MATHEMATICS

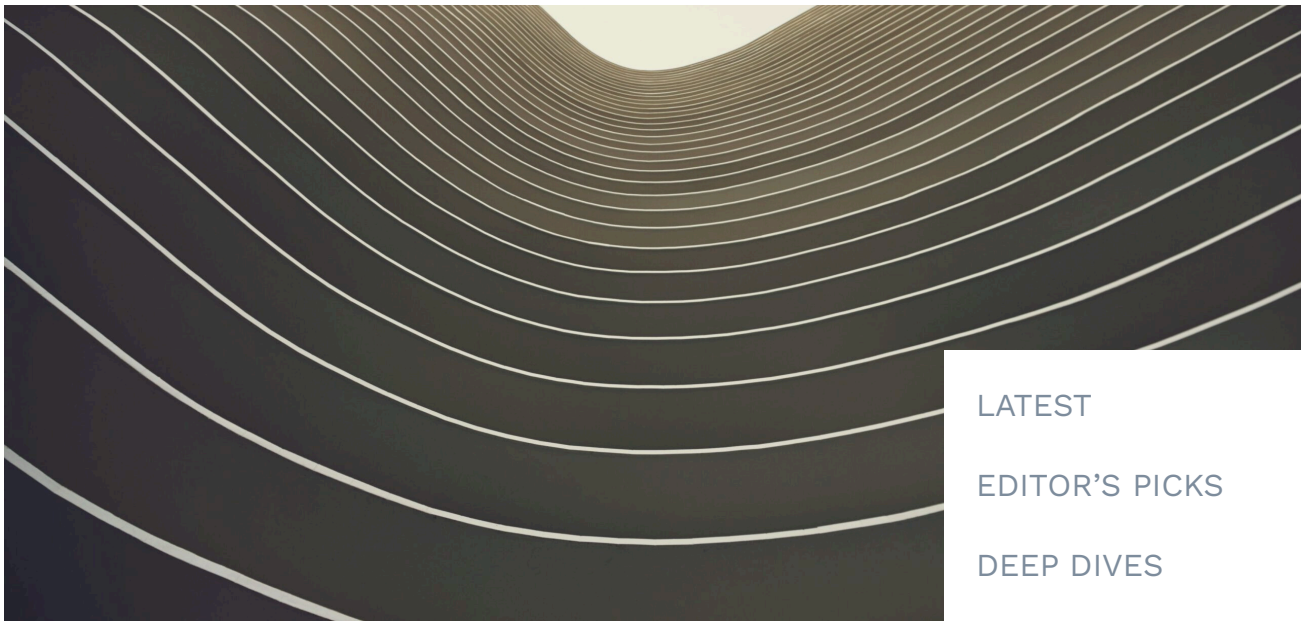
Desmystifying Curvelets

Learn what curvelets are what they can be used for. I
about signal processing and computing computing.

Carlos Costa, Ph.D.
Mar 22, 2023 16 min read

Signal-processing deep dive

Learn what curvelets are, how they are built an
be used for



Edifício Niemeyer, Belo Horizonte. Photo by Matheus Frade on

Introduction

Curvelets are multiscale, oriented, non-adaptive of images and multi-dimensional signals. If som didn't make sense, you're in the right place.

Developed in the early 2000s by Emmanuel Can Donoho [1] in the flurry of wavelet-transform rel processing boom, curvelets were designed to so that plagued previous alternatives. Wavelets ger Fourier transform by using arbitrary – but spec functions instead of complex exponentials. Like transform, they can be readily extended to 2D b repeatedly applying the transform over multiple Constructing 2D wavelets with this naïve approa issues when representing edges that are not exa exactly vertical. In practice, transforms that hav

edges can cause "blocky" artifacts in strongly compressed images. Many transforms suffer from the same fate, including the discrete cosine transform (DCT) which powers the ubiquitous JPEG format (see Figure 1). JPEG2000 which relies on the wavelet transform to improve on JPEG, still suffers from the same blocky artifacts.

Figure 1. "Blocky" artifacts appearing in strongly compressed JPEG images for b
Credit to: Shlomi Tal (CC BY-SA 3.0).

Curvelets don't have these issues – *they can pre*
edges. They also exhibit some nice properties, s
optimal representation of wave-like phenomena
conservation, unitarity (having its adjoint/transp
as its inverse), among others.

Today, curvelets are used in a variety of image-p
including denoising, compression, inpainting, sm
the geophysical community, it has become a po
for several tasks including adaptive subtraction,
preconditioning, among others. In the medical c
been used for segmentation, diagnosis, etc.

While it is true that deep learning has quickly di
algorithms for many of these tasks, curvelets (a
transforms) still have their use. Whereas deep le

possibility to create *adaptive*, multiscale representations of signals, curvelets already offer that predictably, without requiring training, and optimally for many types of signals. One may use curvelets directly as a substitute when no training data is available, or even employing them alongside deep learning methods, thereby reducing model complexity and data requirements.

In this deep dive, we will go over the building bl curvelet transform, focusing on the intuition bel to provide a starting point for users interested i applications which use curvelets. Find all the cc tutorial in the Curvelops repository.

Before Curvelets: the Fourier Transf

To understand the curvelet transform, we need understand the Fourier transform. The 2D FFT (F Transform) tells us what kind of spatial frequen The more energy on the larger (away from the or wavenumbers (spatial frequencies), the faster th along a certain direction. In this section we will to find this direction, as well as how to quantify

Let's start by creating images which vary rapidly particular direction, but not at all in the perpen These will be helpful in understanding what the spectrum (another name for Fourier domain, the domain after applying FFT) is telling us. We will modulating a cosine in the direction normal to a $\cos \theta$).

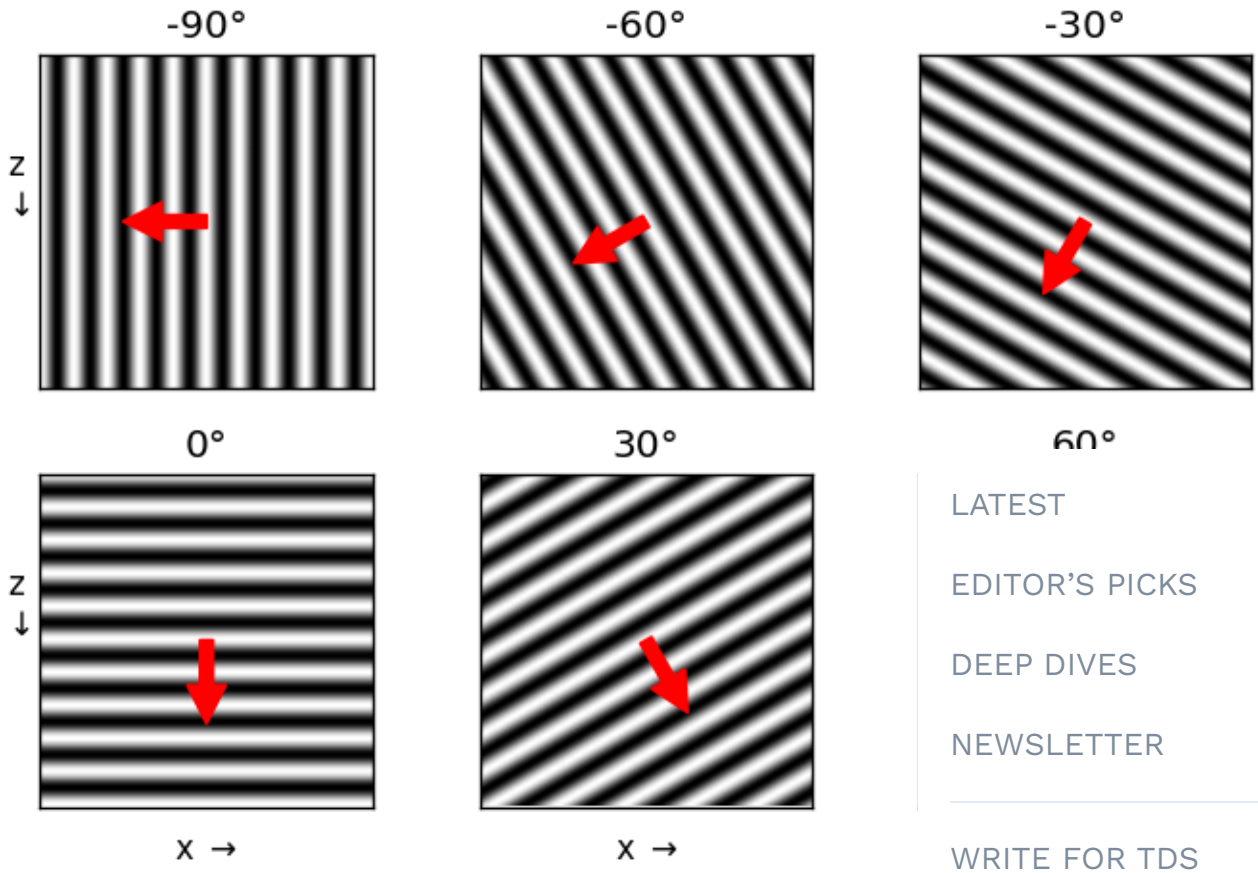


Figure 2. 2D monochromatic images, with direction of change in each image der
own work.

Each plot in this image has an arrow pointing to
direction of maximum variation v. Along the dire
perpendicular to it, the signal does not vary at a

This won't always happen, signals can vary in m
direction at once. The question is, which directi
we find out? Cue the FFT transform. Let's take
images, say the one at 45°, and apply the 2D FF

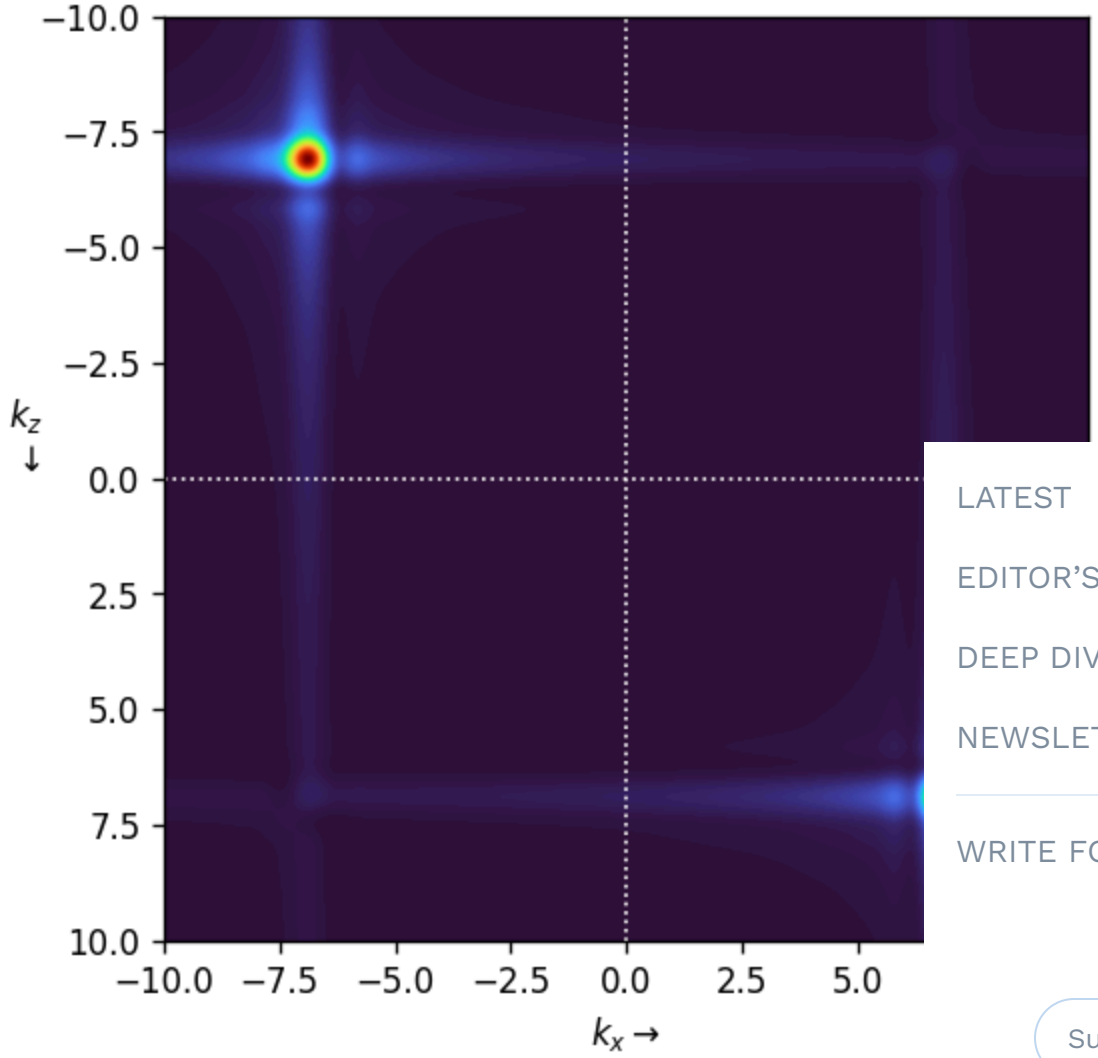


Figure 3. 2D FFT of the monochromatic signal which varies most at the 45° c

In the Fourier domain, we can see that our signa
A property of the 2D Fourier transform for real i
that its Fourier spectrum symmetric about the c
above plot, the top-left quadrant carries the sar
the bottom-right quadrant; the top-right carries
information as the bottom-left quadrant. So we
"ignore" the negative frequencies for one axis. O
to choose the "fastest" axis, in this case the ver

> WARNING: This is not true for complex signals
> change the math much, just the visual interpr

only use real signals here.

So we can scan the (positive kz) k-space to identify a single peak. Its corresponding kx and kz coordinates are given by: [6.931, 6.931]. Not particulaly interesting, until we normalize this vector to [0.707, 0.707] and compare it with the original direction of maximum variation: [0.707, 0.707]. Up to a constant, the vectors are the same! Let's try this for all images created above.

1. Compute the 2D FFT
2. Find the maximum amplitude location in k-
3. (kx-max, kz-max) is the direction of maximu the input

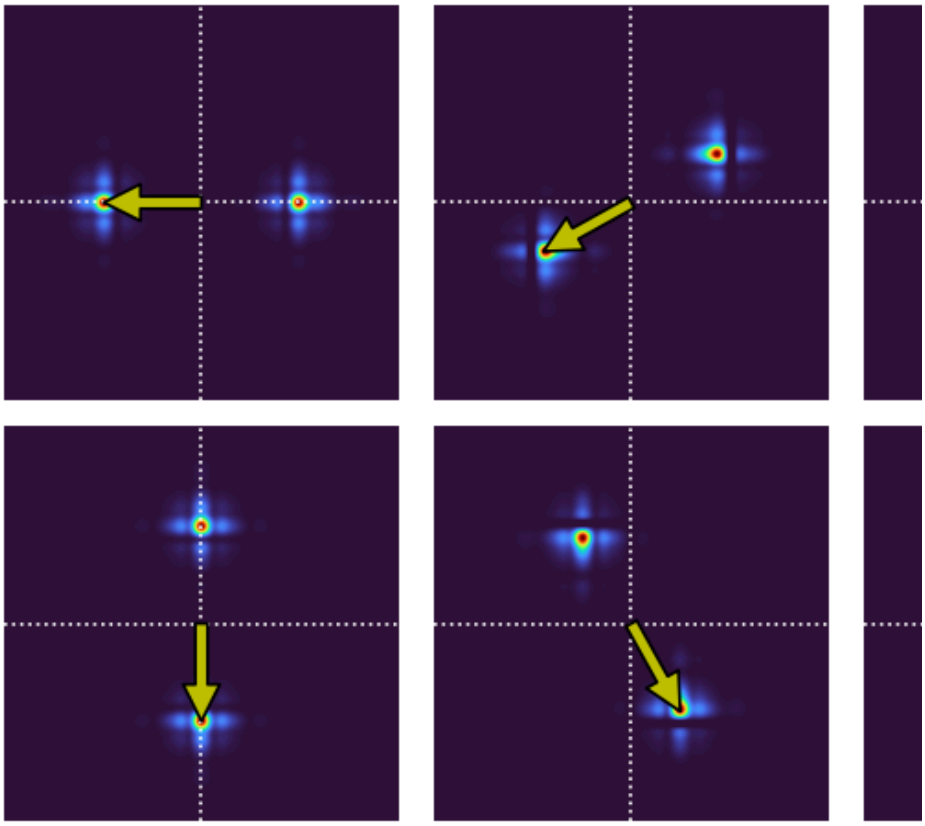


Figure 4. 2D FFT of the monochromatic signals in Figure 2. Cred

Notice how these follow exactly the direction of maximum change in the data. Indeed, if we overlay these normalized vector onto the data-space images, we get:

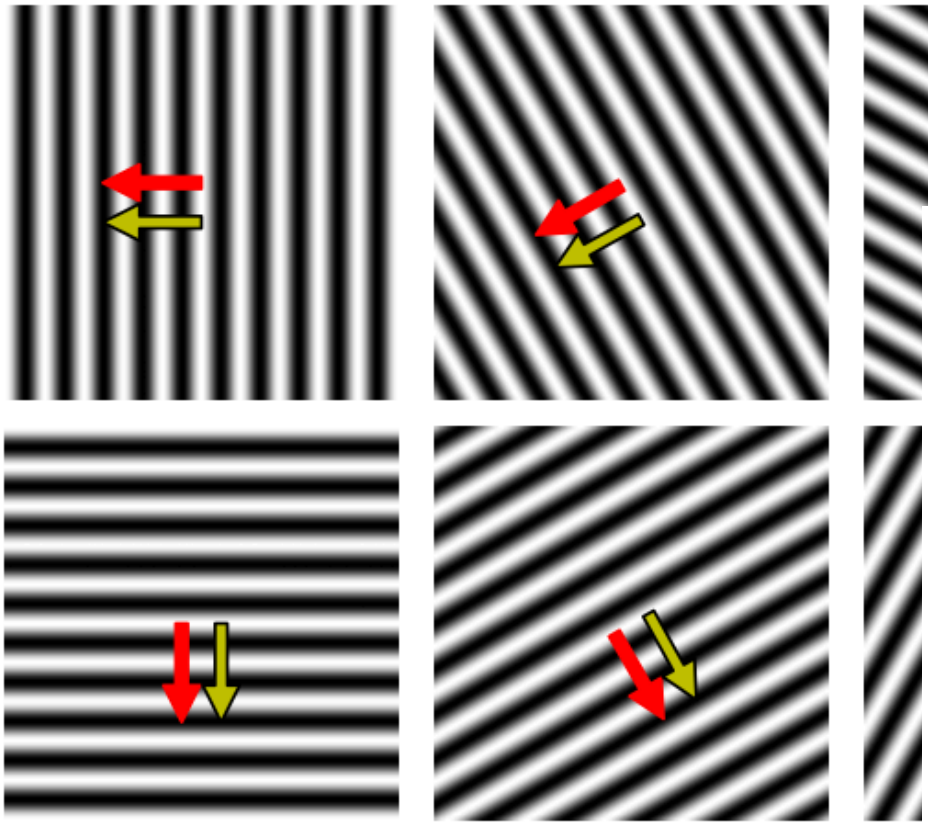


Figure 5. Monochromatic signals in Figure 2. Red arrow: direction of maximur direction of maximum variability estimated from 2D FFT. Credi

Looks good! What does this have to do with the transform? Well, imagine we try applying the sar following signal:

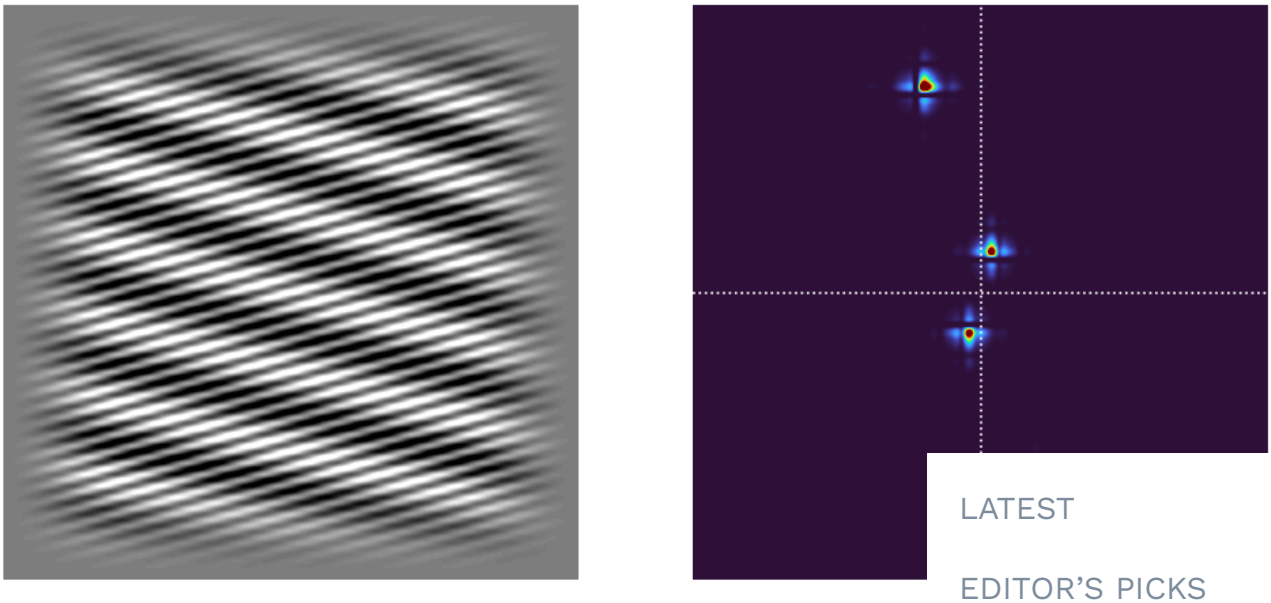


Figure 5. Bichromatic signal in spatial (left) and Fourier (right) domain

We can see that there is a low-frequency signal background, corresponding to the -15° compone is a high frequency component in another direct apply our k-max algorithm, because the low-fre weaker, it won't get picked up. We will think tha direction is 15°. But the FFT spectrum has all th what can we do with it?

Building the Curvelet Transform

Separating scales

The curvelet transform handles this wealth of ir separating the signal at different *scales*, which a concentric regions in the k-space domain. These encompass frequencies that are similar. Let's ap to the example above:

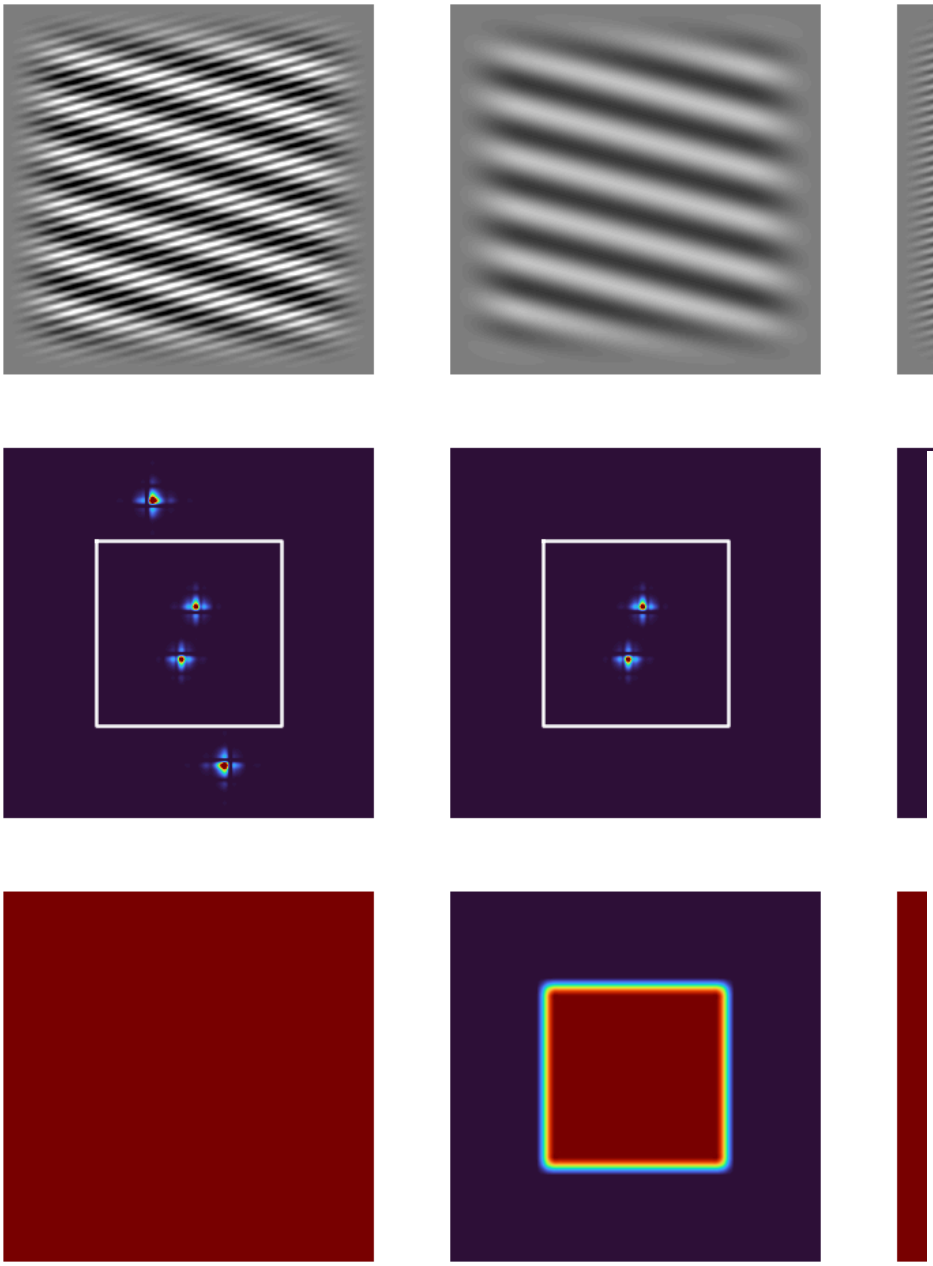


Figure 6. Original bichromatic signal (left column). Low-passed signal (center
(right column). Spatial domain (top row). Fourier domain (middle row). Mask
generate the signals in the top row (bottom row). Deep purple denotes zero va
unity. Credit: own work.

The top-left signal is the original input signal. Th
(second row) is its the Fourier spectrum. The im
(third row) is the "mask" applied to the Fourier s
reconstruct the original signal. It is entirely red,
ones, that is, no change to the spectrum.

The second column, depicts signals from the first region, constructed by a low-pass filter defined by its mask (third row, middle column). Where it zeroes the signal, the mask is purple. Finally, the third column depicts the second region, constructed by a high-pass filter which is `1 - lowpass`.

In the curvelet transform, the number of scales is the first parameter. A choice of scales = 2 would create a division very similar to the one above (but with a different, sl smoothing). The more scales there are, the mor curvelet transform coefficients will be able to s with different frequencies.

And what will these coefficients be? In this case similar to the inverse transform of the low-pass filters, that is, the two rightmost panels in the t caveat (in addition to the smoothing) is that the scale does not require the same sampling as th Indeed, the Nyquist frequency (highest frequenc sampling can represent) of the lowpass signal is original Nyquist frequency. This is clear from the lowpass signal removes all frequencies above N Therefore we could actually *double* the samplin component compared to that of the original sig general rule, and we will see that the coarsest s can be sampled at 2^m times the original samplir equal to the number of scales minus 1.

The finest scale (in this example, scale = 1) canr same way, as its highest frequency remains the original signal.

For visualization's sake, we can apply this resampling to the lower frequency scale to get an idea of what the curvelet coefficients would look like for a scales = 2 and no additional subdivisions (more about this soon).

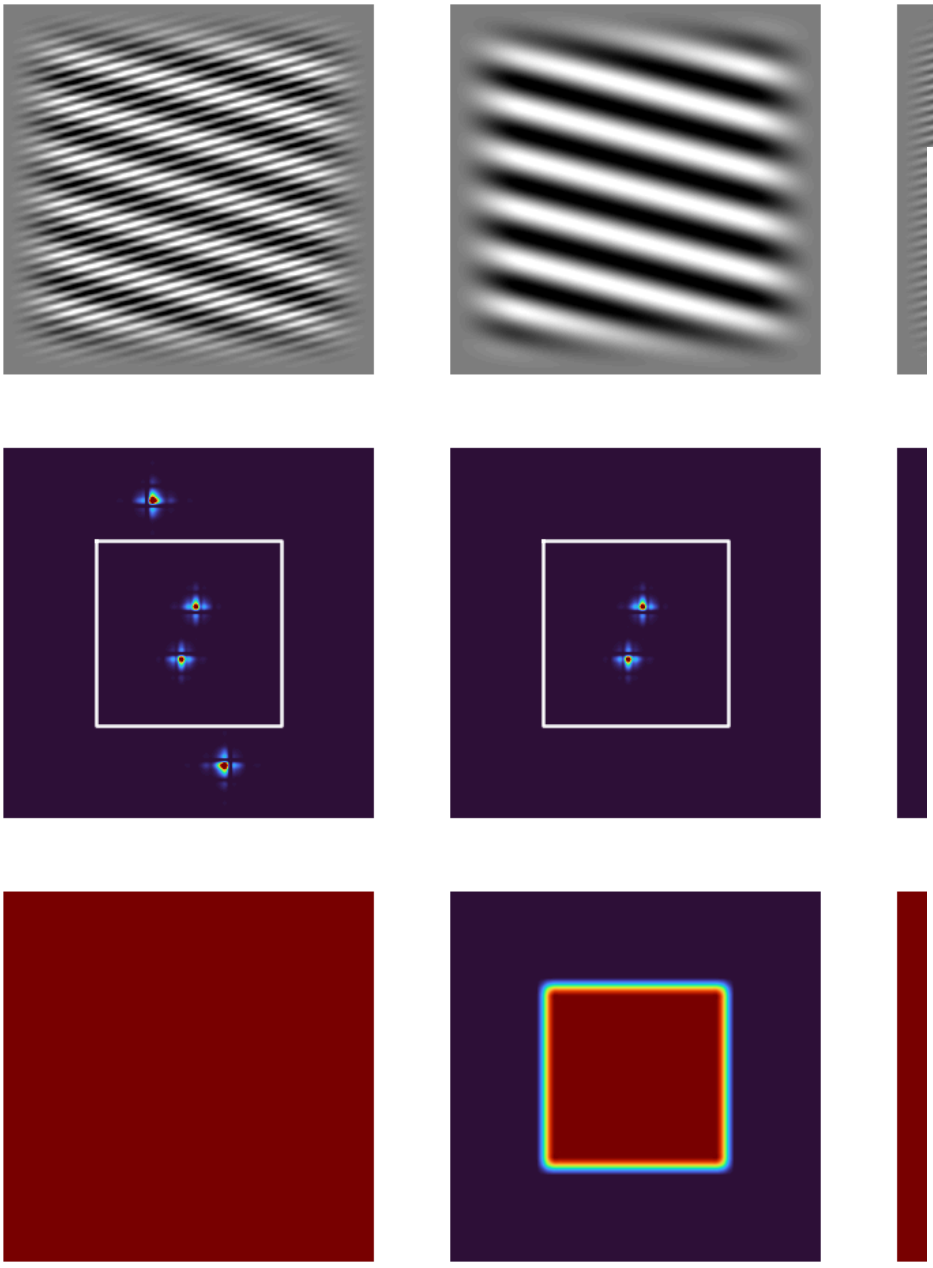


Figure 7. Original signal (left column). Coarsest curvelet scale using scales = curvelet scale (right column). All other panels like Figure 6. Note: this curvelet t finest scale. Credit: own work.

Comparing this plot to the original plot without notice that the signal is accurately represented

strength of the resampled signal is larger. This is because the Fourier transform used to "return" to the spatial domain now has a normalization factor which is smaller than the original. Therefore the signal will be twice as strong (2 = √2 × √2, one square root per dimension, in this case two). This is also a general rule of the curvelet transform and Fourier transforms.

Let's apply this decomposition to another exampl

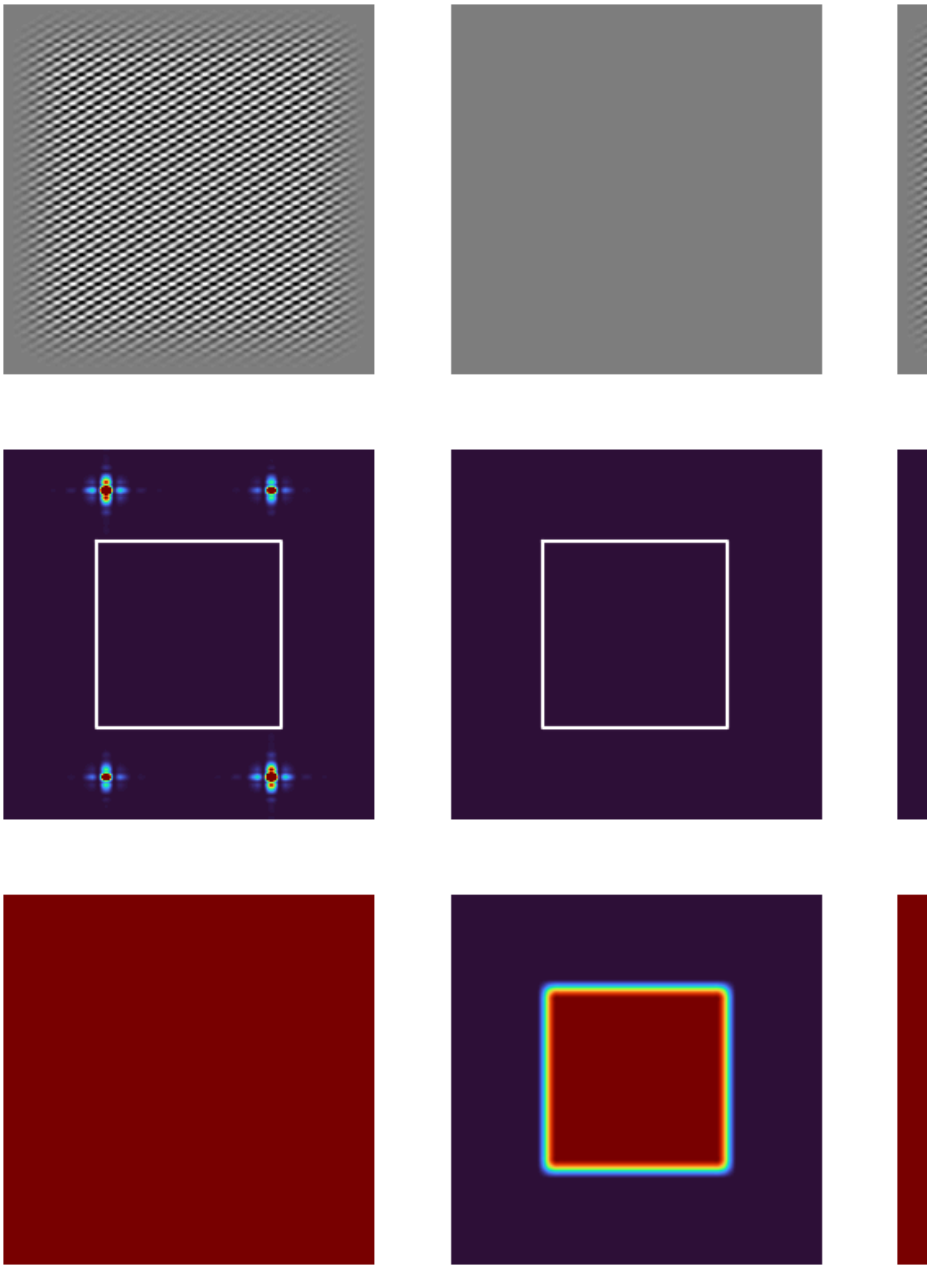


Figure 8. Tetrachromatic signal composed of four signals: two spatial freq orientations. Panels like Figure 7. Credit: own work

Separating Dips

While we have separated in *scale*, we still are not able to distinguish the two separate events, one at -30° and one at 30°. We need another type subdivision, one which splits a signal (of similar frequencies) further, into "subsignals" of similar *direction* (or *dip*, in geophysical lingo). Starting with the second scale (the one after the coarsest), we will further split it scale into *wedges*. These are approximately angular sectors in the *f*. The coarsest scale (scale = 0) is exempt from th because the origin can't be assigned to any part

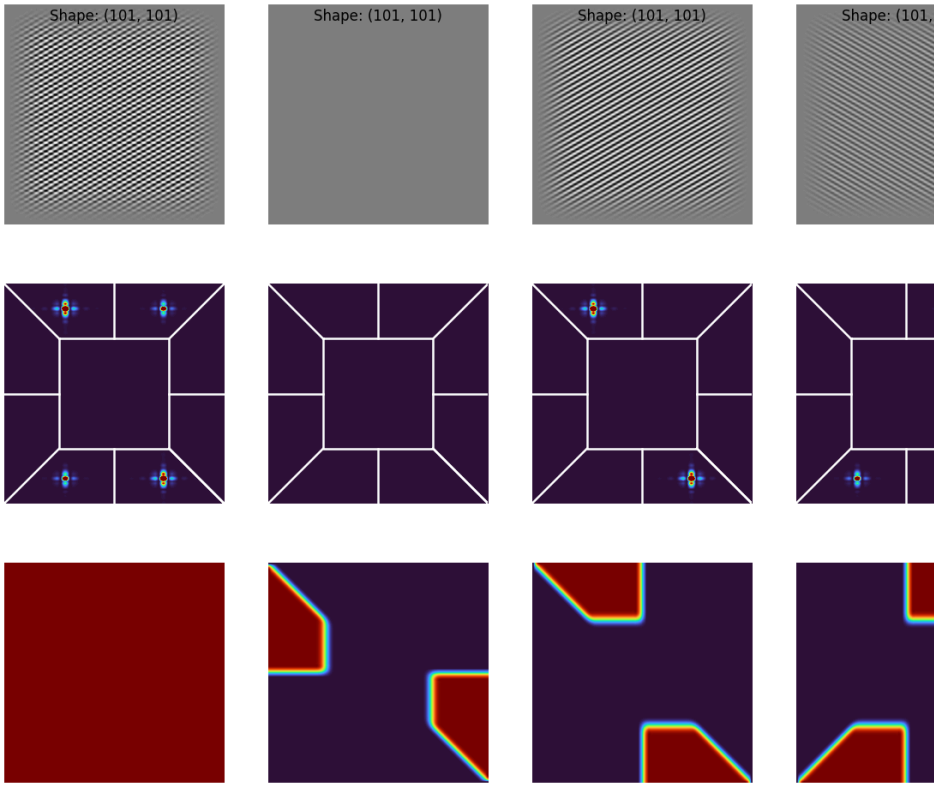


Figure 9. Original tetrachromatic signal (leftmost column). Signal separated by "
(center-left to rightmost columns). Rows like Figures 6, 7 and 8. C

We can see that we have been able to separate two constituent components. For this signal, tw are identically zero. The signals in the first row, second column, are essentially what the curvele

would look like if we applied the curvelet transform with the parameters scales = 2 and wedges = 8.

One small caveat is that we are actually only showing **four wedges**. This is again because of the symmetry of the Fourier domain for real signals. For complex signals, we would have to treat each wedge (without its symmetrical counterpart) separately.

Fast Discrete Curvelet Transform

We now have all ingredients to understand how Transforms are implemented in practice. For exa have some kinks to iron out. One of them is the sum every wedge mask, we don't get an exact va every grid point in the k-space domain. This me "lose" or "create" signal. A more pressing issue i the wedges were transformed back to the spati non-optimal way: we used a full-sized FFT trans transform a small sliver of the k-space domain. curvelet coefficients all shaped the same shape image (in that example, 101 × 101).

Moreover, an important property of the continuc transform is that performing the steps backwar output image *from* curvelet coefficients, should exact same image that was used to obtain these the first place. Mathematically, the transformati is, its adjoint is equal to its inverse. Our implem respect this property.

All these issues can be fixed. One construction which solves all these problems is the Fast Discrete Curvelet Transforms, developed by Emmanuel Candès, Laurent Demanet, David Donoho and Lexing Ying. Its most famous implementation is provided by the CurveLab package, which the Curvelops package wraps via Python (disclaimer: I developed Curvelops). Curvelops thereby provides a linear operator interface relying on PyLops (disclaimer: I am one of the core devs of this lib

Let's explore the FDCT by applying it to the sign compare the curvelet coefficients with what we before.

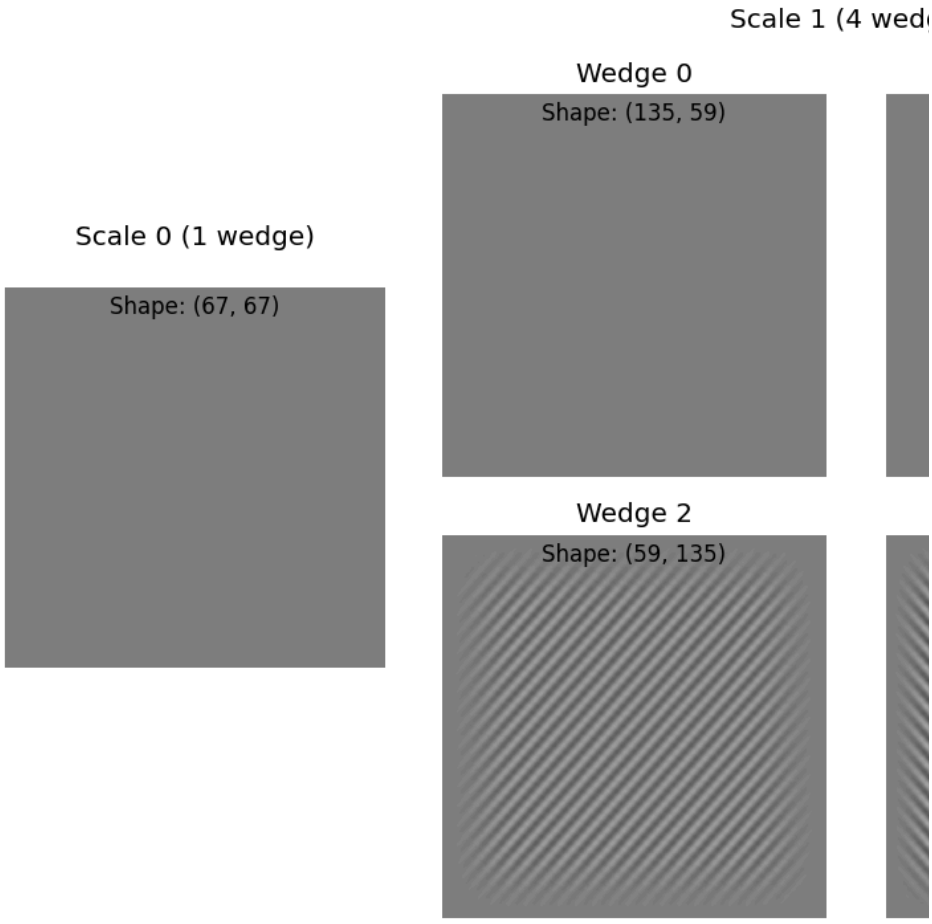


Figure 10. (Real) curvelet coefficients of the tetrachromatic signal. (

These look very similar to what we obtained before, with a few differences. First, as mentioned before, the algorithm actually outputted 8 wedges in the second scale; we merged them due to the symmetry of the Fourier space for real input signals. Second, we can see that the shapes are not the same as the input signal. The coarsest scale is about ⅔ of the input signal. On the second scale, not only the shapes are different, but the aspect ratio is also different. This is because different directio different sampling. The FDCT handles all these i way that is performant and that respects the ur the continuous transform.

Now we are ready to understand the parameters

- **nbscales** (no default, minimum 2): Number more scales, the finer the details the curvel capture. The downsides to more scales are tapering issues. Both of these issues can be setting `allcurvelets=False`.
- **nbangles_coarse** (default: 16, minimum 8, m of 4): Number of angles/wedges at the seco Remember that the coarsest scale is never value is only specified for the second scale double every two scales (counting as if the `nbangles_coarse` wedges). So by setting a `nbs` `nbangles_coarse=8`, the second scale will hav third 16 (as we have seen 2 scales by now), the fourth and last, 32 (as we have seen fou Downsides to increasing this parameter is a

and tapering issues. Mitigate them also with `allcurvelets=False`.

- **allcurvelets** (default: True, boolean): Controls whether we perform wedge subdivision on the finest (last) scale. We did this in the latter example, but not in the one before that. From a practical point of view, despite defaulting to True, you should set this options to False unless there is a strong reason not to.

Examples of the Fast Discrete Curvelet Transfor

Now that we understand these parameters, let's
the some curvelet coefficients from different pa

Figure 11. Python "two-snakes" Logo. Credit: Python Software

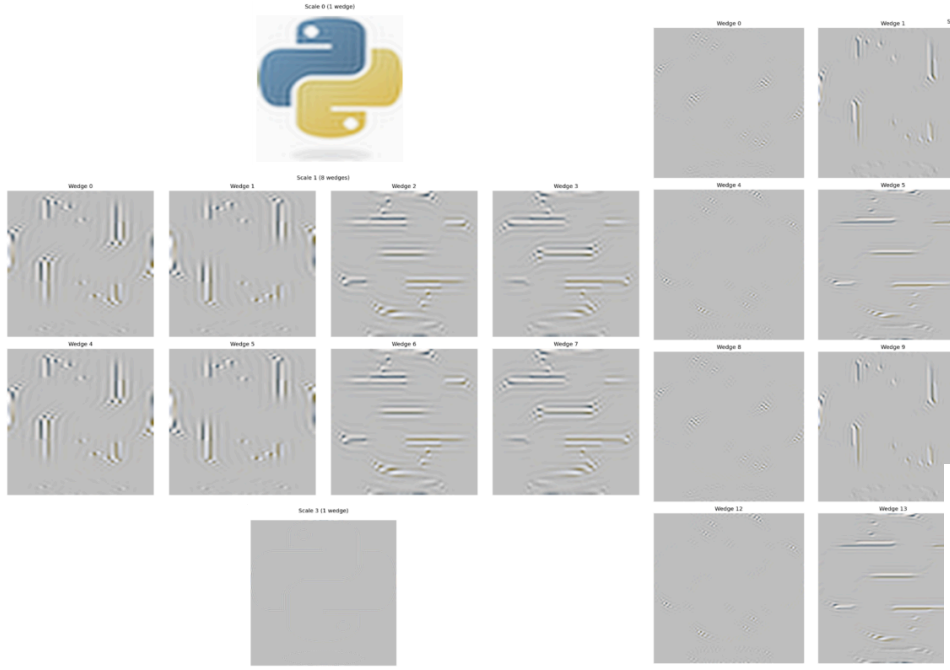


Figure 12. Curvelet coefficients (`nbscales=4, nbangles_coarse=8, allcurvelets=F`
Figure 11. Credit: own work.

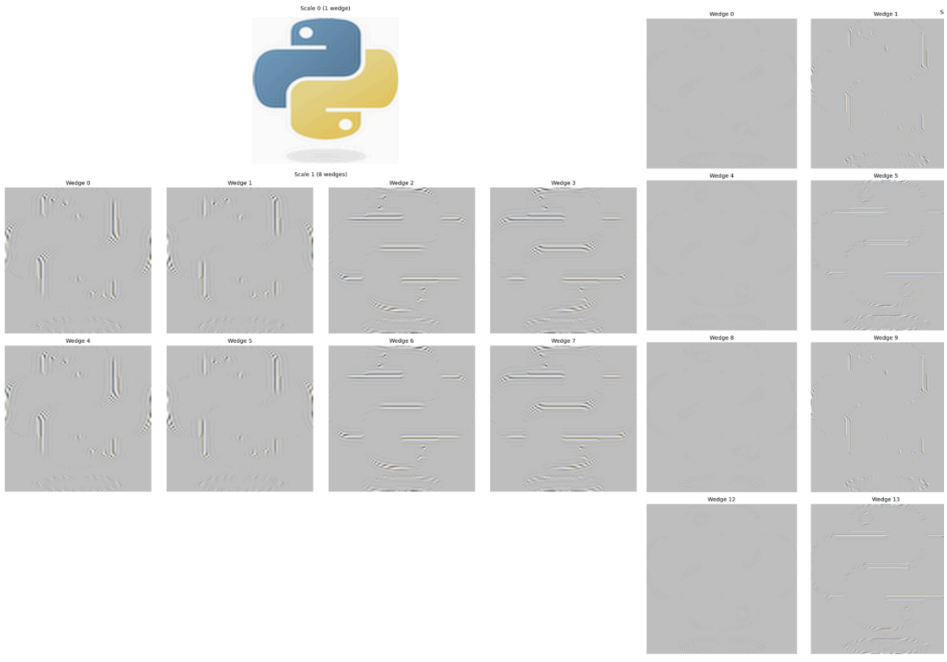


Figure 13. Curvelet coefficients (`nbscales=3, nbangles_coarse=8, allcurvelets=True`
11. Credit: own work.

It is interesting to notice that the "color" of the
the coarsest scale because this scale contains t
frequency aka the DC-component. All other scal
variations on top of that initial image.

Another interesting aspect of the curvelet transform is that it is very easy to interpret its coefficients. They are essentially a "piece" of the original image which varies along certain preferential directions.

But visualizing all of these coefficients can quickly become overwhelming. One way to overcome that is to extract features from a single wedge. In the following example we will subdivide each wedge into rows/cols, of which we will con These energies will be mapped to disks in the F will see below how this visualization can help us preferential directions of events in an image in e

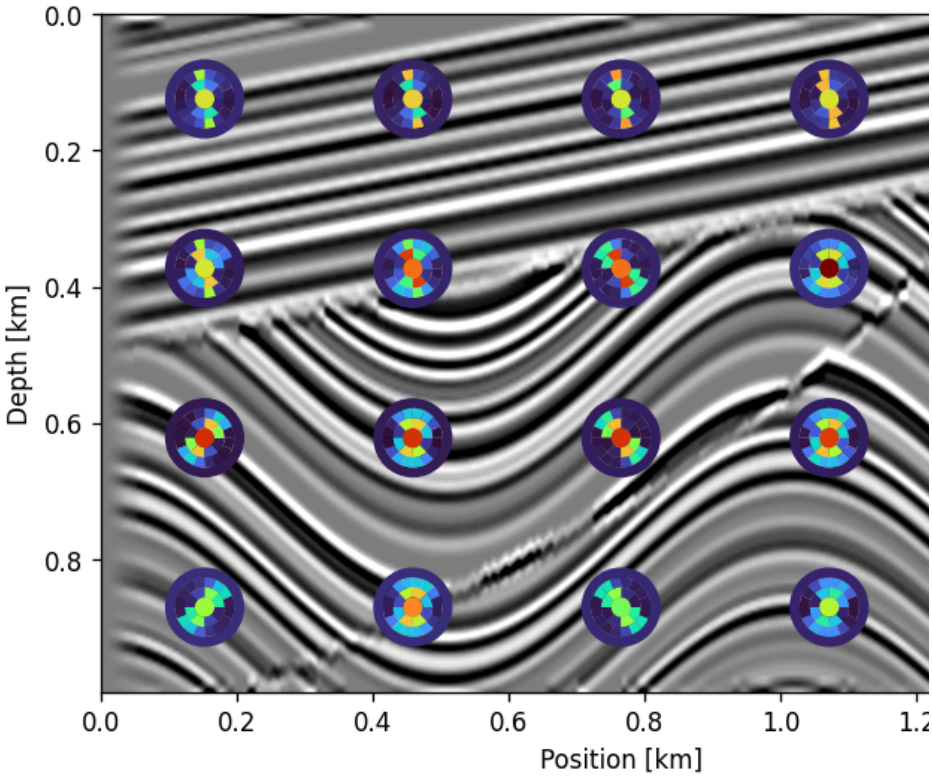


Figure 14. Sigmoid seismic model with curvelet strength disks overlain. Streng perpendicular to the local dip. Credit: own work.

Here we are plotting, overlain on the original inp seismic model) the strength of the curvelet coe certain window in each scale/wedge. These disk

wedge division we discussed above: the closest to the center, the lower the scale; the angular wedges map to the same place in the k-space domain, in a polar projection.

Consequently, similar to our visualization of the k-max vectors, we should see that the strongest energy lies p*erpendicular* to the preferential local dips. As a reminder, this happens because the strongest points in the k-space are those where the image varies the most in that direction. So, perpendicularly t the least. We can identify this behavior in the im near the top of the image, where the structure i

However, these disks give us even more informa preferential dip at any arbitrary location. They al idea of the anisotropy of the image. For example no preferential direction, or many preferential di image is more isotropic. And this information is locally in *space*, but also separated by *scales* (s frequencies).

Of course, even the disks are a aggregate of the that the curvelet transform contains, which atte and versatility!

These are just some examples of the curvelet tr upcoming articles, we will explore how they can tasks!

Key Takeaways

- The **Fourier transform** of images give us an idea of the **preferential directions of change**
- The **curvelet transform** goes a step beyond that, telling us how the image is **varies at each location, in which direction and with which spatial frequency**.
- The curvelet transform is traditionally performed with the FDCT (**Fast Discrete Curvelet Transform**) provided in **Python by curvelops and CurveLab**
- The FDCT can be used in many areas such a **processing and deep learning** (see TorchOpe

Further Reading

[1] Candès, E., Demanet, L., Donoho, D., & Ying, L Discrete Curvelet Transforms. *Multiscale Modeli* *5*(3), 861–899.

[2] Ma, J., & Plonka, G. (2010). The Curvelet Trans *Processing Magazine*, *27*(2), 118–133.

WRITTEN BY

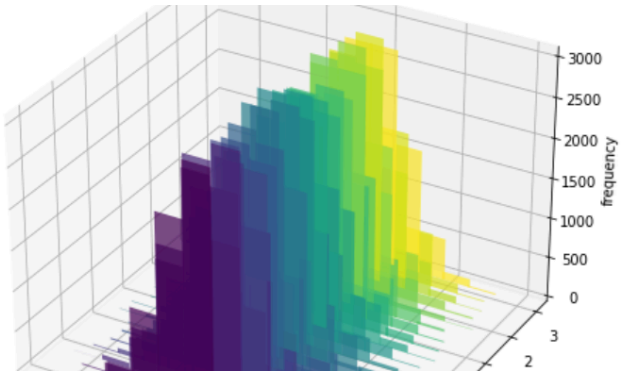
Carlos Costa, Ph.D.

See all from Carlos Costa, Ph.D.

Image Processing | Mathematics | Scientific Computing | Signal Processing | Wavelet

Share This Article

Related Articles



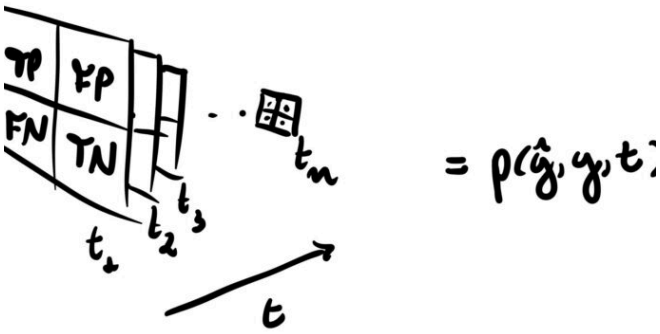
DATA SCIENCE

Must-Know in Statistics: The Bivariate Normal Projection Explained

Derivation and practical examples of this powerful concept

Luigi Battistoni

August 14, 2024 7 min read



MACHINE LEARNING

Binary Classif

Unpacking the Re
Limitations of Tra

gabriel costa

January 12, 2024

DATA SCIENCE

Let's Talk About Math (for Data Scientists)

Our weekly selection of must-read Editors' Picks and original features

TDS Editors

March 23, 2023 3 min read

ARTIFICIAL INTELLI

Teach (and le
transform ge

Understanding th
through a visualiz
the complex plar

Peter Barrett Bryan

March 18, 2022 3

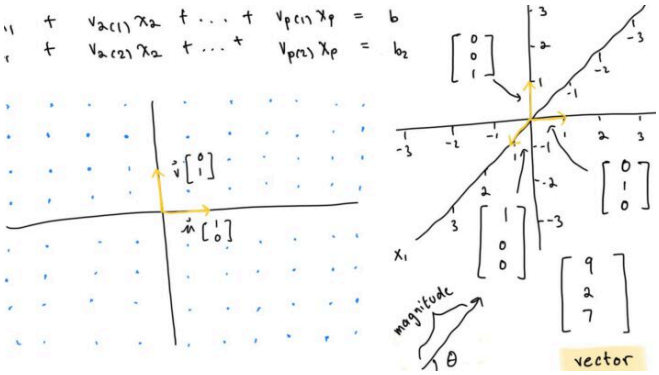
DATA SCIENCE

Multi-Dimensional Exploration Is Possible!

(At least mathematically)

Diego Manfre

October 5, 2023 16 min read



MACHINE LEARNING

Linear Algebra 2: Vector Equations

Vector Equations

Tenzin Migmar

October 14, 2023

MACHINE LEARNING

COUNTLESS 3D- Vectorized 2x Downsampling of Labeled Volume Images Using Python and Numpy

Previously, I demonstrated a fully vectorized algorithm, COUNTLESS, that downsampled labeled images by finding the...

William Silversmith

February 20, 2018 19 min read

Subscribe to Our Newsletter