Isotropic and Steerable Wavelets in N Dimensions. A multiresolution analysis framework.

Release 0.4

P. H. Cerdan [1,2]

June 1, 2017

[1] Institute of Fundamental Sciences, Massey University, New Zealand
[2] p.hernandezcerdan@massey.ac.nz

Abstract

This document describes the implementation of the external module ITKIsotropicWavelets, a multiresolution (MRA) analysis framework using isotropic and steerable wavelets in the frequency domain. This framework provides the backbone for state of the art filters for denoising, feature detection or phase analysis in N-dimensions. It focus on reusability, and highly decoupled modules for easy extension and implementation of new filters, and it contains a filter for multiresolution phase analysis,

The backbone of the multi-scale analysis is provided by an isotropic band-limited wavelet pyramid, and the detection of directional features is provided by coupling the pyramid with a generalized Riesz transform. The generalized Riesz transform of order N behaves like a smoothed version of the Nth order derivatives of the signal. Also, it is steerable: its components impulse responses can be rotated to any spatial orientation, reducing computation time when detecting directional features.

This paper is accompanied with the source code, input data, parameters and output data that the author used for validating the algorithm described in this paper. This adheres to the fundamental principle that scientific publications must facilitate reproducibility of the reported results.

Latest version available at the Insight Journal [http://hdl.handle.net/10380/3558]

Contents

1 Wavelet Multiresolution Analysis

1.1 Introduction

This module is an implementation for ITK of the work made by: [1, 2, 3, 4, 5, 6, 7].

Learning about wavelets from research papers is not easy. The topic is rich and deep, the same tool has spanned different research fields, from theoretical physics, to seismology and signal processing. Yves Meyer has won the Gauss Price (2010), and Abel Price (2017) in Mathematics for "his pivotal role in the development of wavelets and multi-resolution analysis". The detection of Gravitational waves with the LIGO experiment involved the use of wavelets to analyse the signals.

Stephan Mallat [8] had also a fundamental role to develop the Multiresolution Analysis (MRA) and make the implementation available for applications.

The steerable framework [1, 2] is widely used in applications to rotate the filter bank to any direction, avoiding expensive computations.

We will also implement more recent work [4, 6, 5] that uses the Riesz transform (a natural generalization of the Hilbert transform for N dimensions) to provide a generalization and extra flexibility to the steerable pyramid.

1.2 Motivation: spatial and frequency resolution

Wavelets share the same motivation than the short-time FFT, or the windowed Fourier transform: get a representation of the signal/image/function that is well localized in space **and** frequency domains. Heisenberg Uncertainty principle applies here and it is called the Heisenberg-Gabor limit: $\Delta t \cdot \Delta f \geq \frac{1}{4\pi}$, limiting the simultaneous resolution of a signal in time-frequency. The Fourier transform, which is the representation of the signal in the basis of harmonic functions $\{\sin(f), \cos(f)\}\ \forall f \in \mathbb{R}$, is completely localized in frequency $\Delta f = 0$, but has global support in space, i.e infinitely de-localized in space $\Delta t = \infty$.

A few examples to illustrate the concept: In a discrete case, such as an image, this means that the FFT has the highest resolution on frequency, but where, in the spatial domain, a specific frequency occurs is reduced to 'somewhere',

bounded by the size of the image (Figure 1, 2a). Or in signals occurring only in one time-lapse: they can't be differentiated from the same signal occurring continuously, so they will share similar spectra representation (Figure 1). In the same line, non-stationary signals which frequency depends on time will give a spectrum where all those frequencies are represented, but there is no information about when, in time, the change of frequency occurred.

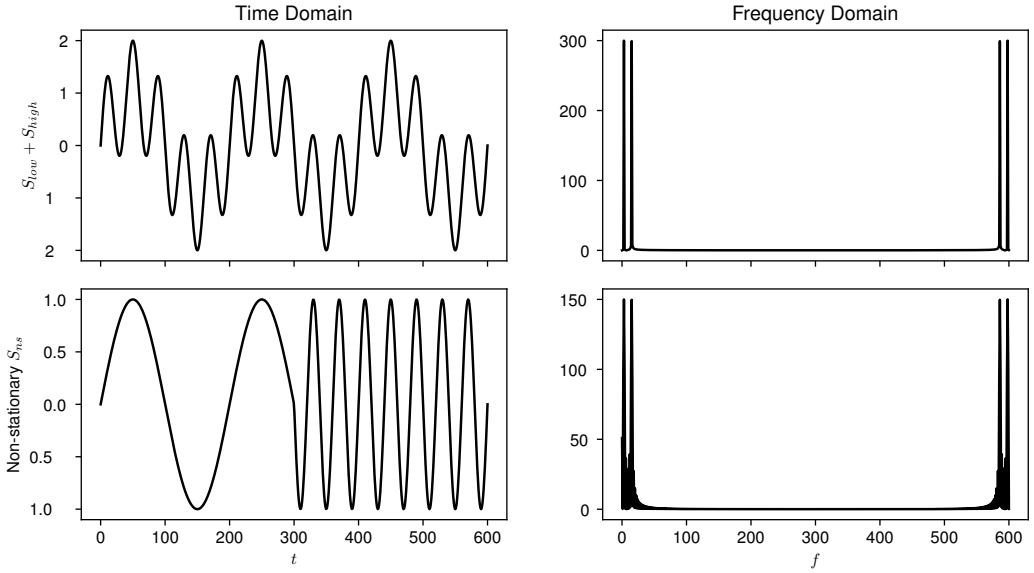
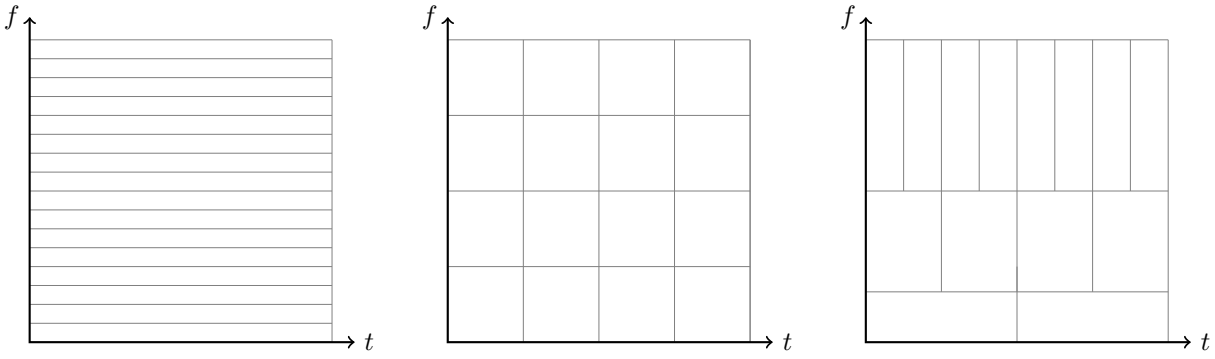


Figure 1: The changes over time/space of a signal are not captured by the FFT.

The short-time FFT add spatial/time localization dividing the original signal in small, consecutive segments, and applying the FFT to each of those. The problem with this approach, besides being computationally expensive, is that events shorter than the time window are still not resolved and that the width of the segment is constant (Figure 2b). Would it be possible to have better spatial resolution for some components of the frequency spectra, such as high frequency components (Figure 2c)?



(a) Fourier Transform, time resolution is the duration of the signal.

(b) Windowed-FFT or Short-Time-FFT, fixed width window

(c) Discrete Wavelet Transform. Adaptive window, better time resolution at high frequencies.

Figure 2: Simultaneous spatial and frequency resolution of different transformations. $\Delta t, \Delta f$

1.3 Wavelet transformation

Let consider the following wavelet decomposition $\forall f \in L_2(\mathbb{R}^d)$

$$f(\boldsymbol{x}) = \sum_{s\in\mathbb{Z}} \sum_{\boldsymbol{l}\in\mathbb{Z}^d} \langle f, \psi_{s,\boldsymbol{l}}\rangle \psi^*_{s,\boldsymbol{l}}(\boldsymbol{x}) \tag{1}$$

The family of functions $\{\psi_{s,\boldsymbol{l}}\}$ is a wavelet *frame* (Def. 1.2), constructed by means of *translations* and *dilations* (Def. 1.1) of the *mother wavelet* function ψ.

$$\psi_{s,\boldsymbol{l}}(\boldsymbol{x}) = |\det(A)|^{-s/2}\psi(A^s\boldsymbol{x} - \boldsymbol{l}) \tag{2}$$

Each dilation, defined by the dilation matrix A, squeezes or stretches the mother wavelet, acting as a change of scale. The translation operator moves and centers the location of the mother wavelet ψ. If the dilation matrix is diagonal with the same dilation factor a in all dimensions, $A = a\mathcal{I}_d$, then Equation 2 becomes:

$$\psi_{s,\boldsymbol{l}}(\boldsymbol{x}) = a^{-d\cdot s/2}\psi(a^s\boldsymbol{x} - \boldsymbol{l}) \tag{3}$$

The mother wavelet function has to have finite energy $\psi \in L_2(\mathbb{R}^d)$, i.e $\int_{-\infty}^{\infty} |\psi(\boldsymbol{x})|^2 d\boldsymbol{x} < \infty$.

Definition 1.1 (Dilations and Translations)**.** Given a function $f \in \mathbb{R}$ we define the dilation and translation operators[9]:

Dilation:	$D_a f(x) := \lvert a\rvert^{-1/2} f(x/a)$	for $a \in \mathbb{R}\backslash\{0\}$
Translation:	$T_b f(x) := f(x-b)$	for $b \in \mathbb{R}$

$\mathbb{R}^d$ generalization:
Given $f \in L_2(\mathbb{R}^d)$, $f : \mathbb{R}^d \to \mathbb{R}$, $\boldsymbol{x} \in \mathbb{R}^d$, the dilation scalar a is replaced by a dilation matrix $A = a\mathcal{I}_d$ [10]. The dilation matrix A is expansive, having all its eigenvalues $|\lambda_i| > 1$, so it is invertible. For the translation operator, the scalar b is replaced for the vector $\boldsymbol{b}$.

Dilation:	$D_A f(\boldsymbol{x}) := \lvert\det(A)\rvert^{-1/2} f(A\boldsymbol{x})$	A expansive matrix
Translation:	$T_{\boldsymbol{b}} f(\boldsymbol{x}) := f(\boldsymbol{x} - \boldsymbol{b})$	for $\boldsymbol{b} \in \mathbb{R}^d$

Definition 1.2 (Frame)**.** A family of functions $\{\phi_{\boldsymbol{k}}\}_{\boldsymbol{k}\in\mathbb{Z}^d}$ is a *frame* of $L_2(\mathbb{R}^d)$ if and only if there exists two positive constants A, B $< \infty$ such that:

$$A\|f\|^2 \leq \sum_{\boldsymbol{k}\in\mathbb{Z}^d} |\langle \phi_{\boldsymbol{k}}, f\rangle|^2 \leq B\|f\|^2, \forall f \in L_2(\mathbb{R}^d) \tag{4}$$

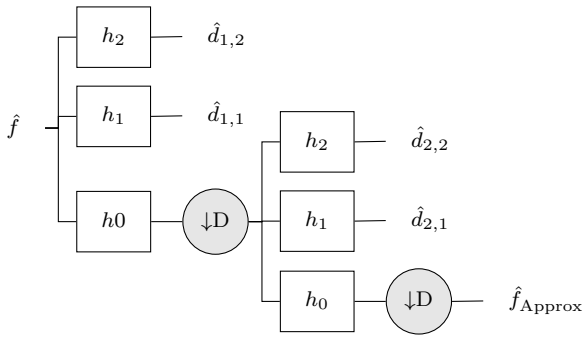
The frame is *tight* if $A = B$. If $A = B = 1$ we have a *Parseval frame* that satisfies the decomposition/reconstruction formula:

$$f = \sum_{\boldsymbol{k}\in\mathbb{Z}^d} \langle \phi_{\boldsymbol{k}}, f\rangle \cdot \phi_{\boldsymbol{k}}, \forall f \in L_2(\mathbb{R}^d) \tag{5}$$

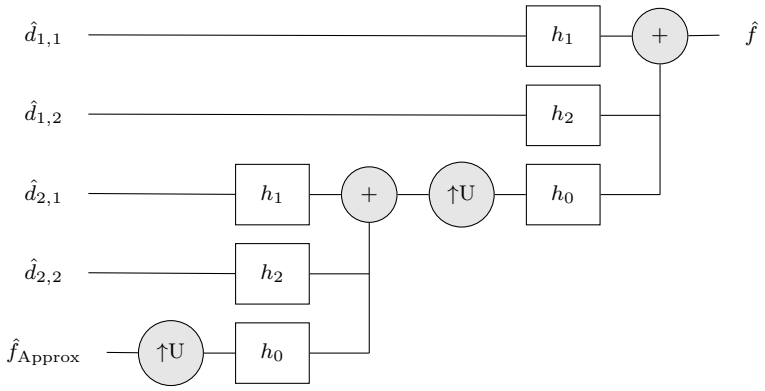
which has the same form than the expansion using an orthonormal basis, however in the frame generalization, the family $\phi_{\boldsymbol{k}}$ may be redundant.

1.4 Wavelet Pyramid

A band-limited pyramid is created by applying, at each level, a low-pass filter h_0 and downsampling by a scale factor of two, see Figure 3a. Because the wavelet frames are tight, we can get perfect reconstruction applying the inverse pyramid from Figure 3b. In Figure 3 we use two levels and two high pass subbands h_1, h_2. The results of this pyramid are the detail coefficients $d_{s,h}$, where $s \in \{1, \ldots, \text{Levels}\}$, and $h \in \{1, \ldots, \text{HighPassSubBands}\}$. Note that these details are in the frequency domain, if the original image was given in spatial domain, an inverse Fourier transform for each $d_{s,h}$ must be performed to get the wavelet coefficients in the spatial domain.



(a) forward/analysis.



(b) inverse/reconstruction.

Figure 3: Forward 3a and Inverse wavelet 3b two-level pyramid with two high pass sub-bands.

Usually in the literature use only one high-pass sub-band, the wavelet filter bank consists then in one high pass filter and one low pass filter (seeFigure 4). The adventages of the subbands is that they provide more frequency resolution [4], at expenses of more computation time, and also they might generate spatial domain distortions due to multiple sharp cutoffs. For phase detection (see the application in Figure 9), they are fundamental.

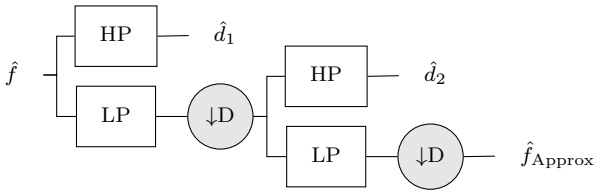


Figure 4: Forward wavelet pyramid with a classic two level filter bank and only one sub-band, HP is the high pass filter, and LP low pass

1.5 Isotropic wavelets

These wavelets are non-separable in Cartesian coordinates, depending on $\|\boldsymbol{\omega}\|$. Isotropic wavelets have the same mother wavelet for each scale. All of the implemented wavelets fulfill the conditions of Prop 1, their shape differ in

the spatial decay, and vanishing moments. Which one is better depend on the application, and developing new wavelets is a research topic. We have implemented four, there are at least [6] two isotropic wavelets missing: Papadakis [11], and Meyer [12]. The advantage of isotropic wavelets is that they are steerable, but in order to get directionality features, the wavelets are coupled with other filters that gather directional information, such as the Riesz transform, see section 2.

The present requirement that the mother wavelet is isotropic constraints the wavelet-design. The goal is to generate a tight wavelet frame and have perfect reconstruction with the inverse pyramid.

Proposition 1. Conditions for the mother wavelet ψ to generate a tight wavelet frame [4, 6, 5]:
Let $h(w)$ be a radial frequency profile such that:

1. (Band-limitedness): $h(\omega) = 0, \forall \omega > \pi$.
2. (Riesz Partition of Unity): $\sum_{i\in\mathbb{Z}} \left|h\left(\left(\left(A^{\mathsf{T}}\right)^{-1}\right)^{i}\omega\right)\right|^2 = 1.$
   If $A = a\mathcal{I}_d$: $\sum_{i\in\mathbb{Z}} \left|h(a^i\omega)\right|^2 = 1$. [4, 13]
3. (Vanishing Moments): $\frac{\mathrm{d}^n h(\omega)}{\mathrm{d}\omega^n}\Big|_{\omega=0} = 0$, for $n = 0, \ldots, N$.

If the mother wavelet ψ is defined by $\hat{\psi}(\boldsymbol{\omega}) = h(\|\boldsymbol{\omega}\|)$, then it generates a tight wavelet frame in $L_2(\mathbb{R}^d)$, where $\hat{\psi}$ is the ND Fourier transform of ψ.

Condition 2 guarantees that the tiling from all the scales fills the frequency domain, see Figure 5.

VOW, variance-optimal wavelets [7] :

$$h(\omega) = \begin{cases} \sqrt{\frac{1}{2} + \frac{\tan\left(\kappa(1+2\log_2\frac{2\omega}{\pi})\right)}{2\tan(\kappa)}}, & \omega \in [\frac{\pi}{4}, \frac{\pi}{2}[\\ \sqrt{\frac{1}{2} - \frac{\tan\left(\kappa(1+2\log_2\frac{\omega}{\pi})\right)}{2\tan(\kappa)}}, & \omega \in [\frac{\pi}{2}, \pi] \\ 0, & \text{otherwise} \end{cases}$$

where $\kappa \in [0, \frac{\pi}{2}]$ is found to be 0.75

Held [4] :

$$h(\omega) = \begin{cases} \cos\left(2\pi q_n(\frac{\omega}{2\pi})\right), & \omega \in]\frac{\pi}{4}, \frac{\pi}{2}] \\ \sin\left(2\pi q_n(\frac{\omega}{4\pi})\right), & \omega \in]\frac{\pi}{2}, \pi] \\ 0, & \text{otherwise} \end{cases}$$

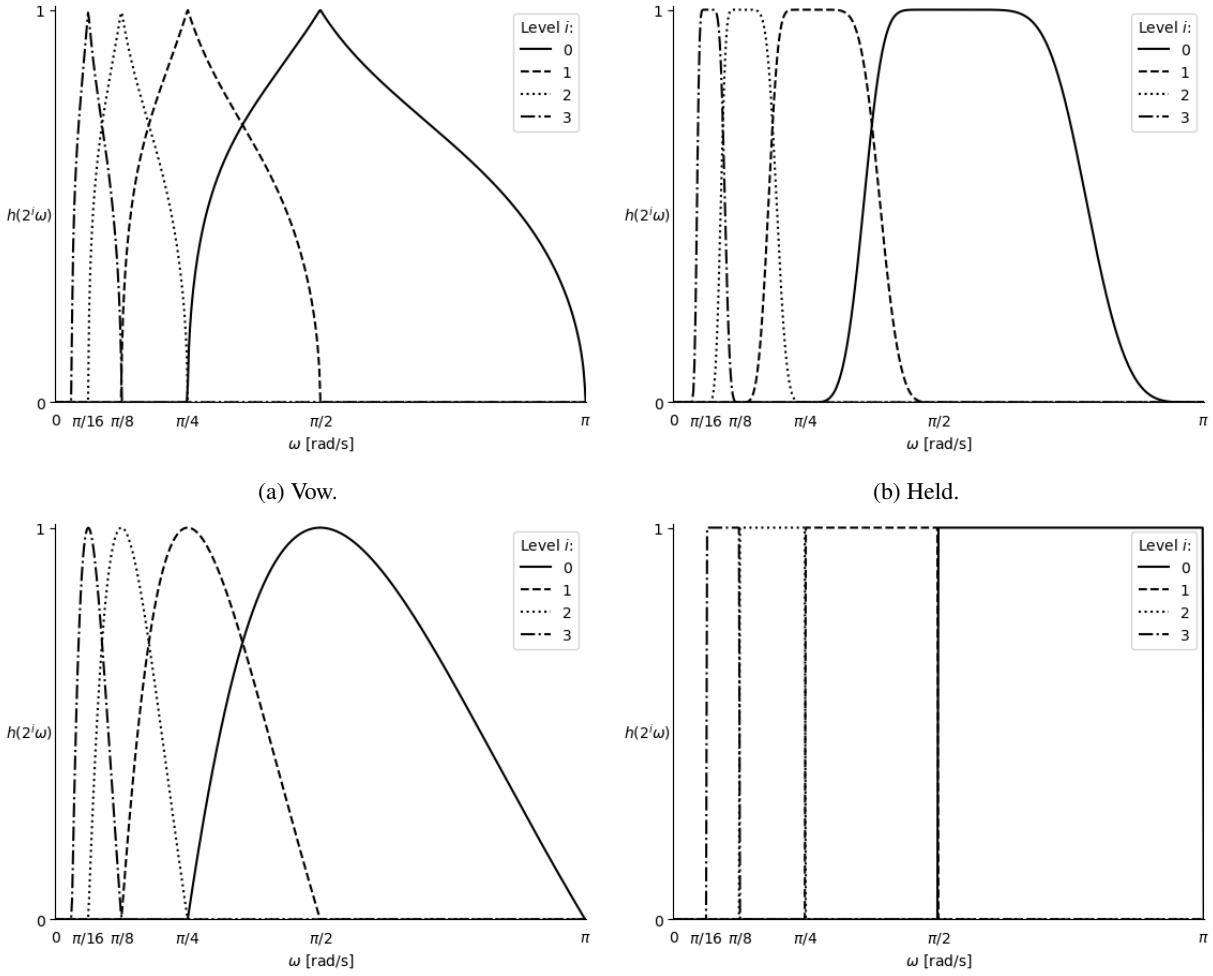
where q_n is a polynomial function of order n

Simoncelli [14, 2] :

$$h(\omega) = \begin{cases} \cos\left(\frac{\pi}{2}\log_2\frac{2\omega}{\pi}\right), & \omega \in]\frac{\pi}{4}, \frac{\pi}{2}] \\ 0, & \text{otherwise} \end{cases}$$

Shannon:

$$h(\omega) = \begin{cases} 1, & \omega \in [\frac{\pi}{2}, \pi] \\ 0, & \text{otherwise} \end{cases}$$



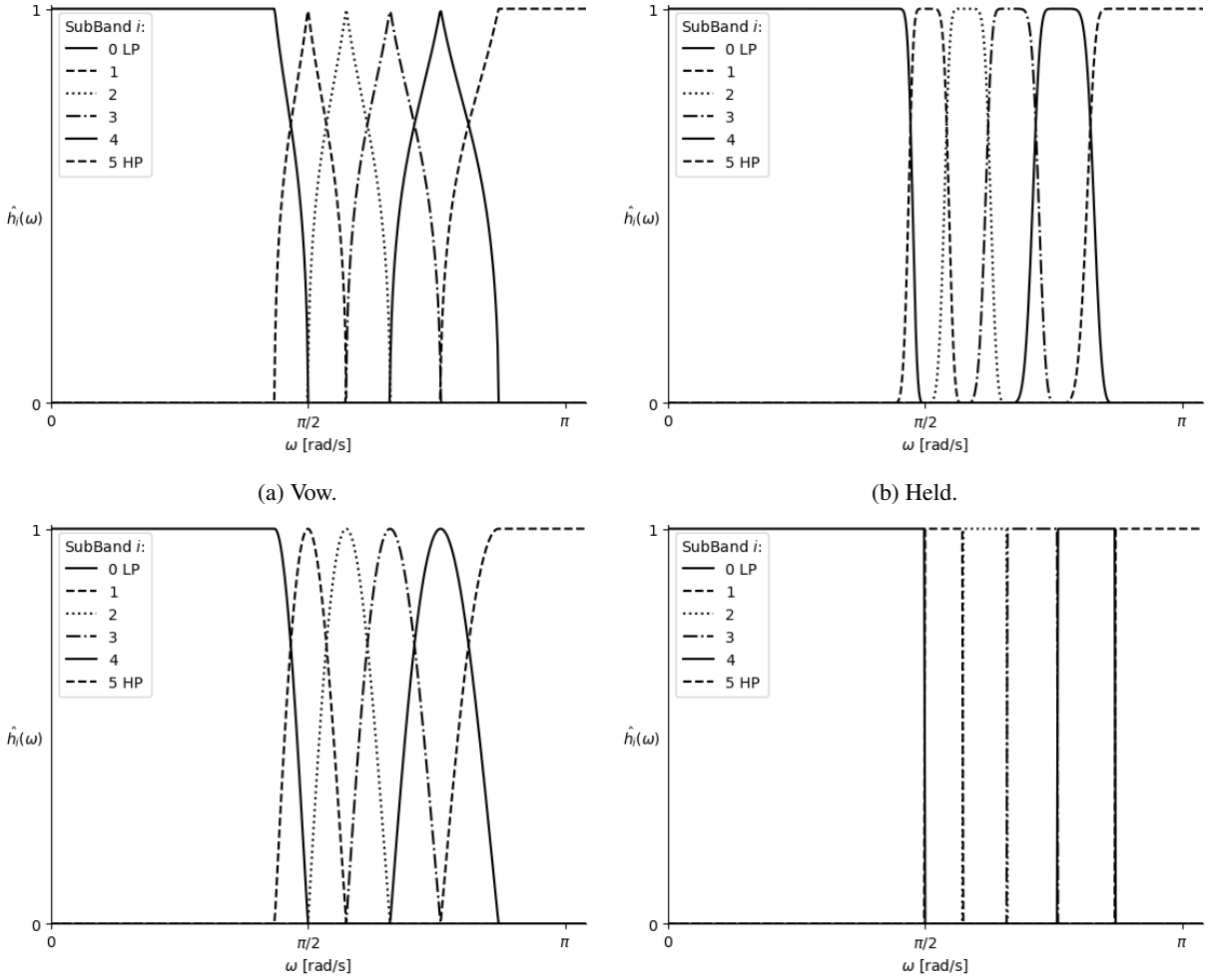
(a) Vow.

(b) Held.

(c) Simoncelli.

(d) Shannon.

Figure 5: Tiling of the frequency domain by isotropic wavelets when the dilation factor is 2. All the wavelets fulfill the conditions from Proposition 1. The mother wavelets are represented at $i = 0$.



(a) Vow.

(b) Held.

(c) Simoncelli.

(d) Shannon.

Figure 6: Shape of SubBands when HighPassSubBands = 5, sub-bands increase the frequency resolution, but can generate extra artifacts in the spatial domain due to the sharp frequency cut-offs.

2 Riesz Transform

The forward or analysis wavelet pyramid outputs a set of wavelet coefficients with information about each scale. A steerable filter [4, 6, 2] can be applied to the output of the wavelet pyramid. This is the main purpose of the isotropy of the mother wavelet, the steerable filter is used to select the orientation where the feature of interest is maximum.

There are mathematical constraints in the steerable filters that can be coupled with the wavelet pyramid. Read more: [15, 6].

One of the transform that can be coupled to the pyramid is a Riesz transform $\boldsymbol{\mathcal{R}}$ of order $N = 1$

$$\boldsymbol{\mathcal{R}} f(\boldsymbol{x}) = \begin{pmatrix} \mathcal{R}_1 f(\boldsymbol{x}) \\ \vdots \\ \mathcal{R}_d f(\boldsymbol{x}) \end{pmatrix} \quad \xleftrightarrow[\mathcal{F}^{-1}]{\mathcal{F}} \quad \hat{\boldsymbol{\mathcal{R}}} \hat{f}(\boldsymbol{\omega}) = -j \frac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|} \hat{f}(\boldsymbol{\omega}) \tag{6}$$

where d is the image dimension and the number of components of the Riesz transform of order $N = 1$. $j = \sqrt{-1}$ and

$\mathcal{F}, \mathcal{F}^{-1}$ are the forward and inverse Fourier Transform.

The Riesz transform is the generalization of the Hilbert transform for ND. The Hilbert transform $\mathcal{H}$ is used to generate the *analytic signal* in 1D.

$$\begin{gathered} f_a(x) = f(x) + j\mathcal{H}f(x) \\ \mathcal{H}f(x) = (h * f)(x) \qquad \xleftrightarrow[\mathcal{F}^{-1}]{\mathcal{F}} \qquad \hat{\mathcal{H}}\hat{f}(\omega) = -jsgn(\omega)\hat{f}(\omega) = -j\frac{\omega}{|\omega|}\hat{f}(\omega) \end{gathered} \tag{7}$$

2.1 Monogenic Signal

There is no direct generalization of the analytical signal in ND, but we can generate a signal with similar properties, called the **Monogenic signal** using the Riesz transform of order 1[16, 3]. The Monogenic signal is a set formed by the original signal and the D-components of the first order Riesz transform.

$$f_a = \{f, \mathcal{R}_x, ..., \mathcal{R}_d\} \tag{8}$$

The amplitude A and the phase P at each location of the Monogenic signal are:

$$A(\mathbf{x_0}) = \sqrt{f(\mathbf{x_0})^2 + A_R(\mathbf{x_0})^2} \tag{9}$$

where $A_R(\mathbf{x_0}) = \sqrt{\sum_{i=1}^{N} \mathcal{R}_i(\mathbf{x_0})^2}$

$$P(\mathbf{x_0}) = \text{atan2}(A_R(\mathbf{x_0}), A(\mathbf{x_0})) \tag{10}$$

The Monogenic signal can be used to perform local phase analysis for feature detection in ND.

2.2 Generalized Riesz Transform

The Riesz transform will map any frame of $L_2(\mathbb{R}^d)$ into another one [4, 17]. This mapping property allows to couple the Riesz transform of any order with the wavelet pyramid and if the wavelet function is isotropic, to get perfect reconstruction when performing the inverse transform. The Riesz transform of order N generate a vector containing smoothed directional Nth derivatives.

We will summarize the multiindex notation introduced in [6].
Consider $\boldsymbol{n} = (n_1, \ldots, n_d)$ a d-dimensional multiindex vector, where the n_i entries are non-negative integers. And then define the following operators and operations:

1. Sum of components: $|\boldsymbol{n}| = \sum_{i=1}^{d} n_i = N$.
2. Factorial: $\boldsymbol{n}! = n_1!n_2!\cdots n_d!$
3. Exponentiation of a vector $z = (z_i, \cdots, z_d) \in \mathbb{C}^d$: $z^{\boldsymbol{n}} = z_1^{n_1} \cdots z_d^{n_d}$
4. Partial derivative of a function $f(\boldsymbol{x}), \boldsymbol{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$: $\partial^{\boldsymbol{n}} f(\boldsymbol{x}) = \frac{\partial^N f(\boldsymbol{x})}{\partial x_1^{n_1} \cdots \partial x_d^{n_d}}$

Riesz transforms can be connected to the partial derivatives [6]:

$$\mathcal{R}^{\boldsymbol{n}}(-\Delta)^{\frac{|\boldsymbol{n}|}{2}} f(\boldsymbol{x}) = (-1)^{|\boldsymbol{n}|}\sqrt{\frac{|\boldsymbol{n}|!}{\boldsymbol{n}!}}\partial^{\boldsymbol{n}} f(\boldsymbol{x})$$

where $(-\Delta)^{\gamma}$ is the fractional Laplace of order γ. From [6]: "Since the inverse of $(-\Delta)^{\frac{|\boldsymbol{n}|}{2}}$ is an isotropic low-pass-filtering operator, the net effect of the higher order Riesz transform is to extract smoothed version of the derivatives of order N of the signal of interested."

The number of components of $\boldsymbol{\mathcal{R}}$ depends on the order of the Riesz transform N, and the dimension of the signal d:

$$M = p(N, d) = \frac{(N + d - 1)!}{(d - 1)!N!} \tag{11}$$

2.3 Generalized Steerable Framework

Simoncelli [2] coupled the multi-resolution analysis using wavelets with the steerable concept developed years before [1]. A steerable framework can be used with polar separable functions. In this case, the radial part of the function has to be calculated only once, and the polar part has only to be computed in those directions that constitute a basis of the space. After this, the filter can be oriented, or steered to any direction, using a weighted linear combination of the basis. This approach gives a directional analysis but keeping it computationally performant.

The steerable framework can be combined with wavelets when these are isotropic, which implies polar separability. Simoncelli's framework has been used in a lot of 2D applications. Freeman in its seminal work [1] wrote a generalized version for any dimension, but it was lately, Chenouard and Unser [4, 6, 5] who used rotation matrices in 3D.

Definition 2.1. A function f is steerable in three dimensions if it can be expressed as:

$$f(\boldsymbol{R}\boldsymbol{x}) = \sum_{m=1}^{M} k_m(\boldsymbol{R}) g_m(\boldsymbol{x}) \tag{12}$$

where $\boldsymbol{x} = (x_1, x_2, x_3)$ is any 3D vector in Cartesian coordinates and $\boldsymbol{R}$ is any rotation matrix in three dimensions. $\{g_m\}_{m=1\ldots M}$ is the set of primary functions, ie , a basis, and $\{k_m\}_{m=1\ldots M}$ is a set of interpolation functions with $M < \infty$.

3 Implementation Details

3.1 Summary

Most of the filters in this module require input images in the dual space (frequency domain). If working with regular spatial-domain images, a `itk::ForwardFFTImageFilter` has to be applied. The decision is based on performance and accuracy, avoiding expensive convolution operations and also multiple Fourier transforms.

Also, because we work in the frequency domain, we add a itk::FrequencyShrinkImageFilter and a itk::FrequencyExpandImageFilter **without** any interpolation. The shrinker chops the high frequency pixels of the image. And the expander adds zeros in the higher frequency bins. These filters require input images to be hermitian –note that the output of a forward FFT on a real image is hermitian.
General FrequencyExpanders and FrequencyShrinkers that can be applied to non-hermitian complex images are not implemented, although some preliminary work can be found in `https://github.com/phcerdan/ITKIsotropicWavelets/pull/31` based on [18].

3.2 Frequency Iterators

Every filter that uses the frequency value of a pixel can be templated with a itk::FrequencyImageRegionIterator.
This kind of iterators add the functions GetFrequencyBin(), GetFrequencyIndex() to a regular `itk::ImageRegionIterator`, helping to abstract the complexity of the frequency layout into the iterator. The layout determines the order and location of the frequency bins: zero frequency –DC component–, Nyquist, low/high, or positive and negative frequencies. The layout changes depending on the parity of the image, the forward Fourier transform algorithm chosen, or if the frequencies have been shifted (`itk::FFTShiftImageFilter`).
Also these frequency iterators have data members: FrequencyOrigin, FrequencySpacing, holding metadata information about the frequency domain.
ITK puts a strong emphasis in the ImageInformation or metadata on spatial domain images: Origin, Spacing, Direction, Index. There are strong requirements when dealing with images with different metadata, for example, multiplication between two images is only possible when they have compatible ImageInformation.
Although, when applying a `itk::ForwardFFTImageFilter`, the output image, which is now in the frequency domain, still have metadata referring to the spatial domain, but nothing about the frequency origin, or spacing between frequencies.

There are three classes of FrequencyImageRegionIterator:

• FFTLayout: itk::FrequencyFFTLayoutImageRegionIterator
This iterator assumes that the frequency image has a "standard layout". Different libraries, such as VNL, FFTW, numpy.fft, generate this layout. Each image dimension is divided in two regions, holding positive and negative frequencies.

- ZeroFrequency: The index holding the zero frequency value is at the origin $[0, \ldots, 0]$ (upper-left, corner).

- Nyquist: When size N is even, the positive Nyquist frequency is located in the middle, index $N/2$, and the negative Nyquist is not stored. When N is odd, there is no Nyquist frequency, but most positive frequency is at index $(N-1)/2$, and most negative frequency is at index $(N+1)/2$

The first index after the origin corresponds to the positive lowest frequency, say 0.1Hz. If this is the index 1, the latest index $N-1$ corresponds to the less negative frequency -0.1Hz

• ShiftedFFTLayout: itk::FrequencyShiftedFFTLayoutImageRegionIterator
The standard layout can be confusing to reason about, a common alternative is to shift the zero frequency bin to the center of the image via `itk::FFTShiftImageFilter`.

• Regular: itk::FrequencyImageRegionIterator.
This iterator is for images that were taken experimentally in the frequency domain. GetFrequency() here is just a wrap of TransformIndexToPhysicalPoint, and GetFrequencyBin is equal to GetIndex. It assumes that the image metadata refers to the frequency domain, so FrequencyOrigin and FrequencySpacing are equal to the regular Origin and Spacing. Iterators with GetFrequency() functions are needed for other classes in the module.

This abstraction will hopefully facilitate the implementation of further filters manipulating images in the frequency domain.

3.3 Wavelet Transform

See subsection 1.4. The Wavelet Transform is templated over one of the IsotropicWavelet functions from subsection 1.5 chosen by the user through a itk::WaveletFrequencyFilterBankGenerator. The generator uses the function to generate an image or a filter bank.

We can compute the maximum levels that an input accept with the ComputaMaxNumberOfLevels, right now the implementation accepts needs inputs of the form s^M, where s is the scale factor chosen for peforming the multiresolution, and M is an integer. Even though this condition is restrictive, we can resize any input using `itk::FFTPadImageFilter`, or if the user pipeline involves neighbor iterators, prefer itk::FFTPadPositiveIndexImageFilter, implemented in this module, that avoids setting negative indices.

The itk::WaveletFrequencyForward generates a set of coefficients generated after applying the high pass filters h_{band} for each level, and an approximation (the result of applying the cascade of low pass filter h_0).

The wavelet coefficients from the forward pyramid can be manipulated to perform further image analysis, for example edge detection, denoise, phase analysis for feature detection, etc. These extra analysis should be independent of the multiresolution framework.

After any manipulation, we can perform an inverse pyramid to reconstruct the image. Doing this require to plug in the modified wavelet coefficients into itk::WaveletFrequencyInverse, if the wavelet coefficients are not modified by any further analysis, we get exactly the same image as the origin image that we input to the forward pyramid.

3.4 Riesz Transform

We have a itk::RieszFrequencyFunction, implementing a Generalized Riesz Function [6] of any order $N > 0$. This function receives an input $\in \mathcal{R}^d$ and outputs a vector of M Riesz Components, see Equation 11. To generate images from this function, we use the itk::RieszFrequencyFilterBankGenerator. To rotate or steer the result of the Riesz transform,use itk::RieszRotationMatrix. Please be aware the multiindex notation in these classes, see subsection 2.2

3.5 Structure Tensor

Given an array of inputs, itk::StructureTensor [6] computes the linear combination (or direction) of inputs that maximizes the response for each location in the image. Instead of only measuring the response at the pixel of interest, it takes into account a local neighborhood. [6].

$$\mathbf{u}(\mathbf{x}_0) = \arg\max_{\|\mathbf{u}\|=1} \int_{\mathbb{R}^d} g(\mathbf{x}-\mathbf{x}_0)\,|\mathbf{I}_\mathbf{u}(\mathbf{x})|^2$$

$$|\mathbf{I}_\mathbf{u}(\mathbf{x})|^2 = \mathbf{u}^T \cdot \mathbf{I}(\mathbf{x}) \cdot (\mathbf{I}(\mathbf{x}))^T \cdot \mathbf{u}$$

I is the required std::vector of input images. These images might be the output of a directional filter to an image (for example, directional derivatives from an image) or the basis of a steerable filter, such as a RieszImageFilter. Instead of just selecting the max response from the vector at every pixel, it uses the response over a local neighborhood, specified using an isotropic Gaussian window $g(\mathbf{x})$. This approach is more robust against noise. The user can control the radius and sigma of this Gaussian kernel. Estimation of the local orientation this way results in an eigen-system with matrix:

$$[\mathbf{J}(\mathbf{x}_0)]_{mn} = \sum_{\mathbf{x}\in\mathbb{Z}^d} g(\mathbf{x}-\mathbf{x}_0) I_m[\mathbf{x}] I_n[\mathbf{x}]$$

where I_m, I_n are input images, $m, n \in 0, N-1$ and N is the total number of inputs. g is a Gaussian kernel.

The solution of the EigenSystem defined by **J** are the N EigenValues and EigenVectors. The output of StructureTensor is a 2D Matrix of size (N,N+1), where the submatrix (N,N) are the EigenVectors, and the last column (N+1) are the EigenValues. The orientation that maximizes the response: $\boldsymbol{u}$ is the EigenVector with largest EigenValue, which is is the Nth column of the output matrix. We can use the calculated direction $\boldsymbol{u}$ to get a new image with max response from the inputs at each pixel with the function ComputeProjectionImageWithLargestResponse(), or any other direction from other eigen vectors with ComputeProjectionImage

Also we can compare eigen values to study the local coherency of each pixel:

$$\chi(\mathbf{x}_0) = \frac{\lambda_N(\mathbf{x}_0) - A(\mathbf{x}_0)}{\lambda_N(\mathbf{x}_0) + A(\mathbf{x}_0)}$$

where $\lambda_N(\mathbf{x}_0)$ is the largest eigen value at pixel $\mathbf{x}_0$, and $A(\mathbf{x}_0) = \frac{1}{N-1}\sum_{i=1}^{N-1}\lambda_i(\mathbf{x}_0)$ is the average of the other eigen values.

3.6 Phase Analysis

This module also implements a base class itk::PhaseAnalysisImageFilter, and a specialization itk::PhaseAnalysisSoftThresholdImageFilter that applies a soft-threshold technique to ignore low amplitude values. The example application 10, uses a Monogenic signal to study the local phase of each wavelet coefficient to perform multi-scale feature detection [16, 15].

4 A guided example:

I recommend the reader interested in extend this module to have a look to the tests for more usage options.

4.1 Input in the frequency domain.

The input for the itk::WaveletFrequencyForward has to be a complex image of a float/double pixel type. If the image is not in the frequency domain already, apply a `itk::ForwardFFTImageFilter` to get it.

Some FFT algorithms only work for specific image sizes, for example, the size has to be a power of two. You can apply a `itk::FFTPadImageFilter` to pad the image with zeros to reach the required size. However, this filter sets some padded areas with negative indices, which can generate problems with neighbor iterators. As a work around, this module introduces itk::FFTPadPositiveIndexImageFilter to avoid negative indices.

4.2 Choosing an isotropic wavelet

Choose a mother wavelet from the options available 1.5, and create a itk::WaveletFrequencyFilterBankGenerator with the type of the mother wavelet as a template parameter.

```
typedef itk::HeldIsotropicWavelet< PixelType, ImageDimension>
  WaveletFunctionType;
typedef itk::WaveletFrequencyFilterBankGenerator< ComplexImageType,
 ↪  WaveletFunctionType >
  WaveletFilterBankType;
```

4.3 Forward / Analysis

Perform the wavelet transform based on input levels and high-frequency sub bands. `ComputeMaxNumberOfLevels` is a static class function to calculate the max level. Currently, the size of input image has to be a multiple of two, but the implementation has the member `m_ScaleFactor` for future extension and relaxation of this constraint. Use FFTPadPositiveIndexImageFilter for padding the image with zeros for a valid size.

```
typedef itk::WaveletFrequencyForward< ComplexImageType, ComplexImageType,
 ↪  WaveletFilterBankType >
  ForwardWaveletType;
typename ForwardWaveletType::Pointer forwardWavelet = ForwardWaveletType::New();
forwardWavelet->SetHighPassSubBands( highSubBands );
forwardWavelet->SetLevels(levels);
forwardWavelet->SetInput(fftFilter->GetOutput());
```

4.4 Inverse / Reconstruction

Before applying the inverse wavelet and get a reconstructed image, you want to modify the wavelet coefficients first. The design of the framework has tried to decouple both pyramids, so we can focus in algorithms that deal only the wavelet coefficients, on not with the details of the multiresolution framework.

Right now, only a phase analysis filter has been developed, but there are plenty of room for more, see section 5. Set the same options used in the forward pyramid. And set the modified wavelet coefficients. If you don't modify them you will reconstruct the original image in the frequency domain.

Perform the wavelet forward with a suitable image:

```
#include "itkHeldIsotropicWavelet.h"
#include "itkVowIsotropicWavelet.h"
#include "itkSimoncelliIsotropicWavelet.h"
#include "itkShannonIsotropicWavelet.h"
#include "itkWaveletFrequencyFilterBankGenerator.h"
#include "itkWaveletFrequencyForward.h"
...

  const unsigned int ImageDimension = 3;
```

```
typedef double                                          PixelType;
typedef std::complex< PixelType >                       ComplexPixelType;
typedef itk::Image< ComplexPixelType, ImageDimension >  ComplexImageType;

// Set the WaveletFunctionType and the WaveletFilterBank
typedef itk::HeldIsotropicWavelet< PixelType, ImageDimension >
      WaveletFunctionType;
typedef itk::WaveletFrequencyFilterBankGenerator< ComplexImageType,
 ↪  WaveletFunctionType >
  WaveletFilterBankType;

// WaveletFrequencyForward
typedef itk::WaveletFrequencyForward< ComplexImageType, ComplexImageType,
 ↪  WaveletFilterBankType >
  ForwardWaveletType;
typename ForwardWaveletType::Pointer forwardWavelet = ForwardWaveletType::New();
forwardWavelet->SetHighPassSubBands( highSubBands );
forwardWavelet->SetLevels(levels);
forwardWavelet->SetInput(fftFilter->GetOutput());
forwardWavelet->Update();

// Result of the wavelet decomposition.
typename ForwardWaveletType::OutputsType analysisWaveletCoeffs =
  forwardWavelet->GetOutputs();
```

With the wavelet coefficients in hand (in the frequency domain), perform a decoupled filter, phase analysis, denoising, feature detection, etc:

```
// Manipulate coefficients for your purposes:
// Denoise, phase analysis, etc...
typename ForwardWaveletType::OutputsType modifiedWaveletCoeffs;
for( unsigned int i = 0; i < forwardWavelet->GetNumberOfOutputs(); ++i )
  {
  ...
  aWaveletCoefficientFilter->SetInput( analysisWaveletCoeffs[i] );
  aWaveletCoefficientFilter->Update();
  modifiedWaveletCoeffs.push_back( aWaveletCoefficientFilter->GetOutput() );
  }
```

And then plug back the modified coefficients to the inverse pyramid to get a reconstruction.

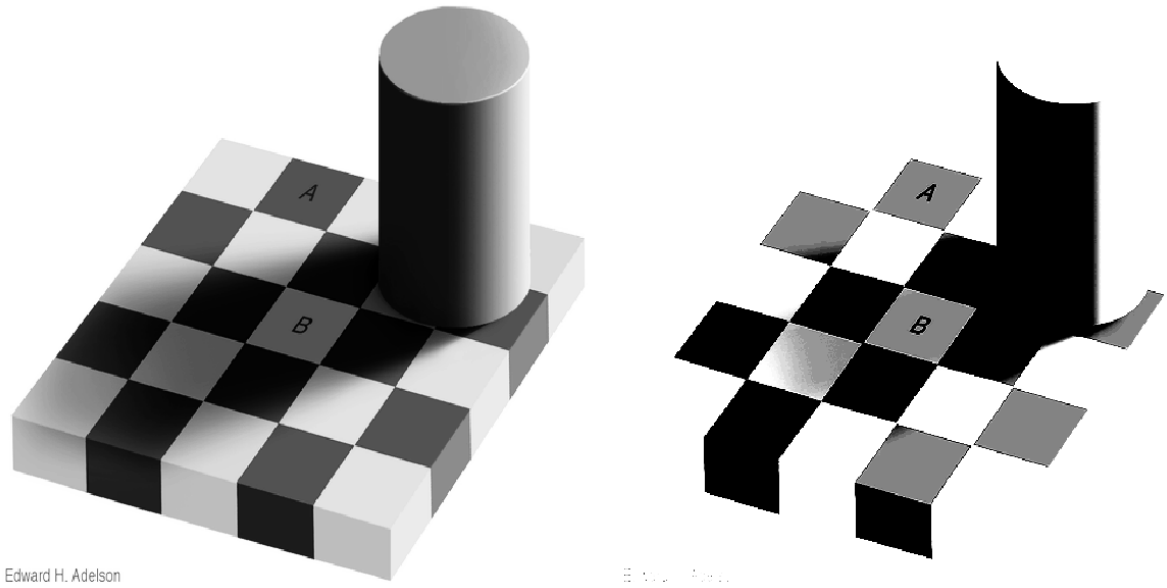
```
// Inverse Wavelet Transform
typedef itk::WaveletFrequencyInverse< ComplexImageType, ComplexImageType,
 ↪  WaveletFilterBankType >
  InverseWaveletType;
typename InverseWaveletType::Pointer inverseWavelet = InverseWaveletType::New();
inverseWavelet->SetHighPassSubBands( highSubBands );
inverseWavelet->SetLevels( levels );
// inverseWavelet->SetInputs( forwardWavelet->GetOutputs() );
inverseWavelet->SetInputs( modifiedWaveletCoeffs );
bool useWaveletFilterBankPyramid = true;
inverseWavelet->SetUseWaveletFilterBankPyramid( useWaveletFilterBankPyramid );
inverseWavelet->SetWaveletFilterBankPyramid(
 ↪  forwardWavelet->GetWaveletFilterBankPyramid() );
inverseWavelet->Update();
```

```
// The output of the inverse wavelet is in the frequency domain.
// Perform an InverseFFT to get a real image.
typename InverseFFTFilterType::Pointer inverseFFT = InverseFFTFilterType::New();
inverseFFT->SetInput( inverseWavelet->GetOutput() );

// Write or visualize the reconstructed output.
writer->SetInput(inverseFFT->GetOutput());
```

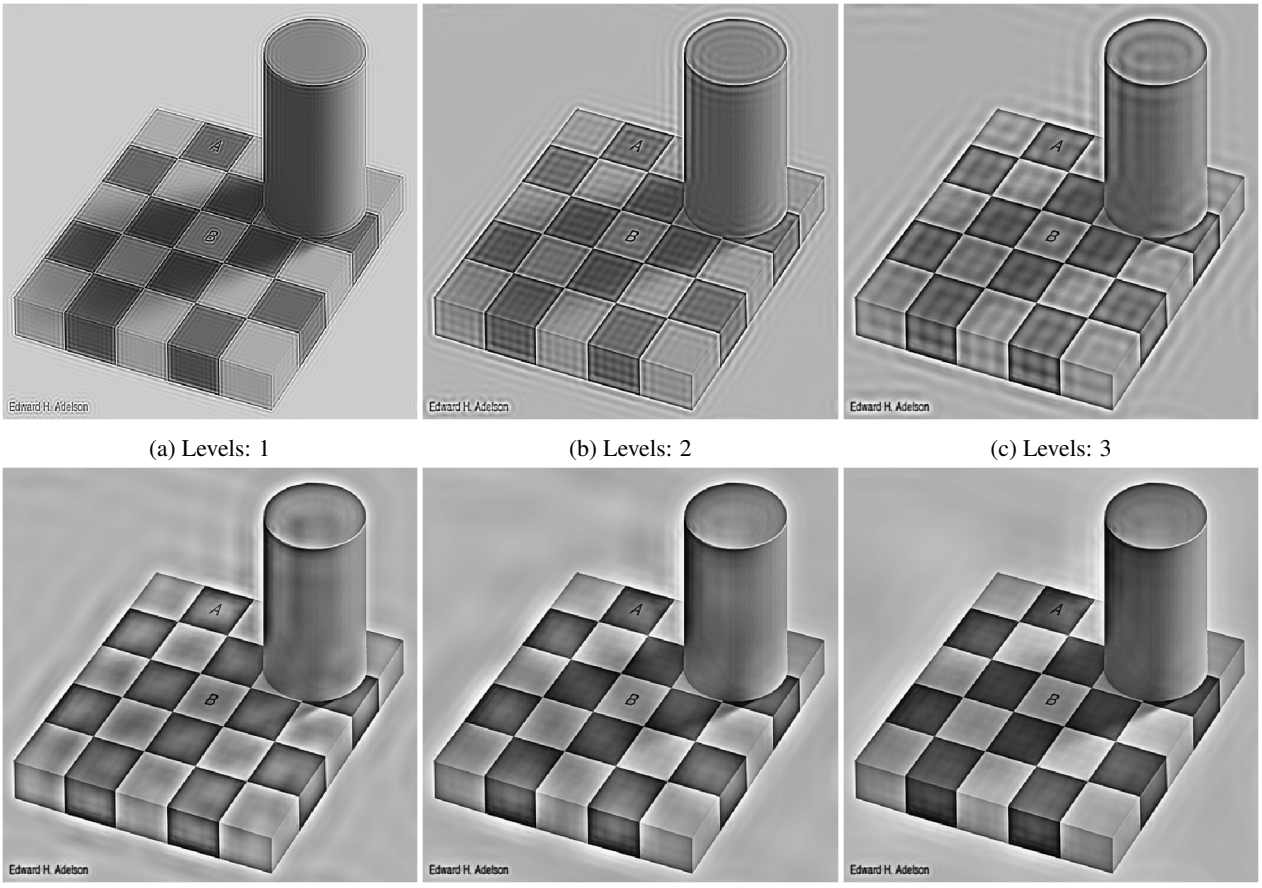
4.5 PhaseAnalyzer

Here we show the results of the test itkRieszWaveletPhaseAnalysisTest with a couple of optical illusions images, a checker board and a Hermann grid.



(a) Original (b) Intensity thresholding

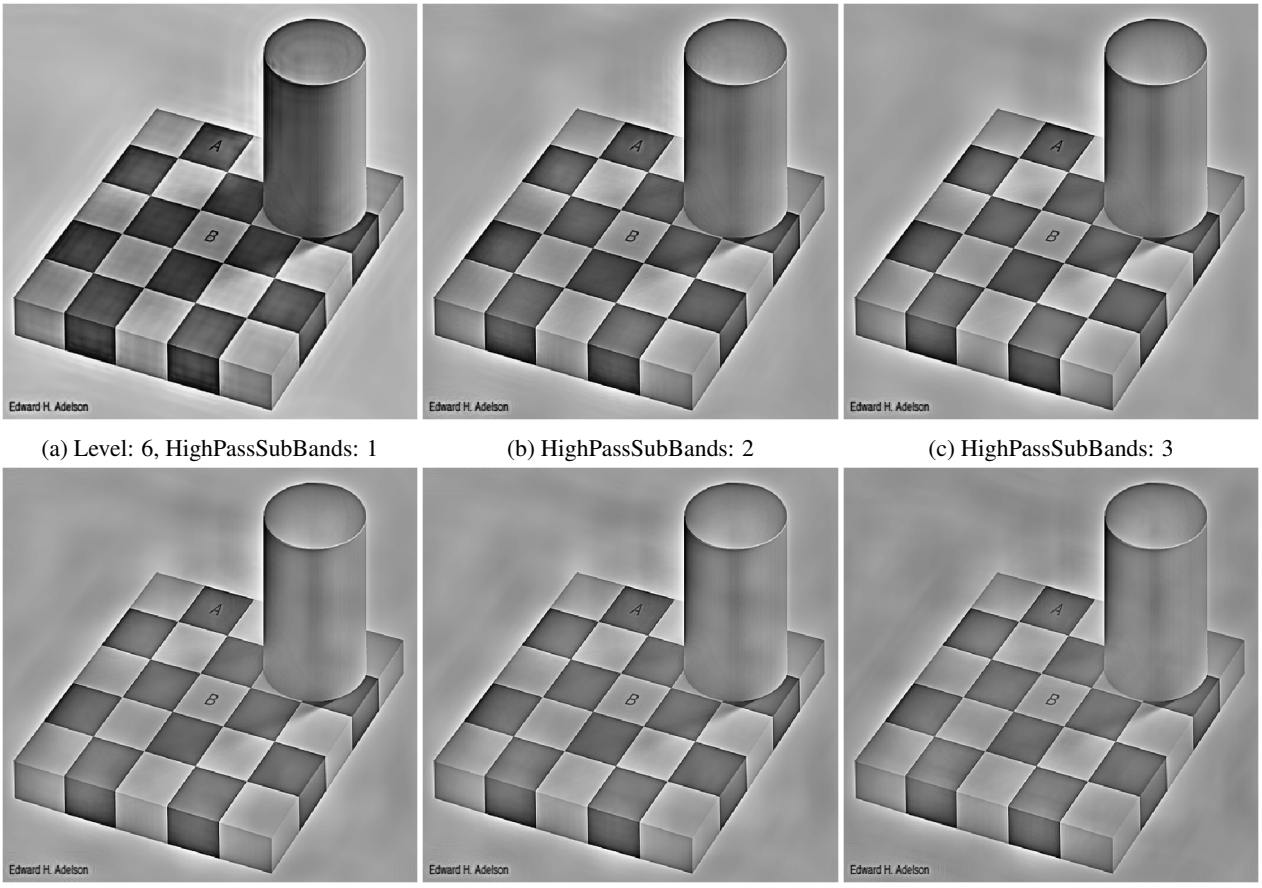
Figure 7: The original image looks like a regular check-board, but it isn't. Pixels in the regions A and B have the same intensity value (129), however our vision system performs local phase analysis that allows us to treat A,B regions as different, keeping a global checker-board structure. 7b uses a non-linear map of intensity-color to enhance the irregular checker board structure.



(a) Levels: 1 (b) Levels: 2 (c) Levels: 3

(d) Levels: 4 (e) Levels: 5 (f) Levels: 6

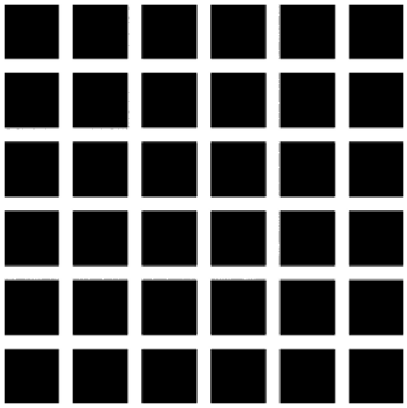
Figure 8: (Results of the phase analysis (with soft threshold) for different number of scales in the wavelet pyramid. The input image is a checker-board of size $512x512$).



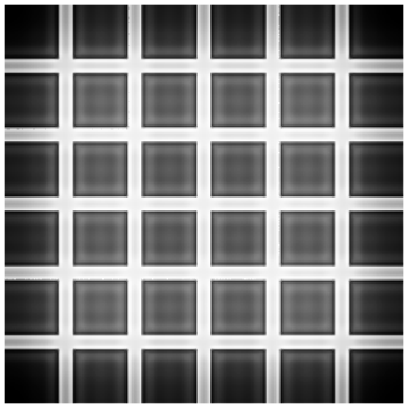
(a) Level: 6, HighPassSubBands: 1 (b) HighPassSubBands: 2 (c) HighPassSubBands: 3

(d) HighPassSubBands: 4 (e) HighPassSubBands: 5 (f) HighPassSubBands: 10

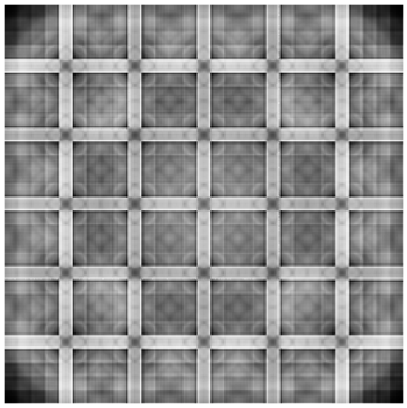
Figure 9: Using six scales (Level: 6), results for different number of high frequency sub-bands.



(a) Hermann Grid



(b) Levels: 8 (max), HighPassSubBands: 1



(c) Levels: 8 (max), HighPassSubBands: 10

Figure 10: The optical illusion generated by 10a the Hermann grid is generated by the local phase analysis of our vision. 10b is the result of a phase analysis with the Monogenic signal for a wavelet pyramid of 8 levels and only one high pass sub band. 10c is the same pyramid but with 10 high pass bands to increase the frequency resolution. Both results use Simoncelli wavelet, but there is not much difference using Held, or Vow mother wavelets.

5 Conclusion and future work

This work provides ITK with a multiresolution analysis based on wavelet decomposition using isotropic and steerable wavelets. Also it provides utilities to work in the dual or frequency domain. Frequency iterators, shrinkers and expanders. It also provides a sub-sampler without interpolation, and a expander with zeros that work in any domain.

As an application, we showed in Figure 10 a local phase analyzer using the wavelet coefficients based on [4].

Future work would be easier to implement with the tools already developed. Some work will only require plumbing operations together in a new filter, for example, creating a specific class for the Simoncelli Steerable Framework [2], and the more general Steerable framework using Riesz transform [6], that will use the already implemented itk::RieszRotationMatrix.

More applications that will require extra work but use the same wavelet backbone are:

- Denoising algorithms can be developed using the exposed wavelet coefficients, for example: Gaussian Scale Mixture Models (GSM)[18], or SURE-LET [19].
- Feature detection without using template matching in an efficient way [20].

Implementation of these algorithms will provide state of the art solutions based on wavelets to a wider audience, and I am happy to link them from the External Repository, https://github.com/phcerdan/ITKIsotropicWavelets to have a reference of wavelet solutions for ITK.

Already implemented but not used in any application yet are: itk::RieszRotationMatrix, providing a steerable framework for the General Riesz Transform [6]. And itk::StructureTensor, a local estimator of the direction –steer– with the highest contribution [5].

6 Acknowledgements

I would like to thank Jon Haitz Legarreta for improving tests and documentation of the module.

References

[1] W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906, September 1991. 03462. 1.1, 2.3

[2] Eero P. Simoncelli and William T. Freeman. The steerable pyramid: A flexible architecture for multi-scale derivative computation. In *ICIP (3)*, pages 444–447, 1995. 00964. 1.1, 1.5, 2, 2.3, 5

[3] Peter Kovesi. Image features from phase congruency. *Videre: Journal of computer vision research*, 1(3):1–26, 1999. 01142. 1.1, 2.1

[4] S. Held, M. Storath, P. Massopust, and B. Forster. Steerable Wavelet Frames Based on the Riesz Transform. *IEEE Transactions on Image Processing*, 19(3):653–667, March 2010. 00062. 1.1, 1.4, 1, 1.5, 2, 2.2, 2.3, 5

[5] N. Chenouard and M. Unser. 3D Steerable Wavelets in Practice. *IEEE Transactions on Image Processing*, 21(11):4522–4533, November 2012. 00016. 1.1, 1, 2.3, 5

[6] M. Unser, N. Chenouard, and D. Van De Ville. Steerable Pyramids and Tight Wavelet Frames in $L_2(\mathcal{R}^d)$. *IEEE Transactions on Image Processing*, 20(10):2705–2721, October 2011. 00000. 1.1, 1.5, 1, 2, 2.2, 2.2, 2.3, 3.4, 3.5, 5

[7] Pedram Pad, Virginie Uhlmann, and Michael Unser. VOW: Variance-optimal wavelets for the steerable pyramid. *constraints*, 3:4, 2014. 00007. 1.1, 1.5

[8] Stephane G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE transactions on pattern analysis and machine intelligence*, 11(7):674–693, 1989. 21999. 1.1

[9] Christopher E. Heil and David F. Walnut. Continuous and discrete wavelet transforms. *SIAM review*, 31(4):628–666, 1989. 01229. 1.1

[10] Tao Qian, Mang I. Vai, and Yuesheng Xu. *Wavelet Analysis and Applications*. Springer Science & Business Media, February 2007. Google-Books-ID: oLG9BAAAQBAJ. 1.1

[11] Juan R. Romero, Simon K. Alexander, Shikha Baid, Saurabh Jain, and Manos Papadakis. The geometry and the analytic properties of isotropic multiresolution analysis. *Advances in Computational Mathematics*, 31(1-3):283–328, October 2009. 1.5

[12] Ingrid Daubechies. *Ten Lectures on Wavelets*. SIAM: Society for Industrial and Applied Mathematics, Philadelphia, Pa, 1 edition edition, May 1992. 24597. 1.5

[13] Akram Aldroubi, Carlos Cabrelli, and Ursula M. Molter. Wavelets on Irregular Grids with Arbitrary Dilation Matrices, and Frames Atoms for $L^2(R^d)$. *arXiv:math/0703438*, March 2007. 00000. 1

[14] Javier Portilla and Eero P. Simoncelli. Image denoising via adjustment of wavelet coefficient magnitude correlation. In *Image Processing, 2000. Proceedings. 2000 International Conference On*, volume 3, pages 277–280. IEEE, 2000. 00044. 1.5

[15] M. Unser, D. Sage, and D. Van De Ville. Multiresolution Monogenic Signal Analysis Using the Riesz-Laplace Wavelet Transform. *IEEE Transactions on Image Processing*, 18(11):2402–2418, November 2009. 00000. 2, 3.6

[16] Michael Felsberg and Gerald Sommer. The monogenic signal. *IEEE Transactions on Signal Processing*, 49(12):3136–3144, 2001. 00593. 2.1, 3.6

[17] M. Unser and D. Van De Ville. Wavelet Steerability and the Higher-Order Riesz Transform. *IEEE Transactions on Image Processing*, 19(3):636–652, March 2010. 00051. 2.2

[18] Javier Portilla, Vasily Strela, Martin J. Wainwright, and Eero P. Simoncelli. Image denoising using scale mixtures of Gaussians in the wavelet domain. *IEEE Transactions on Image processing*, 12(11):1338–1351, 2003. 02184. 3.1, 5

[19] T.. Blu and F.. Luisier. The SURE-LET Approach to Image Denoising. *IEEE Transactions on Image Processing*, 16(11):2778–2786, November 2007. 00272. 5

[20] Zsuzsanna Puspoki and Michael Unser. Template-Free Wavelet-Based Detection of Local Symmetries. *IEEE Transactions on Image Processing*, 24(10):3009–3018, October 2015. 00004. 5