the curvelets as implemented in Curvelab originates mainly from the way the radial window is implemented, especially at the finest scale. For the sake of clarity, we will first explain the 1-D case for the Meyer wavelets and show the implications in higher dimensions later on.

The Meyer wavelets $\hat{\psi}(\xi)$ are defined in Fourier domain as follows :

$$\hat{\psi}(\xi) = \begin{cases} \exp^{-i2\pi\xi} \sin(\frac{\pi}{2}\nu(6|\xi|-1)), & \text{if } 1/6 < |\xi| \leq 1/3 \\ \exp^{-i2\pi\xi} \cos(\frac{\pi}{2}\nu(3|\xi|-1)), & \text{if } 1/3 < |\xi| \leq 2/3 \\ 0 & \text{elsewhere} \end{cases},$$

where ν is a smooth function, that goes from 0 to 1 on $[0,1]$ and satisfies $\nu(x) + \nu(1-x) = 1$. Associated to this wavelet is the Meyer scaling functions defined by

$$\hat{\phi}(\xi) = \begin{cases} 1, & \text{if } |\xi| \leq 1/6 \\ \cos(\frac{\pi}{2}\nu(6|\xi|-1)), & \text{if } 1/6 < |\xi| \leq 1/3 \\ 0 & \text{if } |\xi| > 1/3 \end{cases}.$$

Figure 2 displays in solid lines the graphs of the Fourier transforms of the Meyer scaling and wavelet functions at three scales.

There is a pair of conjugate mirror filters (h, g) associated to (ϕ, ψ) whose Fourier transforms $(\hat{h}, \hat{g})$ can be easily deduced from $(\hat{\phi}, \hat{\psi})$. $\hat{h}$ and $\hat{g}$ are thus compactly supported. As a consequence, the Meyer wavelet transform is usually implemented in the Fourier domain by a classical cascade of multiplications by $\hat{h}$ and $\hat{g}$ [2]. However, the wavelet at the finest scale is supported

[2] Here, we do not deal with an orthonormal version of the Meyer wavelet transform which can be viewed as a lapped orthogonal transform in the Fourier domain and efficiently implemented with folding.

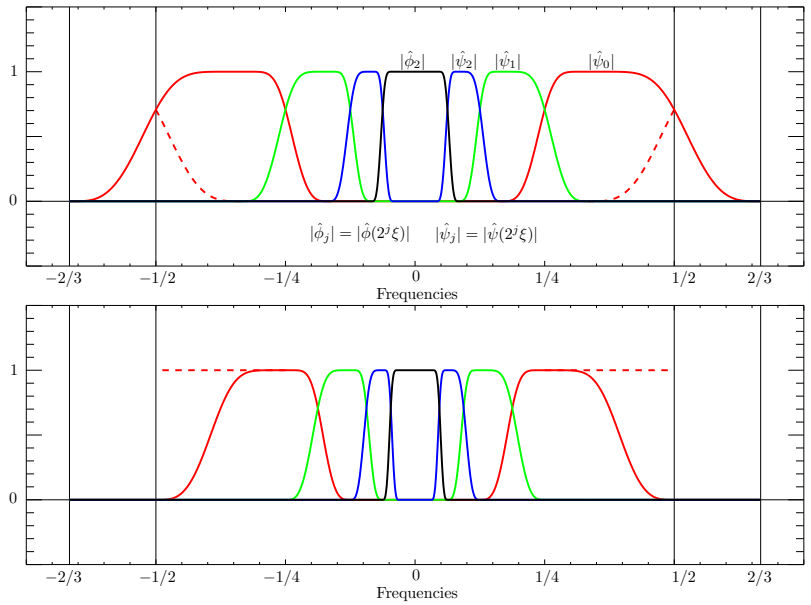


Figure 2: Meyer scaling and wavelets functions in Fourier domain. In the discrete case, we only have access to the Fourier samples inside the Shannon band $[-1/2, 1/2]$, while the wavelet corresponding to the finest scale (solid red line) exceeds the Shannon frequency band to $2/3$. Top: In the Curvelab implementation, the Meyer wavelet basis is periodized in Fourier, so that the exceeding end of the finest scale wavelet is replaced with the mirrored dashed line on the plot. Bottom : In our implementation, the wavelets are shrunk so that they fit in the $[-1/2, 1/2]$ Shannon band, and the decreasing tail of the finest scale wavelet is replaced by a constant (dashed red line) to ensure a uniform partition of the unity.

on $[-2/3, -1/6[\cup]1/6, 2/3]$, hence exceeding the Shannon band. This necessitates to know signal frequencies that we do not have access to.

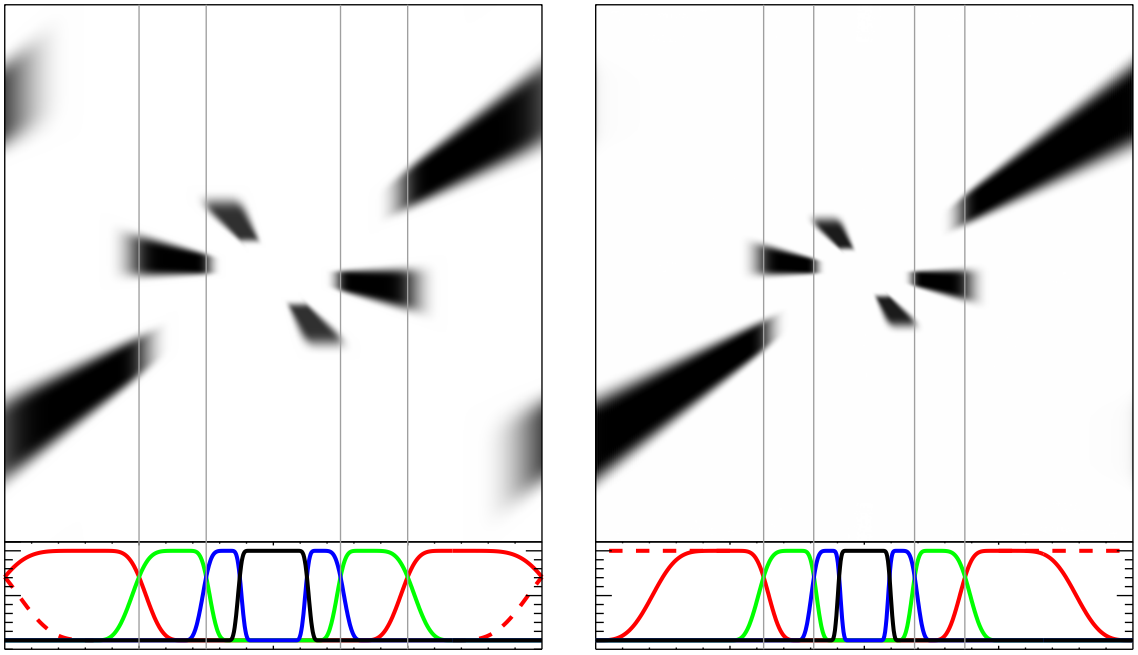
As the FCT makes central use of the FFT, it implicitly assumes periodic boundary conditions. Moreover, it is known that computing the wavelet transform of a periodized signal is equivalent to decomposing the signal in a periodic wavelet basis. With this in mind, the exceeding end of the finest scale wavelet is replaced with its mirrored version around the vertical axis at $|\xi| = 1/2$, as shown in dashed line in the top plot of Figure 2. Consequently, the support of the data to treat is $4/3$ larger than the original one, hence boosting the redundancy by a factor $(4/3)^d$ in d-D. The left part of Figure 3 shows the implication of periodization in two dimensions for the original curvelets in Fourier domain. The finest scale curvelets have the end of their decreasing window at high frequencies periodized[3].

In our implementation, we followed another path. First, toward our quest of reducing the redundancy, the supports of the scaling and wavelet functions (hence filters) were shrank by a factor of $4/3$. Furthermore, to maintain the uniform partition of unity, which plays an important role for isometry of the transform, following [20], the finest scale wavelet is modified by suppressing its decreasing tail so that the wavelet becomes a constant over $]-1/2, -1/4]\cup]1/4, 1/2]$ (see the dashed line at the bottom of Figure 2). The right part of Figure 3 shows the impact of the proposed modifications on the 2-D curvelets in the frequency domain. This strategy and the conclusions carry over to the 3-D case.

This proposed modification of the Meyer wavelets indeed reduces the redundancy of the transform (see Section 2.3). However, this comes at the cost of some changes undergone by the curvelet atoms at the finest scale. First of all, they are less sharply supported in the spatial domain than the original curvelets because of the discontinuity of their Fourier transform, while the decay of

[3]FFT and periodization have another side effect in the spatial domain this time, that curvelet atoms localized near the boundaries wrap around to the opposite edge. Indeed,In the 2-D case, mirror-extended (reflexive boundary conditions) curvelets have been proposed in [22] to mitigate the curvelet-looking artifacts due to periodization, but their redundancy is still as high as regular curvelets.

the other curvelets remain unchanged. This is one possible reason underlying the slight difference in denoising performance between the original and the modified FCT as shown in Section 2.5. Secondly, they obey a parabolic scaling but with a different constant compared to the curvelets at the other scales.



(a) (b)

Figure 3: Top : Examples of 2-D (real) curvelets in Fourier domain at three consecutive scales and different orientations, with the zero frequency at the center. From the outermost edge to the inside, a finest scale curvelet and two lower scale curvelets. (a) According to the Curvelab implementation, and (b) with our modified low-redundancy implementation. Bottom : the corresponding Fourier transforms of 1-D Meyer scaling and wavelet functions.

2.2.2 Ensuring zero-mean subbands

In the original wrapping-based FCT [2, 23], the wedges are wrapped around the origin using a simple modulo operator, which makes every point fit into a well-sized rectangular parallelepiped centered at the origin whose size is designed so as the data does not overlap with itself after wrapping. However, nothing prevents the center of the parallelepiped from receiving a significant non-zero wrapped Fourier coefficient. After an inverse FFT of the wrapped wedge, it is likely to obtain curvelet coefficient subbands with non-zero means. This is obviously unsuitable since curvelet coefficients are expected to represent high frequency content, and typical thresholding-based processing (e.g. denoising) will be hampered in such a situation. Hopefully, the size and position of the wedges are so that this misleading phenomenon is generally prevented in practice. Nevertheless, this is not guaranteed in general.

Therefore, in order to ensure zero-mean curvelet subbands, a straightforward solution is to translate each rectangular parallelepiped where a wedge has to be wrapped in such a way that the center (zero frequency) gets a true zero coefficient, i.e. a point out of the wedge support, and then to wrap the data around the translated box. Doing so, the curvelet subbands are ensured to be zero-mean valued after wrapping. Figure 4 illustrates the difference between the two wrapping strategies in 2-D for the sake of legibility. The technique extends readily to the 3-D case.

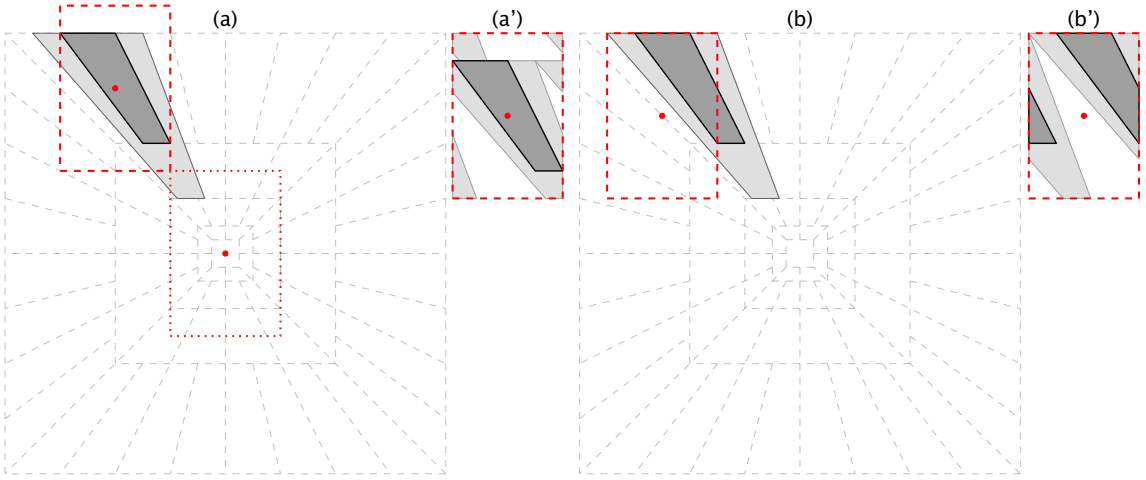


Figure 4: (a) Representation of the influence of a wedge and its overlapping region. The centered dotted rectangle corresponds to the minimal size in which the wedge will be wrapped. (a') The result of the wrapping. (b) The wrapping in a rectangle of the same size, but whose center is chosen such that the zero frequency (big dot) falls outside the support of the wedge. (b') The corresponding wedge after translation and wrapping.

2.3 Redundancy

In this section, we shall quantify analytically the redundancy reduction induced by our new FCT in any dimension d. Without loss of generality, we assume that the data object is a d-D hypercube of unit side. Let N_a be the number of angles along an edge on one face at the finest scale, for a total of N_a^{d-1} orientations on each face. Let $N_f = 2d$ be the number of faces of the d-D data hypercube.

The redundancy of the Cartesian coronization or multiscale separation assuming a dyadic frequency tiling is given by

$$\sum_{j=0}^{J} \left(\frac{1}{2^d}\right)^j, \tag{8}$$

which is upper-bounded by $R_w = \frac{2^d}{2^d-1}$. As explained in Section 2.2.1, an extra-redundancy R_{add} comes into play in the Meyer wavelet transform with the original FCT implementation, while this is avoided by ours:

$$R_{add} = \begin{cases} \left(\frac{4}{3}\right)^d & \text{Original FCT,} \\ 1 & \text{Proposed FCT.} \end{cases} \tag{9}$$

At the finest scale and on each face, there are N_a^{d-1} wedges, where the size of each of them is

$$\frac{3}{8} \times \underbrace{\frac{2}{N_a} \times \ldots \times \frac{2}{N_a}}_{d-1 \text{ times}}. \tag{10}$$

The factor 3/8 corresponds to the radial depth of the scale; see Section 2.2.1 and Figure 2. In the other orthogonal directions, a wedge has a size of $\frac{1}{N_a}$ which we double because of overlapping. The redundancy of a face at the finest scale is then

$$R_f = N_a^{d-1} \cdot \frac{3}{8} \left(\frac{2}{N_a}\right)^{d-1} = 3 \cdot 2^{d-4}. \tag{11}$$

As it can be seen, the R_f redundancy is independent of N_a, and is therefore valid at all scales. For a large enough number of scales, it can be reasonably assumed that coarsest (wavelet) scale has the

	Original FCT		Proposed FCT	
	C	W	C	W
2-D	7.11	3.56	4.00	2.00
3-D	24.38	5.42	10.29	2.29

Table 1: Redundancy of the original FCT and the proposed one in 2-D and 3-D, when wavelets (W) or curvelets (C) are used at the finest scale.

same redundancy as a curvelet subband at the same scale. Consequently, the overall redundancy of the FCT is upper-bounded by (see (8))

$$\begin{aligned} R &= N_f \cdot R_f \cdot R_w \cdot R_{add} \\ &= 3d\frac{2^{2d-3}}{2^d-1} \cdot R_{add}. \end{aligned} \tag{12}$$

In the case where wavelets are used instead of curvelets at the finest scale, the redundancy upper-bound is changed to

$$\begin{aligned} R' &= (N_f \cdot R_f \cdot (R_w - 1) + 1) \cdot R_{add} \\ &= \left(3d\frac{2^{d-3}}{2^d-1} + 1\right) \cdot R_{add}. \end{aligned} \tag{13}$$

Table 1 compares numerically the redundancy of the original and the proposed FCT in 2-D and 3-D, when wavelets (W) or curvelets (C) are implemented at the finest scale. It may be worth mentioning that for the practitioner, the memory storage requirement of the original FCT (as implemented in Curvelab) is twice larger than the one predicted by the redundancy formula. Indeed, the original curvelets are complex and real curvelets are obtained by hermitian symmetry. This explains the redundancy 40[4] claimed by [24] emphasizing the need of lower redundancy curvelets.

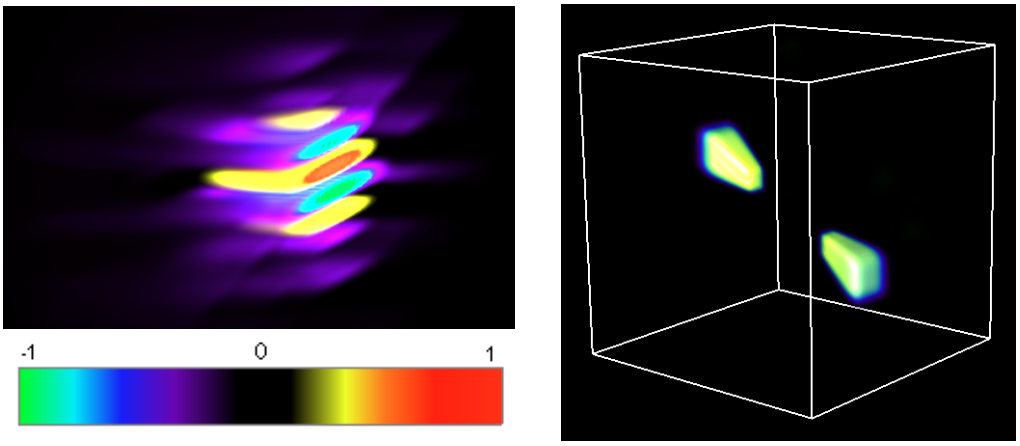


Figure 5: Left : volume rendering of a 3-D curvelet atom in the spatial domain corresponding to our implementation, cut by a vertical plane to see its inner structure. Right : the magnitude if its Fourier transform. The colorbar scale is valid only for the left image.

2.4 Properties

This section enumerates the main properties enjoyed by the FCT according to our new implementation.

[4]In fact, it should be ≈ 50 as can be read from Table 1.

- **Reduced redundancy**: with a reduction factor of $(4/3)^d$ compared to the original version. This is one of the distinctive properties of the proposed FCT, and was the main goal underlying our modifications in the first place. For example, our FCT with full curvelets at all scales is (almost) as redundant as the original one with wavelets at the finest scale. In 3-D, redundancy implied by our implementation is 2.5 times lower that the original FCT with curvelets at all scales. In short, our implementation achieves a low redundancy while maintaining the directional selectivity property at the finest scale unlike the original FCT where wavelets are advocated at the finest scale to lower the redundancy [1, 2].

- **Isometry and fast exact reconstruction**: owing to the uniform partition property of the Meyer wavelets, and the coverage property (2) of the angular window, the collection of curvelets in Fourier obtained by multiplication of the scale and angular windows also ensures a uniform partition of the unity. Therefore, with a proper normalization of the FFT (wrapping is a simple reindexing), the proposed FCT corresponds to a Parseval tight frame (PTF), i.e. the frame operator $\mathrm{CC}^* = \mathrm{I}$, where C^* is the FCT analysis operator and C its adjoint. With the PTF property, C turns out to be also the inverse transform operator associated to a fast reconstruction algorithm (each step of the forward transform is easily invertible).

- **Parabolic scaling**: by construction, the curvelets obey the parabolic scaling law with one short and two long sides $\sim (2^{-j}, 2^{-j/2}, 2^{-j/2})$. Although at the finest scale, this property is less faithful to the continuous construction compared to the original FCT (see also the discussion at the end of Section 2.2.1).

- **Non-equal ℓ_2 norm atoms**: although our FCT implements a PTF, the modified curvelets at the finest scale do not have the same ℓ_2 norm as the curvelets in the other (coarsest) scales. These ℓ_2 norms can nonetheless be calculated analytically so as to normalize the associated curvelets coefficients, which is important for instance in every processing which involves thresholding.

- **Guaranteed zero-mean subbands**: this is a consequence of the wise translation trick prior to wrapping explained in Section 2.2.2. Of course, this operation preserves isometry and ℓ_2 norms.

2.5 Comparison with Curvelab

In the sequel, the new variant of FCT is dubbed LR-FCT, where LR stands low-redundancy. In this section, we compare the 3-D LR-FCT to the original FCT as implemented in the CurveLab toolbox in terms of denoising performance over several datasets, where the noise is additive white Gaussian (AWGN). The goal is to assess experimentally how the LR-FCT version positions itself with respect to the two variants of the CurveLab FCT depending whether wavelets or curvelets are used at the finest scale.

For fair comparison, the three transforms were applied in exactly the same conditions. Our goal was not to pick up a very sophisticated thresholding/shrinkage estimator operating in the transform domain (the literature is very rich on the subject), but rather to get a fair picture about differential performance of the transforms themselves with a simple and fast statistical estimator. This is why we opted for hard thresholding with the popular 3σ threshold, where σ is the AWGN standard deviation[5]. The experiments have been carried out on different sorts of data (3-D space simulation, hyperspectral images, still and fast-moving videos), and on ten noise realizations for each triplet transform-dataset-standard deviation. Figure 6 depicts the evolution of the output peak signal-to-noise ratio (PSNR) as a function of the noise standard deviation (translated in PSNR). One can see that the LR-FCT implementation is not as good as the full redundant CurveLab version, but provides a very good tradeoff between redundancy and the denoising performance as measured by PSNR with less than 1dB lower than the CurveLab FCT (with curvelets at all scales) in the least favorable case, and 2-4 dB better than the CurveLab variant using wavelets at the finest scale. As stated in Section 2.2.1, we believe that the difference in performance between the all-curvelets

[5]The curvelet coefficients are normalized properly by the ℓ_2 norms of the associated curvelets so that this threshold is valid at all subbands. See also previous section.

original FCT and the LR-FCT can be explained by the more oscillating behavior of the LR-FCT atoms at the finest scale.

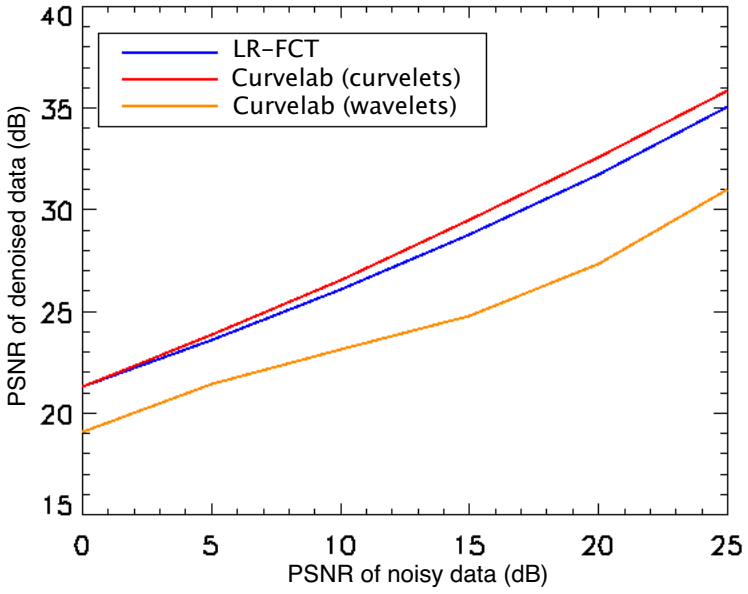


Figure 6: Mean denoising PSNR versus noise level (as measured by the input PSNR) using different FCT implementations. The denoising PSNR was averaged over ten noise realizations and several datasets. The LR-FCT is in blue. Original FCT implementation of Curvelab using curvelets (red) and wavelets (orange) at the finest scale.

3 A wide range of applications

This section does not aim to give state-of-the-art results, and instead concentrates on demonstrating how the new transform can be harnessed easily and directly to a variety of problems with reasonable – and beyond – results. Further work will be required to fine-tune these methods and optimize their performance in each specific application. All results reported in this section, especially those involving videos, are available in full resolution online at `http://arnaud.woiselle.fr/` awaiting inspection by the interested reader.

3.1 LR-FCT denoising : a good tradeoff between efficiency and memory storage

In this experiment, we compare the denoising performance using the LR-FCT to several other 3-D multiscale transforms on various types of datasets (3-D spatial data, hyperspectral images and videos). Again, the noise is AWGN and simple hard thresholding is used. The video datasets included in our experiments were the standard videos used in [] as well as *mobile*, *tempete*, and *coastguard* CIF sequences available at `www.cipr.rpi.edu`. For hyperspectral data, we used dataset from the OMEGA spectrometer on Mars Express (`www.esa.int/marsexpress`) with 128 wavelength from $0.93\mu m$ to $2.73\mu m$. Beside LR-FCT, the other transforms involved in this comparative study are: the dual-tree complex wavelet transform [25, 26], the surfacelet transform [24] and the orthogonal (decimated) and translation-invariant (undecimated) wavelet transforms.

Figures 7 and 8 show the output PSNR after denoising as a function of the input PSNR for each transform. Each point on each curve is the average output PSNR on ten noise realizations. The reader may have noticed that the wavelet results are much better here than those tabulated in [24].

The reason behind this is that unlike those authors, we used the Cohen-Daubechies-Fauveau 7/9 filterbank, which is much better for denoising. Figure8 displays the results for the hyperspectral data from Mars Express (see caption for details). From these experiments, it ca be clearly seen that the proposed LR-FCT compares very favorably to the other multiscale geometrical 3-D transforms, and is particularly better at the low PSNR regime. In a nutshell, it can be safely concluded that LR-FCT provides a very good compromise between denoising performance and memory/CPU requirements.

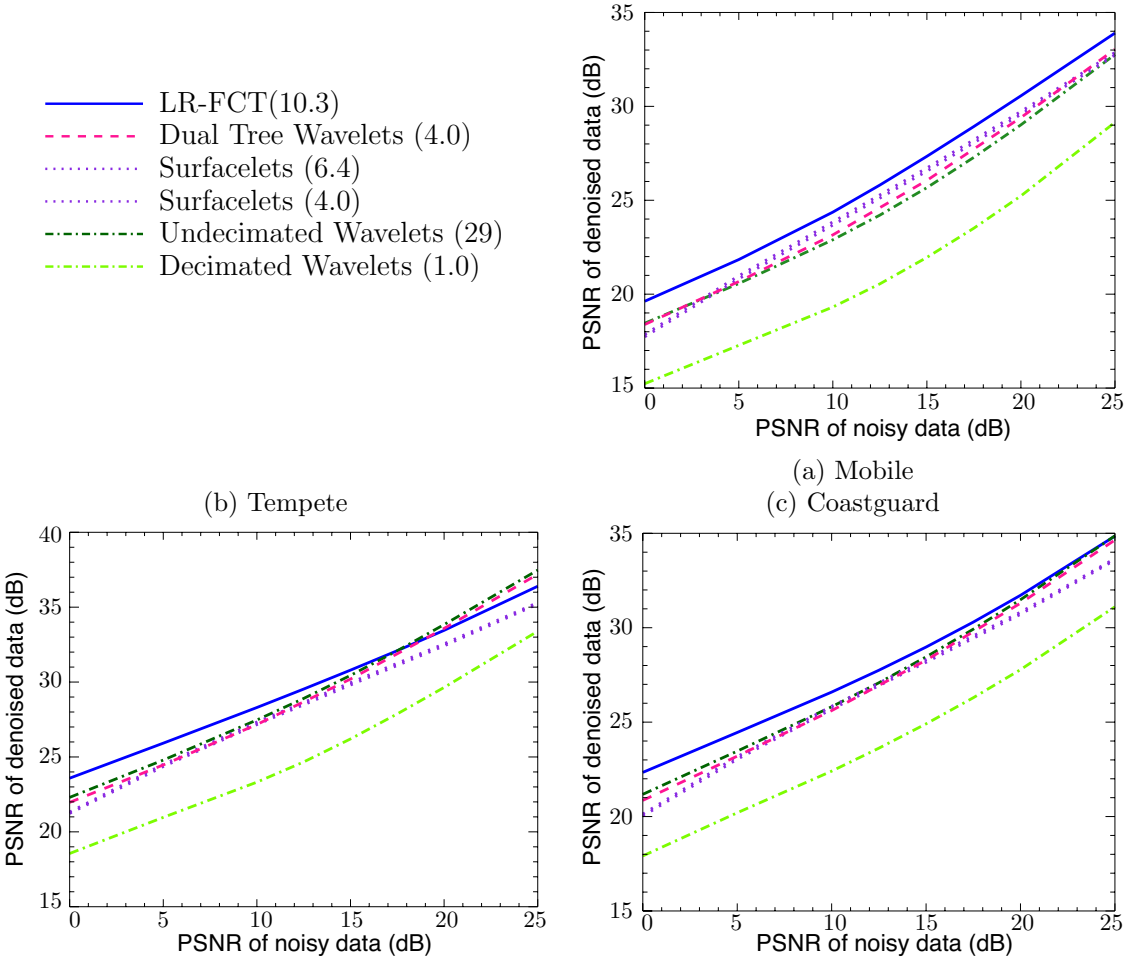


Figure 7: Output PSNR as a function of the input PSNR for three video sequences. (a) *mobile*, (b) *tempete*, and (c) *coastguard* CIF sequence. The redundancy of each transform is indicated in parentheses on the legend.

3.2 Inpainting

Inpainting is to restore missing data information based upon the still available (observed) cues from destroyed, occluded or deliberately masked subregions of the data. Inpainting has received considerable interest and excitement and has been attacked using diffusion and transport PDE/Variational principles, non-local exemplar region fill-in and sparsity-based regularization; see e.g. [8, 9] and references therein.

Let $f \in \mathbb{R}^N$ be a vectorized form of the sought after 3-D data cube which is $\sqrt{N} \times \sqrt{N} \times \sqrt{N}$, and $\mathrm{M} \in \{0,1\}^{P \times N}$, $P < N$ be a binary rectangular matrix where each of its rows is zero except at the entry where a voxel is not missing. The observed (incomplete) data g is then the result of applying the lossy operator M to f:

$$g = \mathrm{M}f + \varepsilon \ .$$

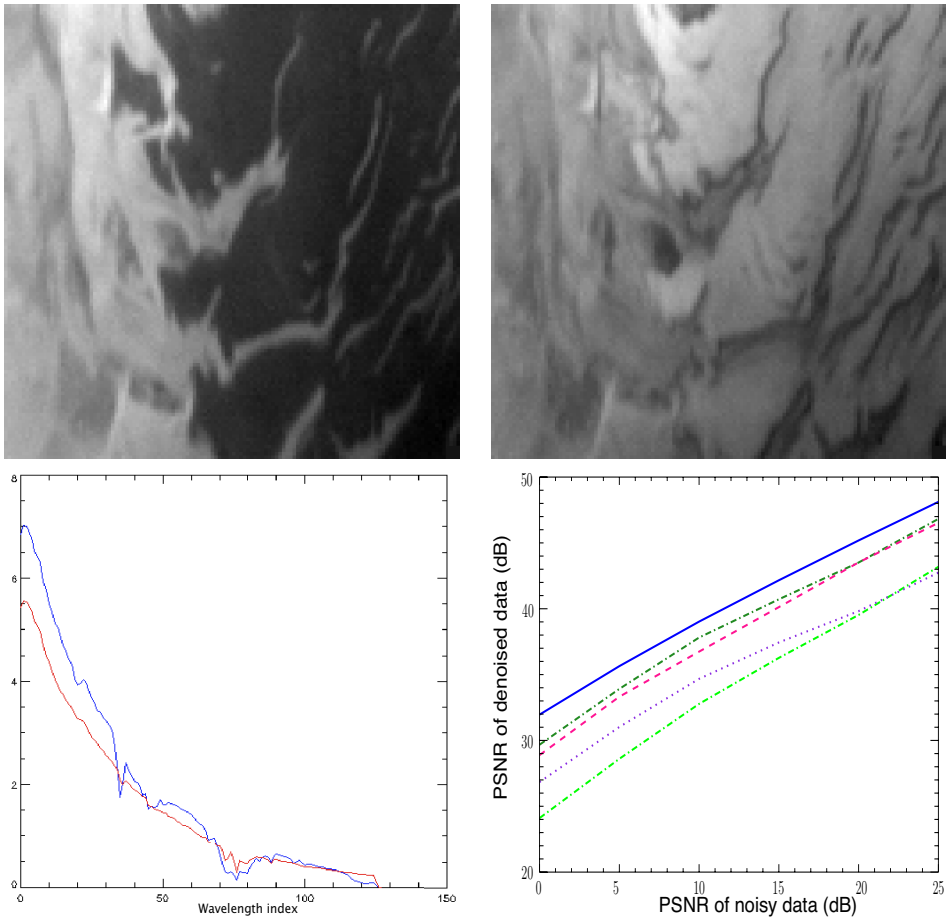


Figure 8: Top row: Mars Express observations at two different wavelengths. Bottom-left: two spectra at two distinct pixels. Bottom-right: output PSNR as a function of the input PSNR for different transforms; see Figure 7 for legend of the curves.

where ε is some noise of finite variance σ^2 that may contaminate the observed values. Restoring f from g is an ill-posed which necessitates some form of regularization to reduce the space of candidate solutions. Here, we promote solutions that are sparse in some prescribed overcomplete dictionary of atoms $\Phi \in \mathrm{R}^{N\times L}$, $L \geq N$, meaning that $x := \Phi\alpha$ (a synthesis prior) can be sparsely represented to a high accuracy by a few number of atoms in Φ. Put formally, we are seeking to solve the following optimization problem :

$$\min_{\alpha\in\mathbb{R}^L} \|\alpha\|_0 \text{ s.t. } \|g - \mathrm{M}\Phi\alpha\|_2 \leq \epsilon(\sigma) \ , \tag{14}$$

where $\|\cdot\|_0$ is the ℓ_0 pseudo-norm that counts the number of nonzero entries of its argument, and $\epsilon(\sigma)$ is the constraint radius that depends on the noise variance. This is very challenging NP-hard optimization problem, and one has to resort to alternative formulations or greedy algorithms to attempt to solve it. For instance, convex ℓ_1 relaxation could be used instead of the ℓ_0 penalty.

Here, we use the algorithm devised in [8] which can be viewed as a stagewise hybridization of matching pursuit with block-coordinate relaxation. The adjective "stagewise" is because their algorithm exploits the fact that the dictionary is structured (union of transforms $\Phi = [\Phi_1, \cdots, \Phi_K]$) with associated fast analysis and synthesis operators Φ_k^{T} and Φ_k; see [8, 27] for details. For the reader convenience, Algorithm 2 recalls the main steps of this inpainting algorithm.

In the following experiments, this algorithm was used with a dictionary containing two transforms: the LR-FCT and the undecimated discrete wavelet transform (UDWT), in order to better take into account the morphological diversity of the features contained in the data. Figure 9

Algorithm 2: Inpainting Algorithm.

Data: Observed data g and mask M.
Input: Dictionary $\Phi = [\Phi_1 \cdots \Phi_K]$, number of iterations T_{iter}, final threshold τ (e.g. 3).
begin
  Initial components $f_k^{(0)} = 0, k = 1, \cdots, K$.
  Initial residual $r^{(0)} = g$.
  Initial threshold: let $k^\star = \arg\max_k \|\Phi_k^{\mathrm{T}} g\|_\infty$, set $\lambda_0 = \max_{k \neq k^\star} \|\Phi_k^{\mathrm{T}} g\|_\infty$.
  for $t = 1$ **to** T_{iter} **do**
    for $k = 1$ **to** K **do**
      Compute marginal residuals $r_k^{(t)} = r^{(t-1)} + f_k^{(t-1)}$.
      Update kth component coefficients by thresholding $\alpha_k^{(t)} = \text{Thresh}_{\lambda_{t-1}}\left(\Phi_k^{\mathrm{T}} r_k^{(t)}\right)$.
      Update kth component $f_k^{(t)} = \Phi_k \alpha_k^{(t)}$.
    Update the inpainted data $f^{(t)} = \sum_{k=1}^{K} f_k^{(t)}$.
    Update the residuals $r^{(t)} = g - \mathrm{M} f^{(t)}$.
    Update the threshold $\lambda_t = \lambda_0 - t\left(\lambda_0 - \tau\sigma\right)/T_{\text{iter}}$.
end
Result: The estimate $f^{(T_{\text{iter}})}$ of f.

shows the inpainting result on a synthetic cerebral MRI volume available on BrainWeb [28] at `http://www.bic.mni.mcgill.ca/brainweb/` with two masks: 80% random missing voxels, and 10% missing z slices. We can see that even with 80% missing voxels, we can still see incredibly faint details in the restored anatomical structures such as in the gyri and the cerebellum.

For video inpainting, we artificially created a mask containing 5% random missing pixels on each frame, a missing line, three clusters of 12 dead pixels each, two consecutive missing frames, and a rotating swirl two-pixel wide. The top row of Figure 10 shows the central frame of the original video, as well as one of the two consecutive missing frames and one frame of the masked video. The inpainting of these missing areas was performed either with a dictionary containing the LR-FCT alone, or another dictionary containing both the LR-FCT and the UDWT. The results are displayed respectively in the middle and bottom rows of Figure 10. While inpainting using the LR-FCT alone or combining it with wavelets give similar PSNR values, visually, inpainting with both transforms yields a smoother result.

3.3 Video De-interlacing

Video is often acquired in interlaced format, meaning that every even frame contains only the even lines and similarly for odd frames. The problem of reconstructing the missing lines in the sequence is called de-interlacing and has been addressed with many dedicated methods : using simple spatio-temporal neighborhood linear interpolation [29, 30], using a directional interpolation to better preserve the edges in the images [31, 32, 33], with more complicated methods relying on motion estimation and compensation [34, 35, 36], and recently with variational inpainting [37]. In this section, it will be demonstrated that adpating the idea of sparsity-based inpainting as described in Section 3.2, and introducing a proper regularization that takes into account the special structure of the mask, will yield very promising de-interlacing results.

De-interlacing could be naively viewed as a 3-D inpainting problem, where the mask is nonzero at the observed even/odd lines in the consecutive even/odd frames; see Figure 11 for a graphical illustration. However, the major issue that rises with such a mask is that it exhibits a strong coherence with some curvelet atoms[6]. Indeed, curvelets are plate-shaped atoms, and those oriented at 45 degrees are very correlated with the inclined planes formed by the alternating even-odd lines along the time axis. One could think of excluding such atoms from the curvelet dictionary (e.g. by just zeroing their coefficients after the forward FCT in the above inpainting algorithm), or weight

[6] A similar complicating coherence phenomenon occurs also in super-resolution.

	DOI [32]	STELA [33]	STIM [37]	LR-FCT
Stockholm	31.48	30.79	**37.04**	34.17
Shields	35.30	32.00	34.62	**35.34**

Table 2: PSNR the de-interlacing results on the video sequences *stockholm* and *shields* using our method and three other ones from the literature.

the FCT coefficients differently depending on those of the mask. These solutions were tried but the results were not conclusive.

Following this discussion, we chose to introduce a regularization term acting directly in the spatial domain. In fact, we implicitly assume that the video to be restored does not contain structures that correlate strongly with the mask, otherwise there is no chance that such structures will be identifiable. Thus, a wise regularization would penalize formation of singularities (edges) in the de-interlaced video that coincide with those of the mask, which is also consistent with the way the correlation between the mask and the curvelets manifests itself. Put formally, the de-interlacing optimization problem to solve writes

$$\min_{\alpha \mathbb{R}^T} \|\alpha\|_0 + \mu \|\nabla(\Phi\alpha) \odot \nabla f_{\text{mask}}\|_1 \text{ s.t. } \|g - \mathrm{M}\Phi\alpha\|_2 \leq \epsilon(\sigma), \tag{15}$$

where f_{mask} is the video representing the mask (Figure 11) vectorized in R^N, ∇ is the spatio-temporal discrete gradient operator that produces a vector field in $\mathbb{R}^{N\times 3}$, $\odot$ is the Hadamard product, and for $u = (u_x, u_y, u_t) \in \mathrm{R}^{N\times 3}$, $\|u\|_1 = \sum_i |u[i]|$ where $|u[i]| = \sqrt{u_x[i]^2 + u_y[i]^2 + u_t[i]^2}$. It is not difficult to see that the above penalty can be rewritten as a weighted TV semi-norm:

$$\|\nabla(\Phi\alpha) \odot \nabla f_{\text{mask}}\|_1 = \|\mathrm{W}_{\text{mask}} (\nabla(\Phi\alpha)) \|_1 = \|\Phi\alpha\|_{\text{TV},\mathrm{W}_{\text{mask}}} ,$$

where W_{mask} is a (diagonal) weighting operator whose entries are given by ∇f_{mask}, and which acts on vector fields in $\mathbb{R}^{N\times 3}$ by component-wise multiplication by its entries. This yields yet another nice interpretation of our penalty, i.e. promote candidate solutions that are smooth at the locations where the mask changes abruptly, which is exactly the goal we were pursuing.

The resulting optimization problem we end up with is even more challenging that the original inpainting problem. Inspired by the work in [38] and [8], we slightly modified Algorithm 2 by incorporating a (sub)gradient-descent correction step with respect to this penalty, namely by adding a step of the form:

$$f^{(t)} \leftarrow f^{(t)} + \mu\gamma \mathrm{div}\left(\mathrm{W}_{\text{mask}} G\left(\mathrm{W}_{\text{mask}}(\nabla f^{(t)})\right)\right)$$

just after the update of $f^{(t)}$, where $G(.)$ is a subgradient of the ℓ_1 norm in $\mathbb{R}^{N\times 3}$, and γ is the descent step-size. Although this update is quite heuristic, the obtained algorithm works fairly well in practice. Further work is needed to solve exactly the above problem (or rather its convex relaxation) with theoretical guarantees. We will be left to a future work.

Figures 12, 13, 14, 15 show close-ups of the de-interlacing results on the test sequences *stockholm* and *shields*, and compare them the methods proposed in [32, 37, 33]. We will refer to them respectively as DOI, STIM, and STELA. The quantitative PSNR results are grouped in Table 2.

Although the proposed method is not intended to compete with methods specifically designed for de-interlacing in terms of computational cost, this experiment at least provides a proof of concept that sparsity-based regularization with the LR-FCT transform is efficient for de-interlacing. The results clearly show that the proposed method is at the level of the state-of-the-art both visually and in terms of PSNR. For some videos, it even outperforms the other methods with visually more pleasant quality and less artifacts than dedicated methods (see e.g. Figure 13).

3.4 Morphological Component Analysis

The concept of morphological diversity, introduced by Starck *et al.* [39, 38] is based on the fact that complex signals are linear superposition of more elementary signals (the so-called morphological components). There, such complex signals cannot be efficiently (sparsely) represented with a

dictionary built upon a single transform, but rather but rather with an overcomplete dictionary built by amalgamating several sub-dictionaries. Once the sub-dictionaries are identified, solving a sparse decomposition problem in such an overcomplete dictionary yields as a by-product a separation of the structural/morphological components. The Morphological Component Analysis (MCA) algorithm proposed [39] was proposed to achieve this goal.

A key to the success of this separation process is a good choice of the sub-dictionaries. Each sub-dictionary has to be chosen such that its atoms match the shape of the (geometrical) structures they are intended to sparsify, while leading to a non-sparse (or at least not as sparse) representation on the other signal content. For instance, the isotropic wavelet transform [27] is a very good candidate for representing isotropic singularities. On the other hand, the curvelet transforms have plate-like atoms well adapted for representing smooth 3-D objects with smooth singularities along smooth surfaces. Using a dictionary built as a union of the 3-D isotropic wavelets and LR-FCT sub-dictionaries, we exemplify the MCA algorithm on a toy 3-D dataset, containing parts of 3-D shells, and 3-D Gaussians. The results in the noiseless and noisy cases are shown in Figures 16-17.

4 Conclusion

We have proposed a new variant of the fast curvelet transform in d-D with a special emphasis on 3-D, lowering its redundancy from 25 to about 11 in 3-D, providing a versatile tool to use in a wide range of inverse problems. We have shown good results in video and spectral data denoising, video and MRI inpainting, sparse component separation, and extended the sparse inpainting approach to the de-interlacing problem. Following the philosophy of reproducible research, a C++/Matlab toolbox implementing the proposed LR-FCT and reproducing the experiments in this paper are made available for download at *http://arnaud.woiselle.fr*.

Aknowledgements

We wish to thank Michael Elad for constructive discussions. We also wish to thank Luis Garrido for giving us the de-interlaced sequences presented in [37] to which we compare in Section 3.3.

References

[1] L. Ying, L. Demanet, and E. Candès, "3D Discrete Curvelet Transform," in *Proceedings of Wavelets XI conference, San Diego*, July 2005.

[2] E. Candès, L. Demanet, D. Donoho, and L. Ying, "Fast discrete curvelet transforms," *SIAM Multiscale Model. Simul.* **5**(3), pp. 861–899, 2006.

[3] E. Candès and D. Donoho, "Curvelets – a surprisingly effective nonadaptive representation for objects with edges," in *Curve and Surface Fitting: Saint-Malo 1999*, A. Cohen, C. Rabut, and L. Schumaker, eds., Vanderbilt University Press, (Nashville, TN), 1999.

[4] J. Starck, D. Donoho, and E. Candès, "Very high quality image restoration by combining wavelets and curvelets," in *SPIE conference on Signal and Image Processing: Wavelet Applications in Signal and Image Processing IX, San Diego, 1-4 August*, A. Laine, M. Unser, and A. Aldroubi, eds., SPIE, 2001.

[5] J.-L. Starck, E. Candès, and D. Donoho, "The curvelet transform for image denoising," *IEEE Transactions on Image Processing* **11**, pp. 670–684, June 2002.

[6] G. Hennenfent and F. Herrmann, "Seismic denoising with nonuniformly sampled curvelets," *IEEE Computing in Science and Engineering* **8**, pp. 16–25, May 2006.

[7] J.-L. Starck, F. Murtagh, E. Candès, and D. Donoho, "Gray and color image contrast enhancement by the curvelet transform," *IEEE Transactions on Image Processing* **12**(6), pp. 706–717, 2003.

[8] M. Elad, J.-L. Starck, P. Querre, and D. Donoho, "Simultaneous cartoon and texture image inpainting using morphological component analysis," *Applied and Computational Harmonic Analysis* **19**, pp. 340–358, 2005.

[9] M. J. Fadili, J.-L. Starck, and F. Murtagh, "Inpainting and zooming using sparse representations," *The Computer Journal* **52**(1), pp. 64–79, 2007.

[10] J. Starck, M. Nguyen, and F. Murtagh, "Deconvolution based on the curvelet transform," in *International Conference on Image Processing*, pp. II: 993–996, 2003.

[11] J. Starck, M. Nguyen, and F. Murtagh, "Wavelets and curvelets for image deconvolution: a combined approach," *Signal Processing* **83**, pp. 2279–2283, 2003.

[12] D. Donoho, "Wedgelets: nearly minimax estimation of edges," *The Annals of Statistics* **27**(3), pp. 859–897, 1999.

[13] J. Romberg, M. Wakin, and R. Baraniuk, "Multiscale wedgelet image analysis: fast decompositions and modeling," in *in IEEE Int. Conf. on Image Proc. 2002*, **3**, pp. 585–588, Jun. 2002.

[14] V. Chandrasekaran, M. Wakin, D. Baron, and R. Baraniuk, "Surflets : A sparse representation for multidimensional functions containing smooth discontinuities," *Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on* , July 2004.

[15] V. Chandrasekaran, M. Wakin, D. Baron, and R. Baraniuk, "Representation and compression of multidimensional piecewise functions using surflets," *IEEE Transactions on Information Theory* **55**, pp. 374–400, Jan. 2009.

[16] Y. Lu and 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.

[17] E. Candès and D. Donoho, "New tight frames of curvelets and optimal representations of objects with C^2 singularities," *Communications on Pure and Applied Mathematics* **57**(2), pp. 219–266, 2003.

[18] J. Ma and M. Hussaini, "Three-dimensional curvelets for coherent vortex analysis of turbulence," *Appl. Phys. Letters* **91**, October 2007.

[19] F. Herrmann and G. Hennenfent, "Non-parametric seismic data recovery with curvelet frames," *Geophysical Journal International* **173**(1), pp. 233–248, 2008.

[20] J. Starck, A. Bijaoui, B. Lopez, and C. Perrier, "Image reconstruction by the wavelet transform applied to aperture synthesis," *Astronomy and Astrophysics* **283**, pp. 349–360, 1999.

[21] S. Mallat, *A wavelet tour of signal processing*, Academic Press, 1998.

[22] L. Demanet and L. Ying, "Curvelets and wave atoms for mirror-extended images," in *SPIE Wavelets XII conference*, August 2007.

[23] L. Demanet, *Curvelets, Wave Atoms, and Wave Equations*. PhD thesis, California Institute of Technology, May 2006.

[24] Y. Lu and M. Do, "Multidimensional directional filter banks and surfacelets," *IEEE Transactions on Image Processing* **16**(4), pp. 918–931, 2007.

[25] N. Kingsbury, "Complex wavelets for shift invariant analysis and filtering of signals," *Applied and Computational Harmonic Analysis* **10**, pp. 234–253, May 2001.

[26] I. Selesnick, "The double-density dual-tree DWT," *IEEE Transactions on Image Processing* **52**, pp. 1304–1314, May 2004.

[27] J.-L. Starck, F. Murtagh, and M. Fadili, *Sparse Signal and Image Processing: Wavelets, Curvelets and Morphological Diversity*, Cambridge University Press, Cambridge, UK, 2010.

[28] K. Remi, A. Evans, and G. Pike, "MRI simulation-based evaluation of image-processing and classification methods," *IEEE Transactions on Medical Imaging* **18**(11), p. 1085, 1999.

[29] A. M. Tekalp, *Digital Video Processing*, Prentice Hall, 1995.

[30] G. D. Haan and E. B. Bellers, "Deinterlacing: An overview," *Proc. IEEE* **86**, pp. 1839–1857, Sep. 1998.

[31] T. Doyle, "Interlaced to sequential conversion for EDTV applications," in *Proc. 2nd Int. Workshop Signal Processing of HDTV*, pp. 412–430, 1990.

[32] H. Yoo and J. Jeong, "Direction-oriented interpolation and its application to de-interlacing," *IEEE Trans. Consum. Electron.* **48**, pp. 954–962, Apr. 2002.

[33] H. Oh, Y. Kim, Y. Jung, S. Ko, and A. Morales, "Spatio-temporal edge-based median filtering for deinterlacing," in *IEEE International Conference on Consumer Electronics*, pp. 52–53, IEEE; 1999, 2000.

[34] F. Wang, D. Anastassiou, and A. Netravali, "Time-recursive deinterlacing for IDTV and pyramid coding," *Signal Processing: Image Communication* **2**(3), pp. 365–374, 1990.

[35] S. Yang, Y. Jung, H. Young, and R. Park, "Motion compensation assisted motion adaptive interlaced-to-progressive conversion," *IEEE transactions on circuits and systems for video technology* **14**(9), pp. 1138–1148, 2004.

[36] G. D. Haan and E. B. Bellers, "De-interlacing of video data," *IEEE Trans. Consum. Electron.* **43**, pp. 819–825, Aug. 1997.

[37] C. Ballester, V. C. M. Bertalmío, L. Garrido, A. Marques, and F. Ranchin, "An inpainting-based deinterlacing method," *IEEE Transactions on Image Processing* **16**, pp. 2476–2491, Oct. 2007.

[38] J. Starck, M. Elad, and D. Donoho, "Image decomposition via the combination of sparse representations and a variational approach," *IEEE Transactions on Image Processing* **14**, pp. 1570–1582, October 2005.

[39] J. Starck, M. Elad, and D. Donoho, "Redundant multiscale transforms and their application for morphological component analysis," *Advances in Imaging and Electron Physics* **132**, pp. 287–348, 2004.

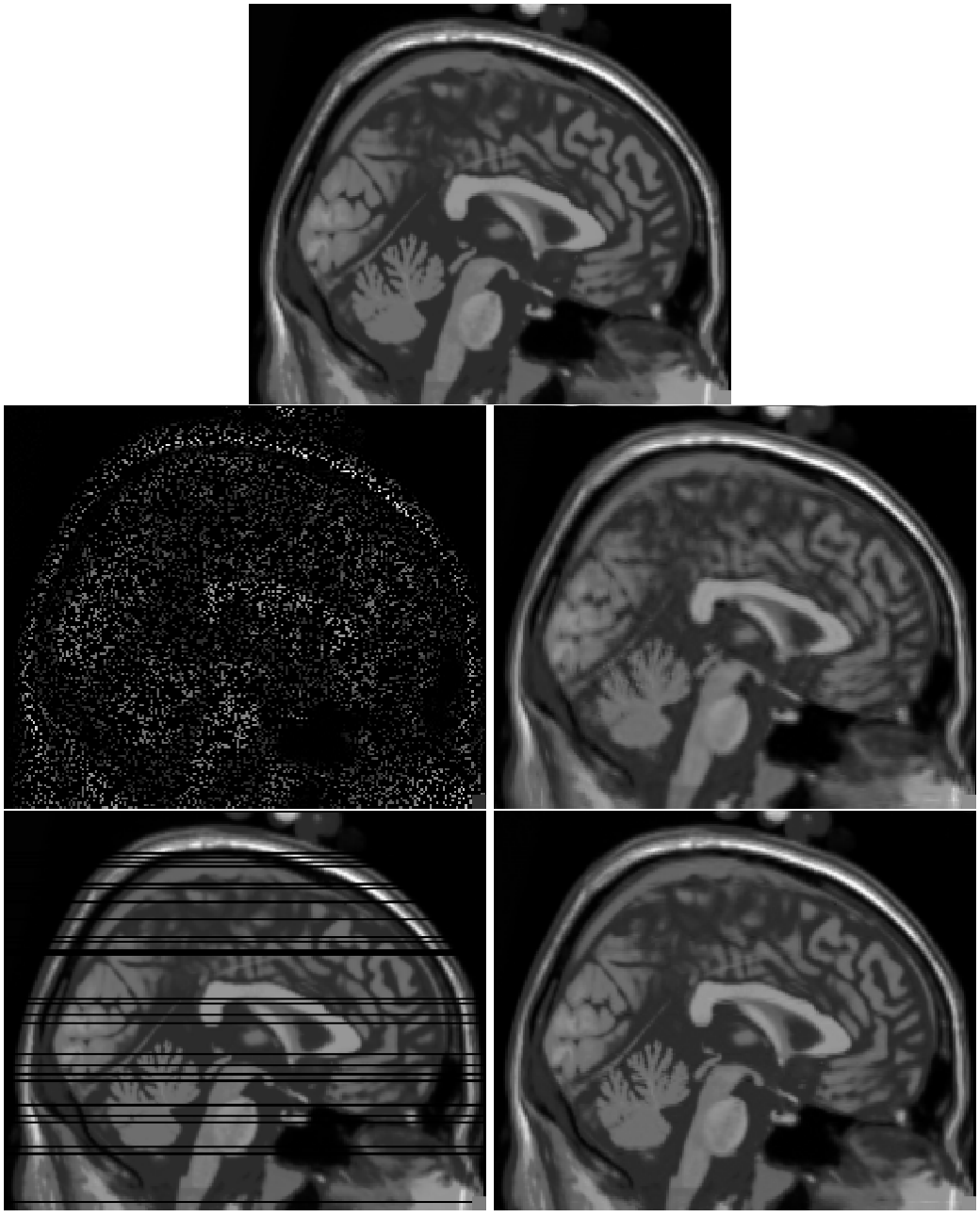


Figure 9: Top : a sagittal $((y, z))$ slice of the original synthetic MRI volume from BrainWeb[28]. Left column : the data with missing areas, random 80% missing voxels, and 10% missing z slices. Right : inpainting results with a LF-FCT+UDWT dictionary.

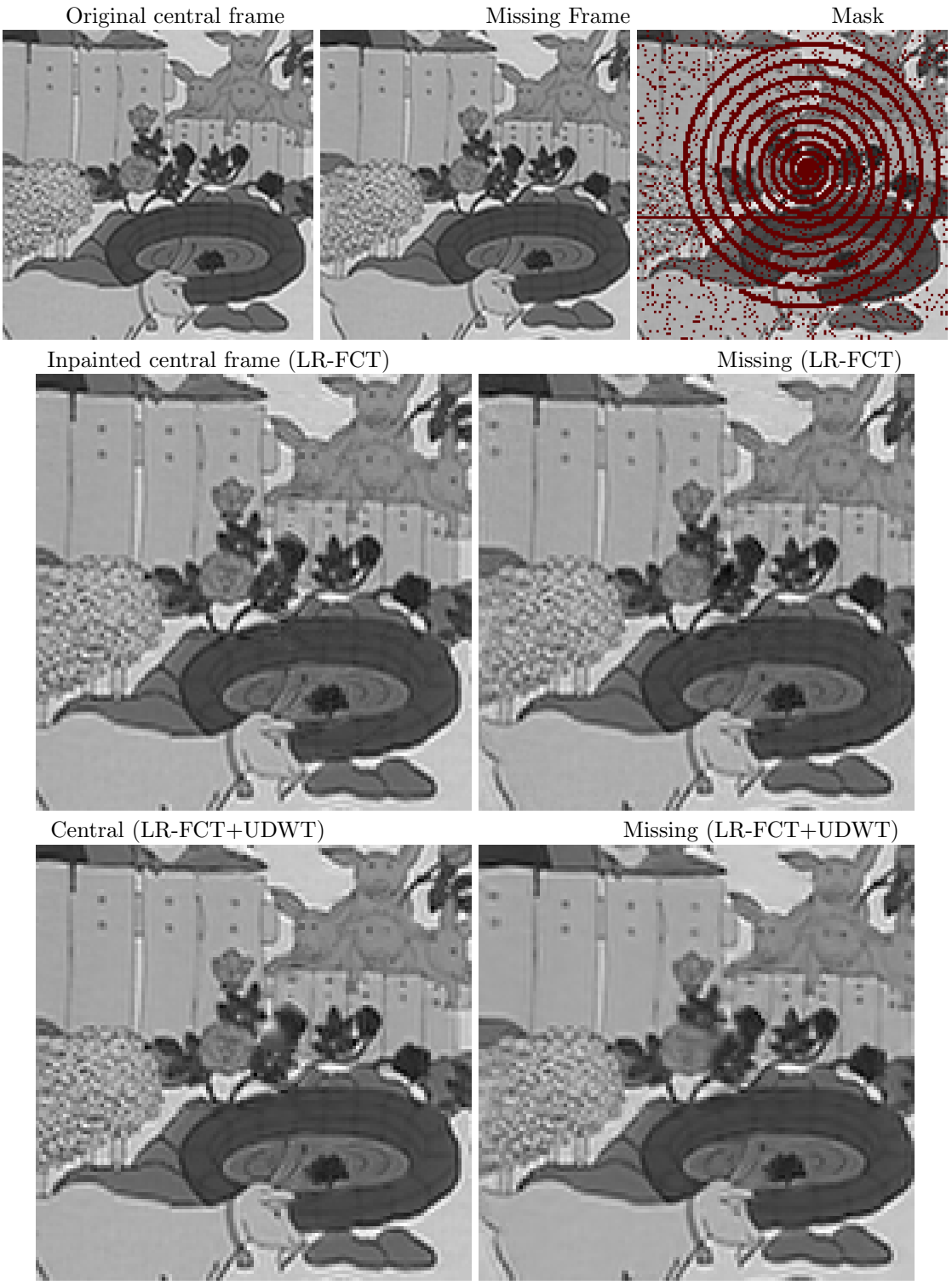


Figure 10: Inpainting the *mobile* video. Top row : the central frame of the original video, a missing frame, and a frame of the masked video. Middle row : central and missing frames inpainted using only LR-FCT. Bottom row : using LR-FCT and UDWT.

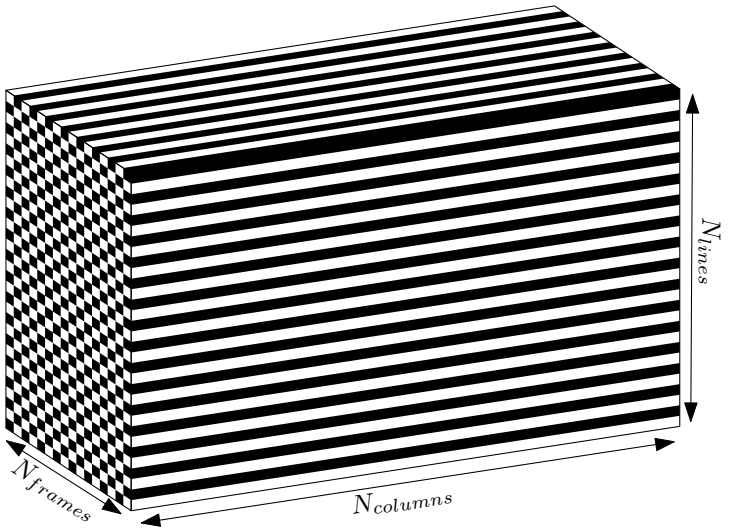


Figure 11: Example of interlacing mask on 16 frames of a video, black and white indicating known and lost pixels.

Figure 12: Close-ups of the centeral part of a frame from the *shields* video sequence de-interlaced using different methods. Top : Original, and de-interlaced result using STELA. Bottom : results using STIM and proposed method.

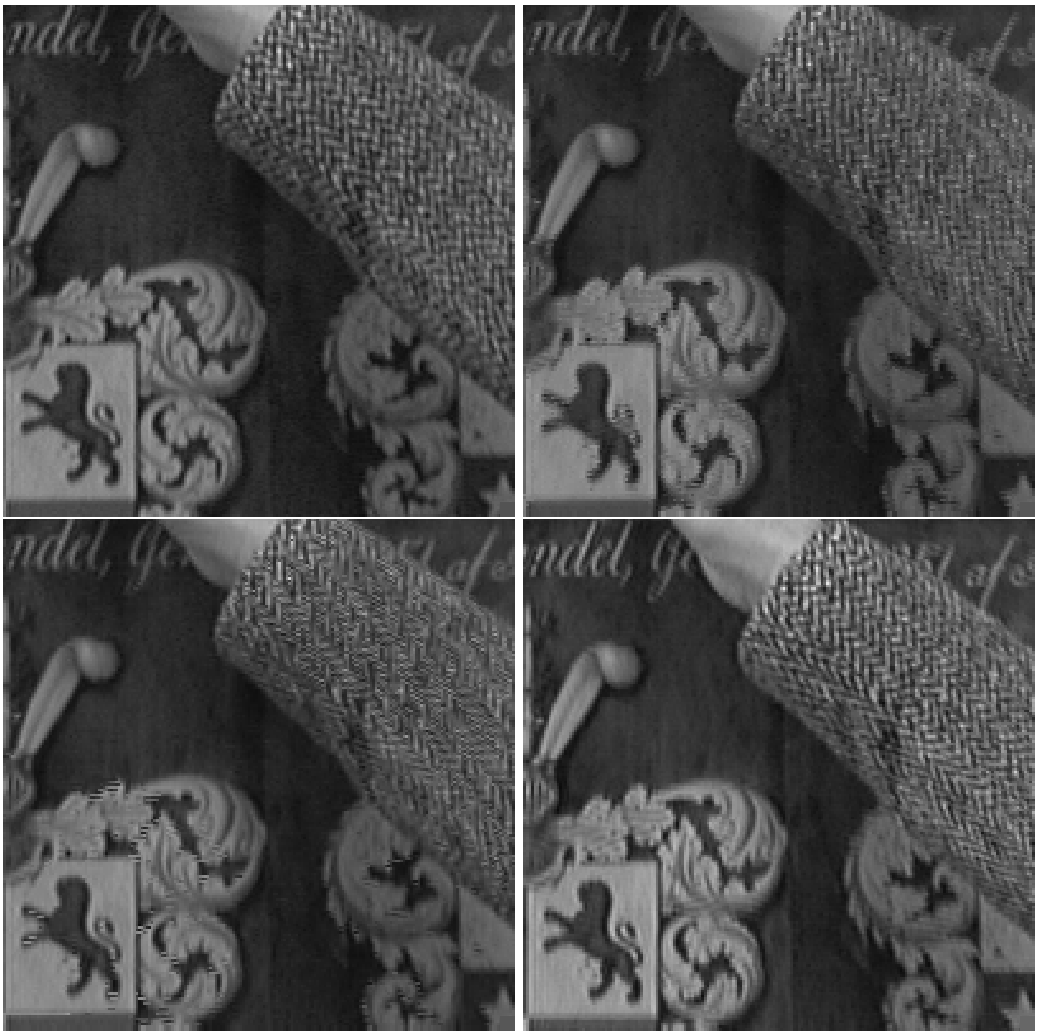


Figure 13: Close-ups of a fast-moving part of the *shields* video sequence de-interlaced using different methods. Top : Original, and de-interlaced result using STELA. Bottom : results using STIM and proposed method.

Figure 14: Close-ups of the stairs in the *stockholm* video sequence de-interlaced using different methods. Top : Original, and de-interlaced result using STELA. Bottom : results using STIM and proposed method.

Figure 15: Close-ups of a building in the *stockholm* video sequence de-interlaced using different methods. Top : Original, and de-interlaced result using STELA. Bottom : results using STIM and proposed method.

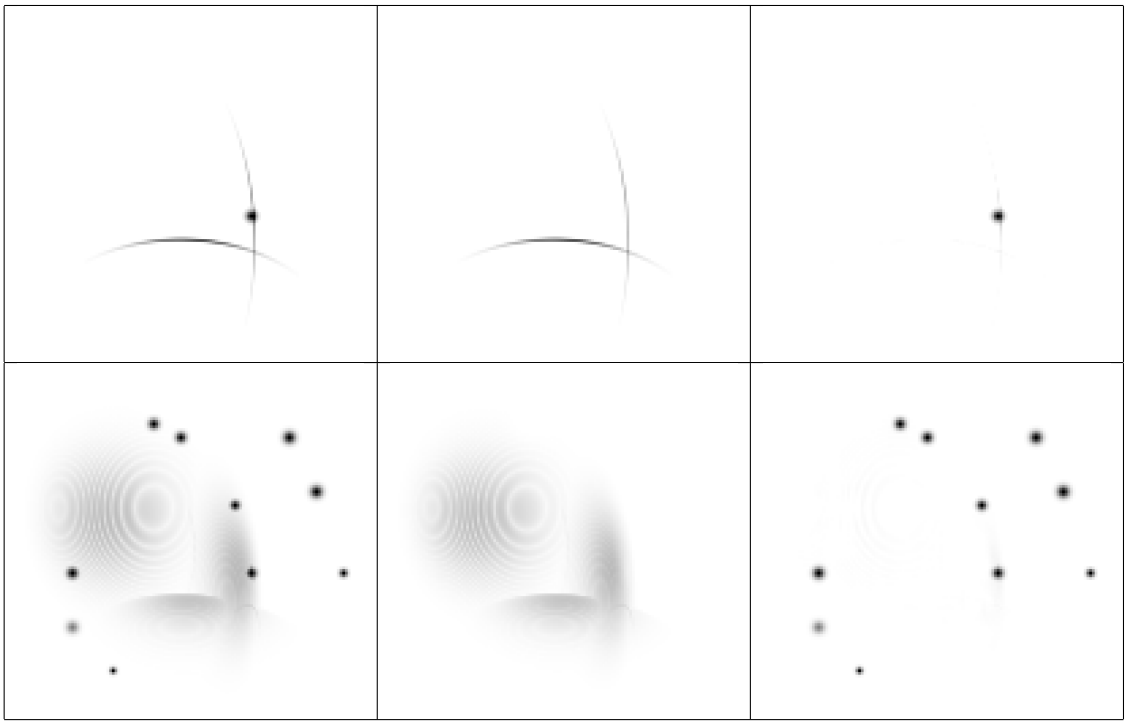


Figure 16: Morphological component separation results from a data cube containing pieces of 3-D shells and 3-D gaussians. From left to right : original dataset, and morphological components extracted with the LR-FCT and the isotropic wavelet transform. Top : a 2-D slice of each 3-D cube. Bottom: maximum intensity projection view.

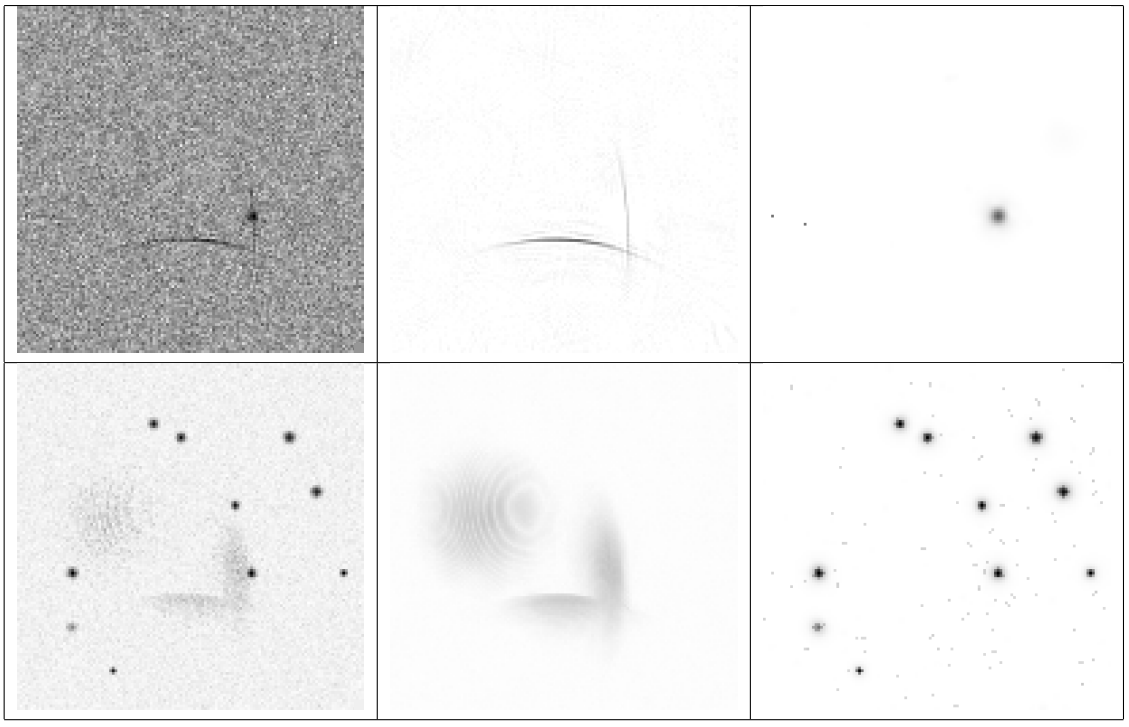


Figure 17: The same as Figure 16, with an additive white Gaussian noise. The two components are jointly separated and denoised.