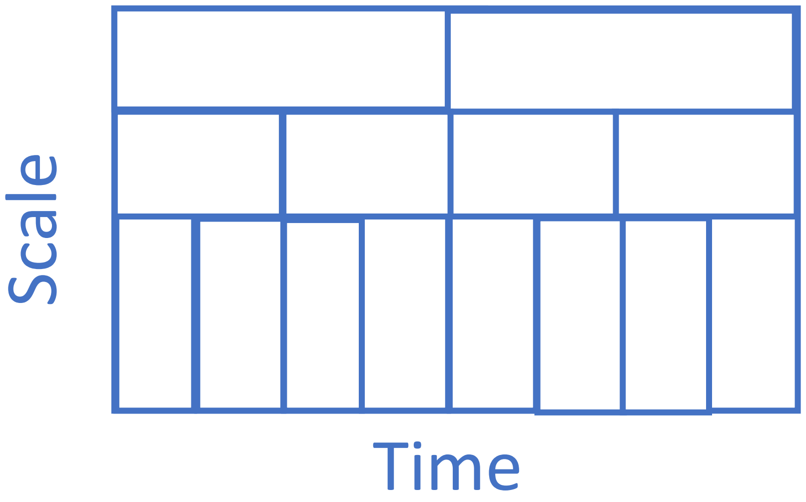
Portraying a signal (wavelet)



Obtained from the function $\psi(t)$, known as the wavelet.

$$F(\tau, s) = \frac{1}{\sqrt{|s|}} \int_{-\infty}^{+\infty} f(t) \boxed{\psi^* \left(\frac{t - \tau}{s} \right)} dt$$

Wavelet (kernel, level)

- Expanded kernels (large values of s) = we lose temporal information but gain good information on the low frequencies (such as a homogeneous xCH_4 field at background).
- Shrunken kernels (small values of s) = better temporal information; we lose low frequencies but gain insights into high frequencies (small-scale fluctuations).
- In discrete form, s is equal to 2^{-j} (j is the index of levels).

scaling function

$$W_\phi[j_0, k] = \frac{1}{\sqrt{M}} \sum_n f[n]\phi_{j_0,k}[n].$$

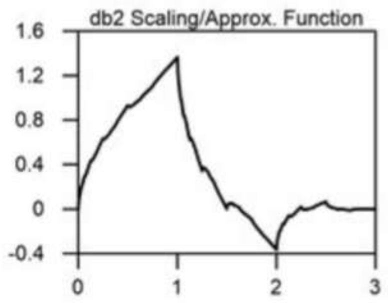
Approximation (low pass)

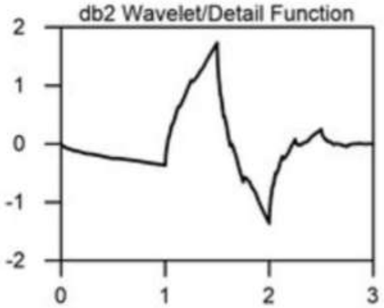
Wavelet function

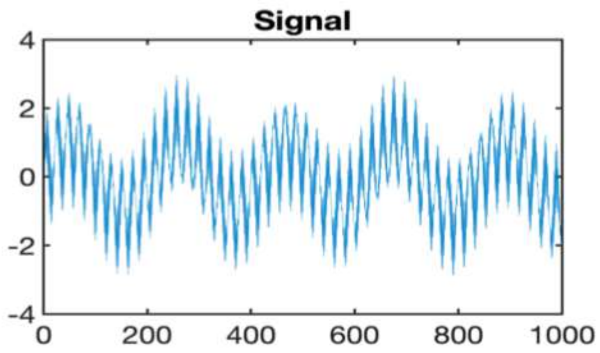
$$W_\psi[j, k] = \frac{1}{\sqrt{M}} \sum_n f[n]\psi_{j,k}[n]$$

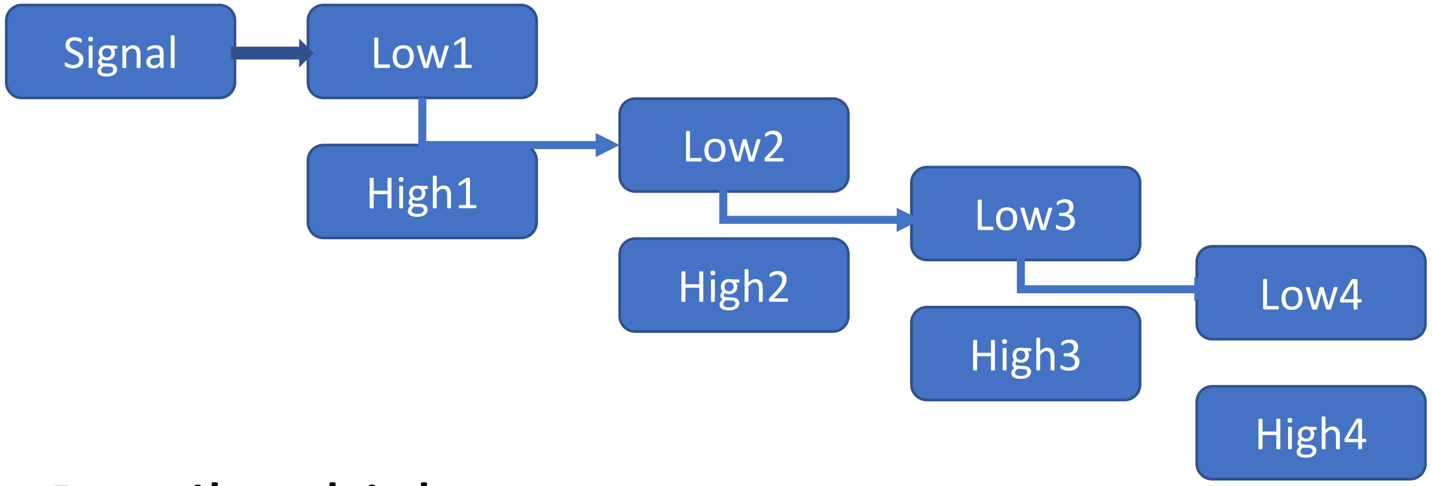
Details (high pass)

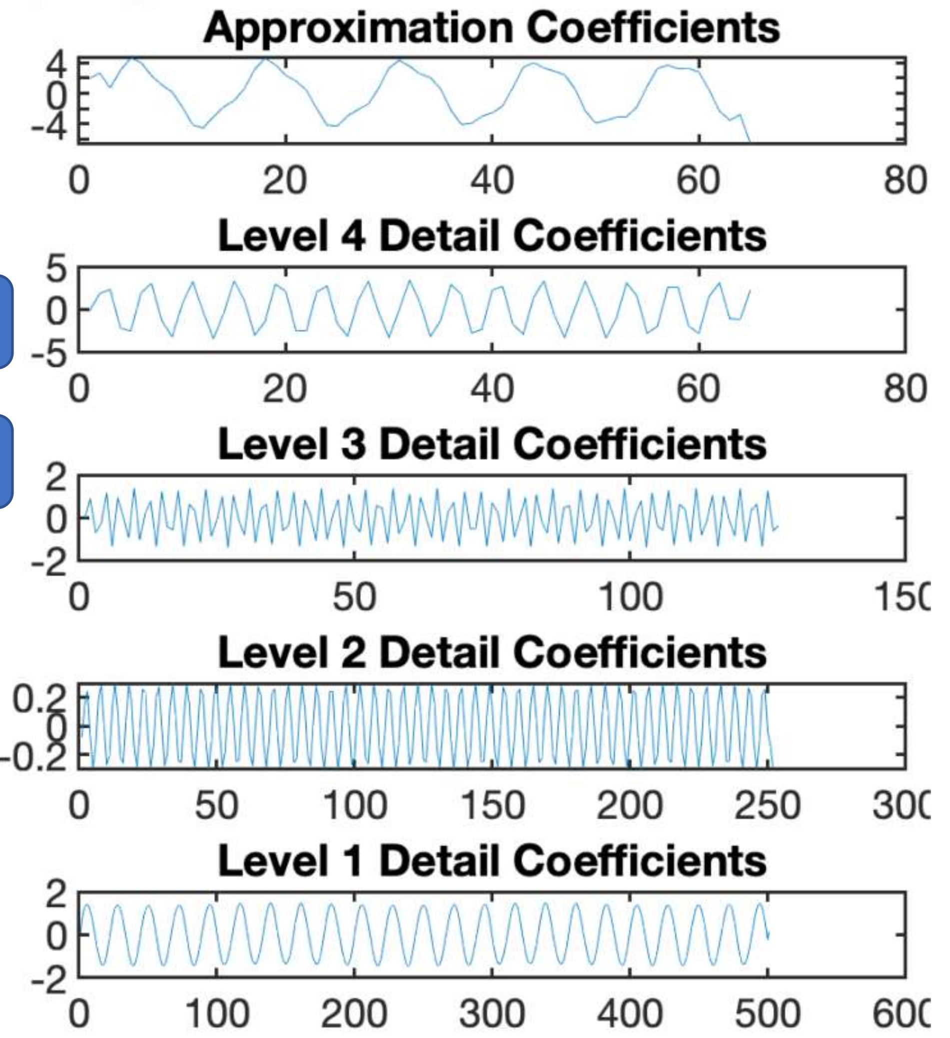
1-D











- Details = highs
- Approximation = Low4
- 64(j=4) 128(j=3) 250(j=2) 500(j=1)

The signal can be reconstructed by

$$f[n] = \frac{1}{\sqrt{M}} \sum_k W_\phi[j_0, k]\phi_{j_0,k}[n] + \frac{1}{\sqrt{M}} \sum_{j=j_0}^{\infty} \sum_k W_\psi[j, k]\psi_{j,k}[n].$$

2D

- The details consist of three components:

$$W_\phi(j_0,m,n) = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)\phi_{j_0,m,n}(x,y), \quad (3.36)$$

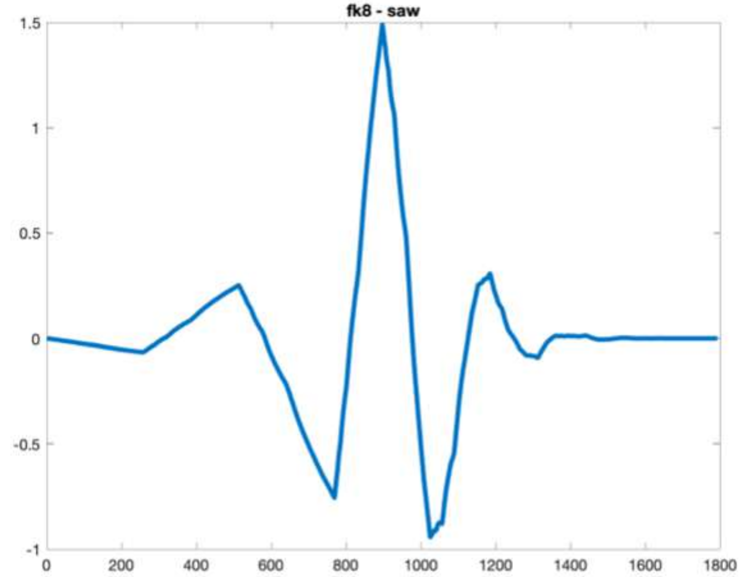
$$W_\psi^i(j,m,n) = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)\psi^i_{j,m,n}(x,y), i = \{H,V,D\} \quad (3.37)$$

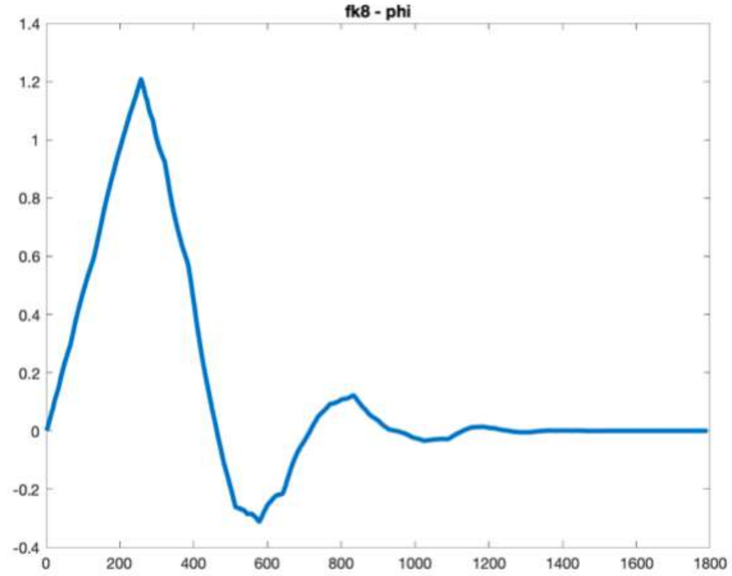
$$f(x,y) = \frac{1}{\sqrt{MN}} \sum_m \sum_n W_\phi(j_0,m,n)\phi_{j_0,m,n}(x,y) + \frac{1}{\sqrt{MN}} \sum_{i=H,V,D} \sum_{j=j_0}^{\infty} \sum_m \sum_n W_\psi^i(j,m,n)\psi^i_{j,m,n}(x,y) \quad (3.38)$$

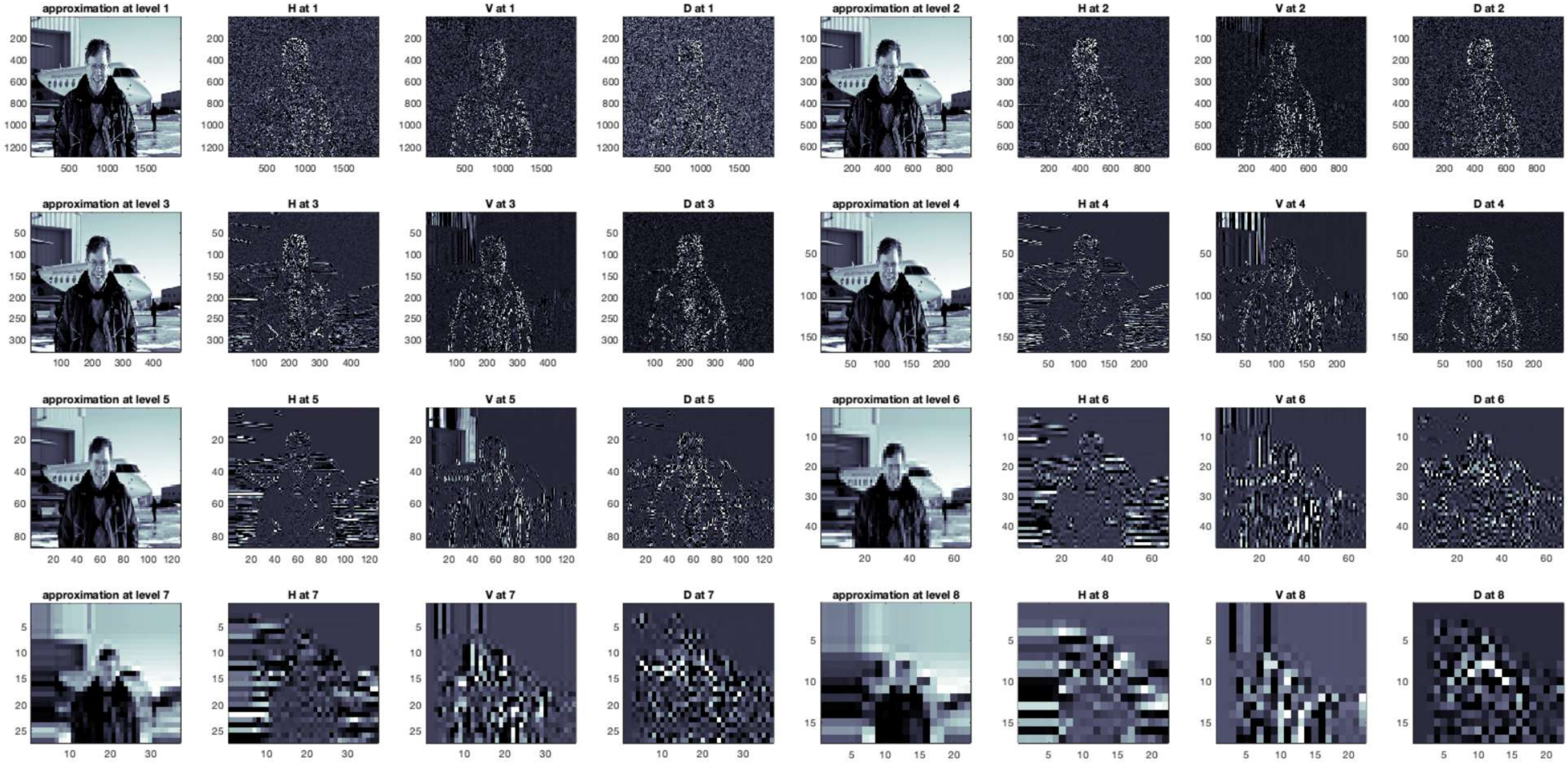
$$\begin{aligned} \phi(x,y) &= \phi(x)\phi(y), \\ \psi^H(x,y) &= \psi(x)\phi(y), \\ \psi^V(x,y) &= \phi(x)\psi(y), \\ \psi^D(x,y) &= \psi(x)\psi(y). \end{aligned}$$

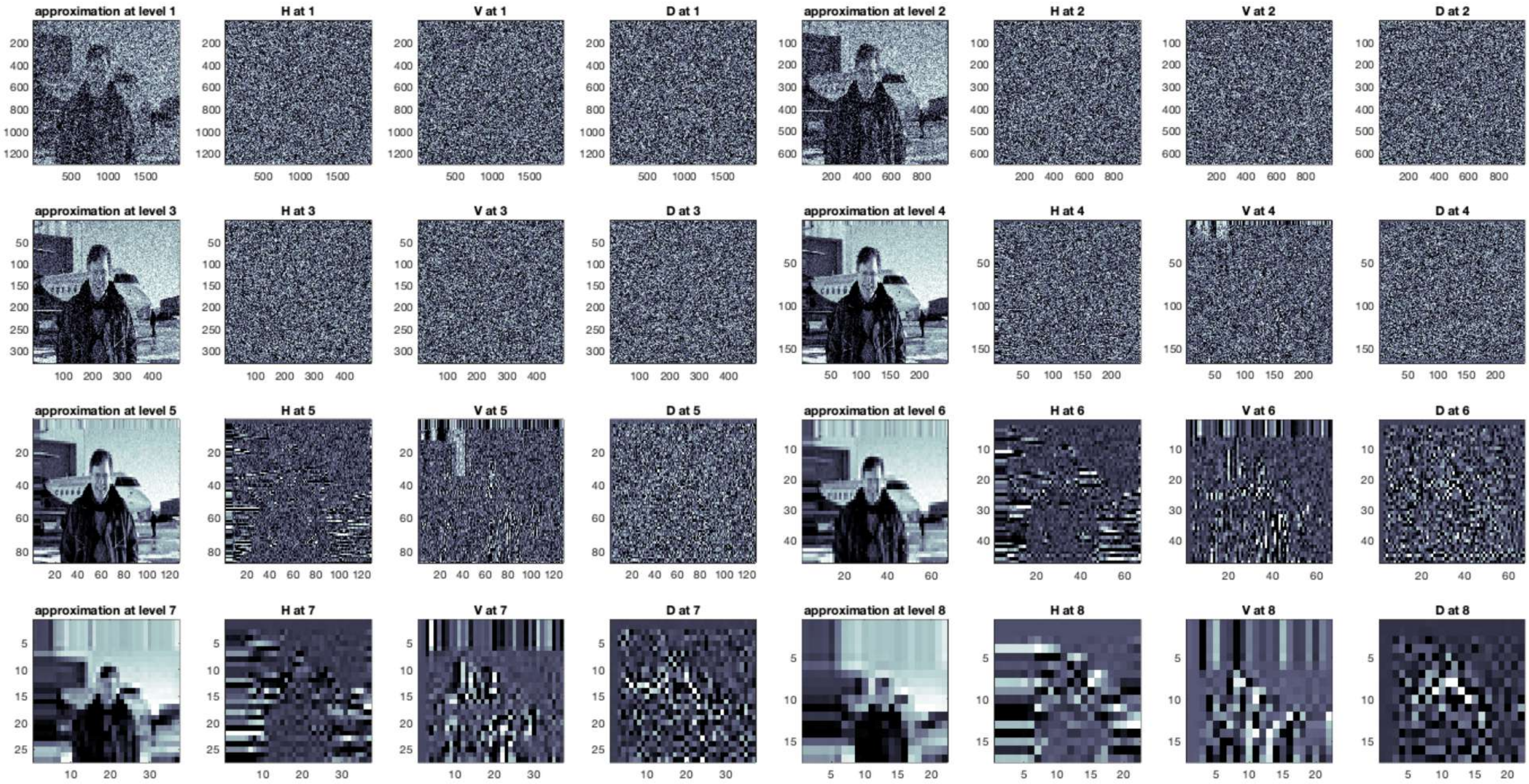
Ref: http://disp.ee.ntu.edu.tw/tutorial/WaveletTutorial.pdf

2D wavelet (sample: steve.jpg)



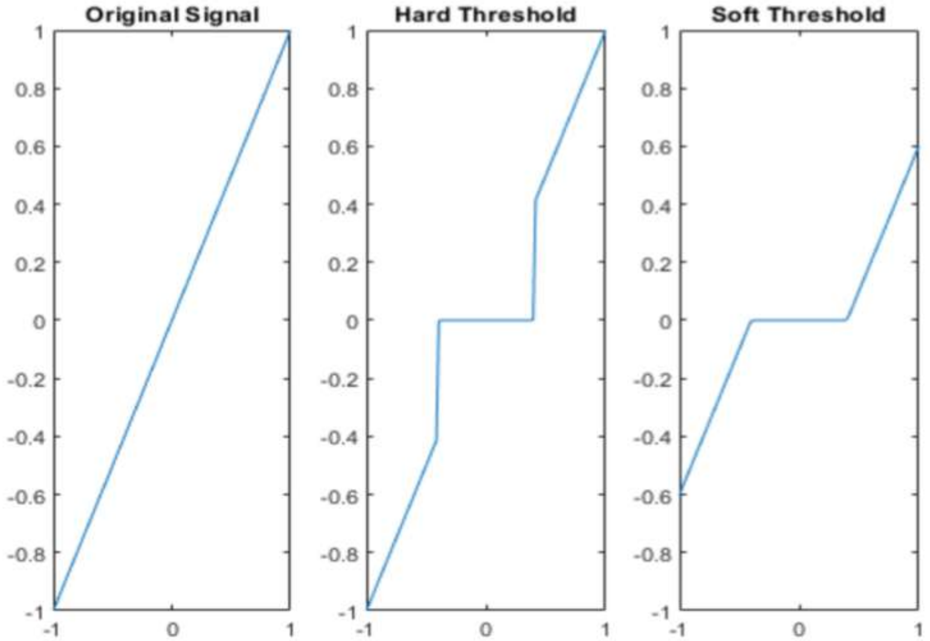


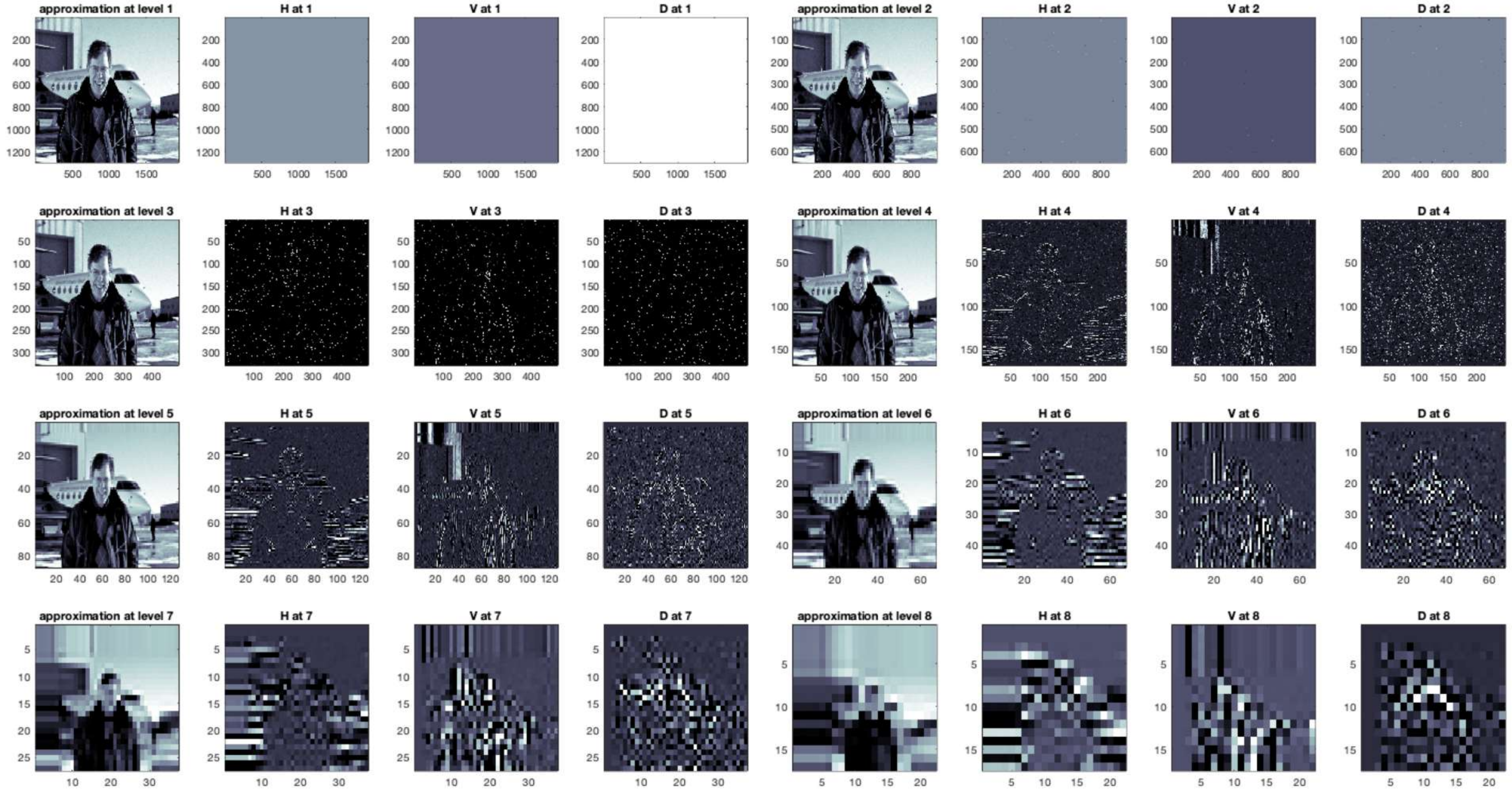

approximation at level 1
H at 1
V at 1
D at 1
approximation at level 2
H at 2
V at 2
D at 2
approximation at level 3
H at 3
V at 3
D at 3
approximation at level 4
H at 4
V at 4
D at 4
approximation at level 5
H at 5
V at 5
D at 5
approximation at level 6
H at 6
V at 6
D at 6
approximation at level 7
H at 7
V at 7
D at 7
approximation at level 8
H at 8
V at 8
D at 8


approximation at level 1
200 400 600 800 1000 1200
500 1000 1500
H at 1
200 400 600 800 1000 1200
500 1000 1500
V at 1
200 400 600 800 1000 1200
500 1000 1500
D at 1
200 400 600 800 1000 1200
500 1000 1500
approximation at level 2
100 200 300 400 500 600
200 400 600 800
H at 2
100 200 300 400 500 600
200 400 600 800
V at 2
100 200 300 400 500 600
200 400 600 800
D at 2
100 200 300 400 500 600
200 400 600 800
approximation at level 3
50 100 150 200 250 300
100 200 300 400
H at 3
50 100 150 200 250 300
100 200 300 400
V at 3
50 100 150 200 250 300
100 200 300 400
D at 3
50 100 150 200 250 300
100 200 300 400
approximation at level 4
50 100 150
50 100 150 200
H at 4
50 100 150
50 100 150 200
V at 4
50 100 150
50 100 150 200
D at 4
50 100 150
50 100 150 200
approximation at level 5
20 40 60 80
20 40 60 80 100 120
H at 5
20 40 60 80
20 40 60 80 100 120
V at 5
20 40 60 80
20 40 60 80 100 120
D at 5
20 40 60 80
20 40 60 80 100 120
approximation at level 6
10 20 30 40
20 40 60
H at 6
10 20 30 40
20 40 60
V at 6
10 20 30 40
20 40 60
D at 6
10 20 30 40
20 40 60
approximation at level 7
5 10 15 20 25
10 20 30
H at 7
5 10 15 20 25
10 20 30
V at 7
5 10 15 20 25
10 20 30
D at 7
5 10 15 20 25
10 20 30
approximation at level 8
5 10 15
5 10 15 20
H at 8
5 10 15
5 10 15 20
V at 8
5 10 15
5 10 15 20
D at 8
5 10 15
5 10 15 20

Which thresholds?

- There are myriad ways to determine the level of noise (unknown) in a signal.
- We use SUREShrink (Donoho and Johnstone, 1995). It provides an unbiased estimate of the mean-squared error of the signal at different levels.
- Steve’s noisy image thresholds:
  - 200.2582(1) 137.3754(2) 101.2476(3)
  - 56.4954(4) 33.9540(5) 20.0959(6)
  - 10.8232(7) 11.1459(8)




approximation at level 1
H at 1
V at 1
D at 1
approximation at level 2
H at 2
V at 2
D at 2
approximation at level 3
H at 3
V at 3
D at 3
approximation at level 4
H at 4
V at 4
D at 4
approximation at level 5
H at 5
V at 5
D at 5
approximation at level 6
H at 6
V at 6
D at 6
approximation at level 7
H at 7
V at 7
D at 7
approximation at level 8
H at 8
V at 8
D at 8

Before, after

MSATwtdenoiser

https://github.com/ahsouri/MSATwtdenoiser/blob/main/MSATwtdenoiser/MSATwtdenoiser.py

First step: decompose the image into H,V, and D for the defined number of levels

```python
def signal2wt(self,image,wtname,level):
    '''
    Applying wavelet transform to the image (signal)
    ARGS:
        image[m,n] (float), np.array
        wtname (char) -> e.g, 'db2'
        level (int)
    OUTs:
        coeff [level] (list): wavelet approximation(level=0) and details (>0)
    '''
    import pywt
    import numpy as np

    coeffs = pywt.wavedec2(image, wtname,level=level)

    return coeffs
```

https://pywavelets.readthedocs.io/en/latest/ref/wavelets.html

Estimate the noise level for detail coefficients

- We have three details and n levels. So we will estimate noise threshold for 3*n

```
def wtdenoiser(self,coeffs2,wtname,level):
    '''
    Removing details based on the SURE threshold
    ARGS:
        image[m,n] (float)
        wtname (char) -> e.g, 'db2'
        level (int)
    OUTs:
        denoised[m,n] (float): denoised image
    '''
    import numpy as np
    import pywt
    from scipy.special import erfcinv
    from cv2 import resize
    from cv2 import INTER_NEAREST
```

Estimate the noise level for detail coefficients

```
# varwt
normfac = -np.sqrt(2)*erfcinv(2*0.75)
# finding thresholds for each level
thr = np.zeros((level,1))
cfs_denoised = []
cfs_denoised.append(coeffs2[0]) #approx
for lev in range(level):
    cfs = coeffs2[lev+1]
    sigmaest = np.median(np.abs(cfs))*(1/normfac)
    thr = self.ThreshSURE(cfs/sigmaest)
    thr = sigmaest*thr
    cfs_denoised.append(list(pywt.threshold(cfs,thr,'soft'))) #details
    #denoised = pywt.idwt2(list(pywt.threshold(cfs,thr,'soft')),wtname)
```

= N(0, 1)

Loop over levels

details

The sigma of details

Finding the noise thresholds

Masking values below threshold based on a soft threshold

ThreshSURE

It follow SureShrink based on ”Adapting to Unknown Smoothness via Wavelet Shrinkage” https://www.jstor.org/stable/2291512?origin=JSTOR-pdf

```python
def ThreshSURE(self,x):
    """
    Threshold based on SURE
    ARGS:
        x[3,n,m] (float): H,V,D details
    OUTs:
        Thr[1x1] (float): Noise/signal threshold
    """
    import numpy as np

    x=x.flatten()
    n = np.size(x)
    sx = np.sort(np.abs(x))
    sx2 = sx**2
    n1 = n-2*np.arange(0,n,1)
    n2 = np.arange(n-1,-1,-1)
    cs1 = np.cumsum(sx2,axis=0)
    risk = (n1+cs1+n2*sx2)/n
    ibest = np.argmin(risk)
    thr = sx[ibest]

    return thr
```

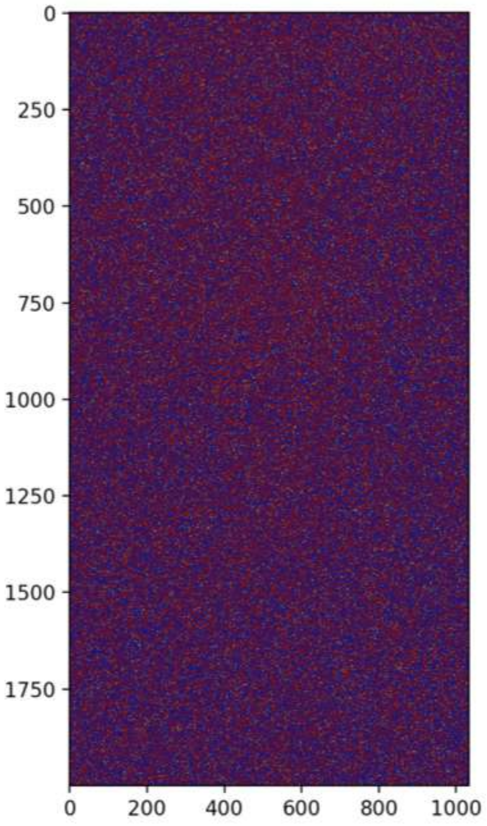
Reconstructing

- Once details have been removed based on abs(values)<thresholds for each level, we can easily put the denoised coefficients back together to reconstruct the image using (see the third eq in slide 4):

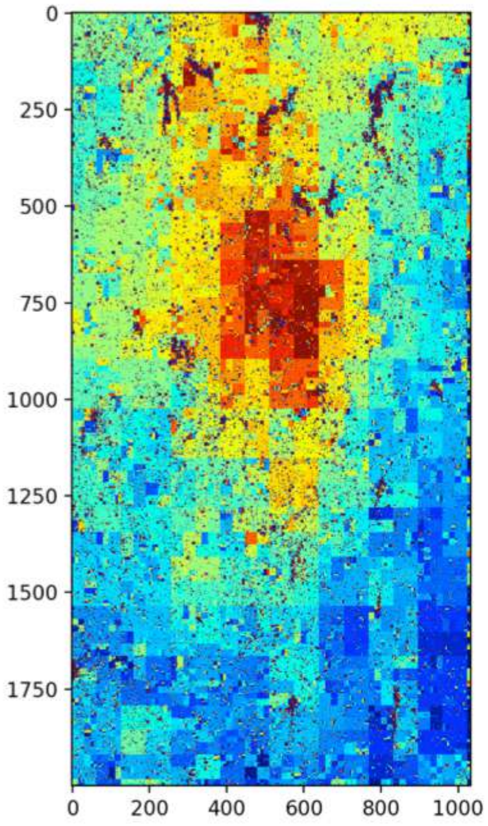
```
cfs_denoised.append(list(pywt.threshold(cfs,thr,'soft'))) #details
#denoised = pywt.idwt2(list(pywt.threshold(cfs,thr,'soft')),wtname)
```

```
#reconstruct the signal
denoised = pywt.waverec2(cfs_denoised,wtname)
```

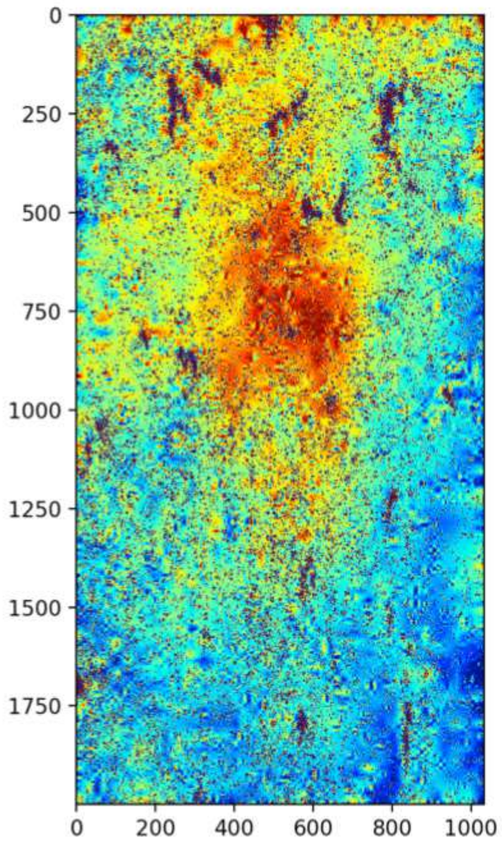
Example:



db1



db3



db6