Algorithme Á Trous ("hole algorithm")

Begin with image I (2D array of intensities)

This reflects my own implementation....

$$K = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}; \quad K^n \equiv \mathcal{F}^{-1}\left[(\mathcal{F}(K))^n\right]$$

$$B_j \equiv \begin{cases} I, & j = 0 \\ I * K^{\alpha_j}, & \alpha_j \equiv 2^{j-1}, \quad 1 \le j \le N \end{cases}$$

Aside: if kernel K has width σ, then the blurring width of B_j is:

$$\sigma_j = \begin{cases} 0, & j = 0 \\ \sqrt{\alpha_j \sigma^2} = 2^{(j-1)/2}\sigma, & j > 1 \end{cases}$$

$$= 0, \sigma, \sqrt{2}\sigma, 2\sigma, 2\sqrt{2}\sigma, \ldots$$

The wavelet transform W of I is then:

$$W_{j+1} \equiv \begin{cases} B_j - B_{j+1}, & 0 \le j \le N-1 \\ B_j = B_N, & j = N \end{cases}$$

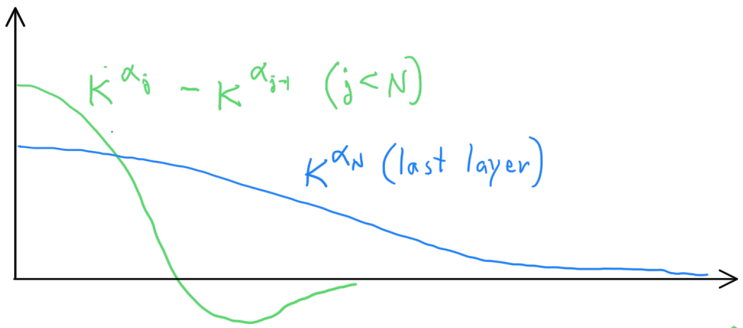
Matlab/Octave

Inverse:

$$\sum_{j=1}^{N+1} W_j = (B_0 - B_1) + (B_1 - B_2) + \cdots + (B_{N-1} - B_N) + B_N$$

$$= B_0 = I$$

The basis elements are:



In Fourier space, the basis elements $(j < N)$ look like donuts (beignets). So maybe I can call it Algorithme á Beignet.