

# Denoising Images Project

---

ELABORAZIONE DI IMMAGINI

DARIO CIVALE - 0622701620 - [D.CIVALE1@STUDENTI.UNISA.IT](mailto:D.CIVALE1@STUDENTI.UNISA.IT)

PAOLO MANSI - 0622701542 - [P.MANSI5@STUDENTI.UNISA.IT](mailto:P.MANSI5@STUDENTI.UNISA.IT)

VINCENZO SALVATI - 0622701550 - [V.SALVATI10@STUDENTI.UNISA.IT](mailto:V.SALVATI10@STUDENTI.UNISA.IT)

# Guided Filter

Filter which use a **guided image** to filter the input.

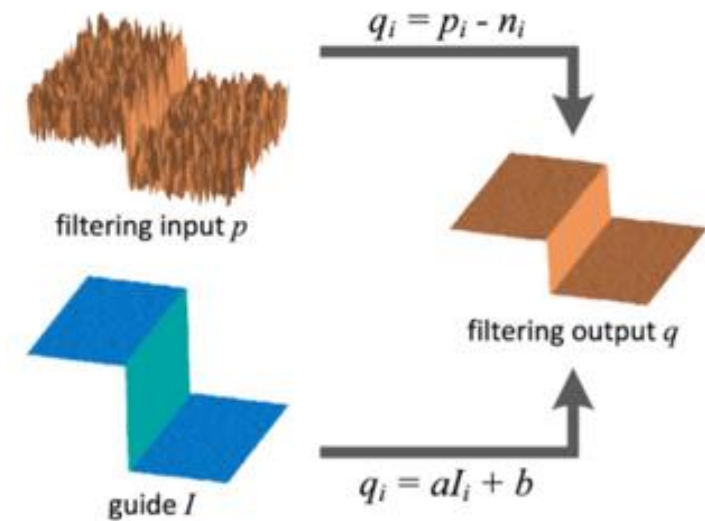
Main use:

- texture transfer;
- **denoising**;
- etc...

## *Local Linear Model*

$$q_i = a_k * I_i + b_k, \forall i \in w_k$$

Edge preserved:  $\nabla q = \nabla I$



$q$  = output  
 $p$  = input  
 $a, b$  = unknown parameters  
 $I$  = guided image

# Parameters

---

**Cost function** to be minimized:

$$E(a_k, b_k) = \sum_{i \in w_k} ((a_k * I_k + b_k - p_i)^2 + \varepsilon * a_k^2), \text{ where } \varepsilon \text{ is a regularization parameter}$$

**Deriving the function**, we get the parameters :

$$a_k = \frac{\frac{1}{|w_k|} * \sum_{i \in w_k} I_i * p_i - \mu_k * \bar{p}_k}{\sigma_k^2 + \varepsilon} \quad b_k = \bar{p}_k - a_k * \mu_k, \text{ where:}$$

$\mu_k$  = mean of  $I$  in  $w_k$   
 $\sigma_k^2$  = variance of  $I$  in  $w_k$   
 $\bar{p}$  = mean of  $p$  in  $w_k$   
 $|w_k|$  = pixels in the window

Because of **overlapping windows for each pixel**, we consider the **mean of a and b**:

$$\bar{a}_i = \frac{1}{|w_k|} \sum_{k \in w_i} a_k$$

$$\bar{b}_i = \frac{1}{|w_k|} \sum_{k \in w_i} b_k$$



$$q_i = \bar{a}_i * I_i + \bar{b}_i$$

# Denoising

---

Denoising predicts that  $\mathbf{I} = \mathbf{p}$ :

$$a_k = \frac{\sigma_k^2}{\sigma_k^2 + \varepsilon} \quad b_k = (1 - a_k) * \mu_k$$

$$\varepsilon = 0 \rightarrow a_k = 1, b_k = 0$$

$$\varepsilon > 0 \left\{ \begin{array}{ll} \sigma_k^2 \gg \varepsilon \rightarrow a_k \approx 1, b_k \approx 0 & \text{High variance: preserving edge} \\ \sigma_k^2 \ll \varepsilon \rightarrow a_k \approx 0, b_k \approx \mu_k & \text{Flat Patch: smoothing} \end{array} \right.$$

# Performance indexes b&w

## Gaussian

Filtro (med e var)



- MSE: 0.01  
- PSNR: 69.09  
- SSIM: 0.54

Filtro OpenCV



- MSE: 0.01  
- PSNR: 68.33  
- SSIM: 0.46

## Salt and Pepper

Filtro (med e var)



- MSE: 0.01  
- PSNR: 66.39  
- SSIM: 0.48

Filtro OpenCV



- MSE: 0.02  
- PSNR: 66.17  
- SSIM: 0.47

## Poisson

Filtro (med e var)



- MSE: 0.00  
- PSNR: 74.39  
- SSIM: 0.79

Filtro OpenCV



- MSE: 0.00  
- PSNR: 75.11  
- SSIM: 0.76

## Speckle

Filtro (med e var)



- MSE: 0.00  
- PSNR: 72.72  
- SSIM: 0.71

Filtro OpenCV



- MSE: 0.00  
- PSNR: 73.49  
- SSIM: 0.73

# Performance indexes RGB

## Gaussian

Filtro (med e var)



- MSE: 0.01  
- PSNR: 69.17  
- SSIM: 0.50

Filtro OpenCV



- MSE: 0.01  
- PSNR: 68.60  
- SSIM: 0.40

## Salt and Pepper

Filtro (med e var)



- MSE: 0.02  
- PSNR: 65.69  
- SSIM: 0.41

Filtro OpenCV



- MSE: 0.02  
- PSNR: 65.87  
- SSIM: 0.42

## Poisson

Filtro (med e var)



- MSE: 0.00  
- PSNR: 72.55  
- SSIM: 0.81

Filtro OpenCV



- MSE: 0.00  
- PSNR: 77.42  
- SSIM: 0.84

## Speckle

Filtro (med e var)



- MSE: 0.00  
- PSNR: 72.16  
- SSIM: 0.81

Filtro OpenCV



- MSE: 0.00  
- PSNR: 76.73  
- SSIM: 0.85

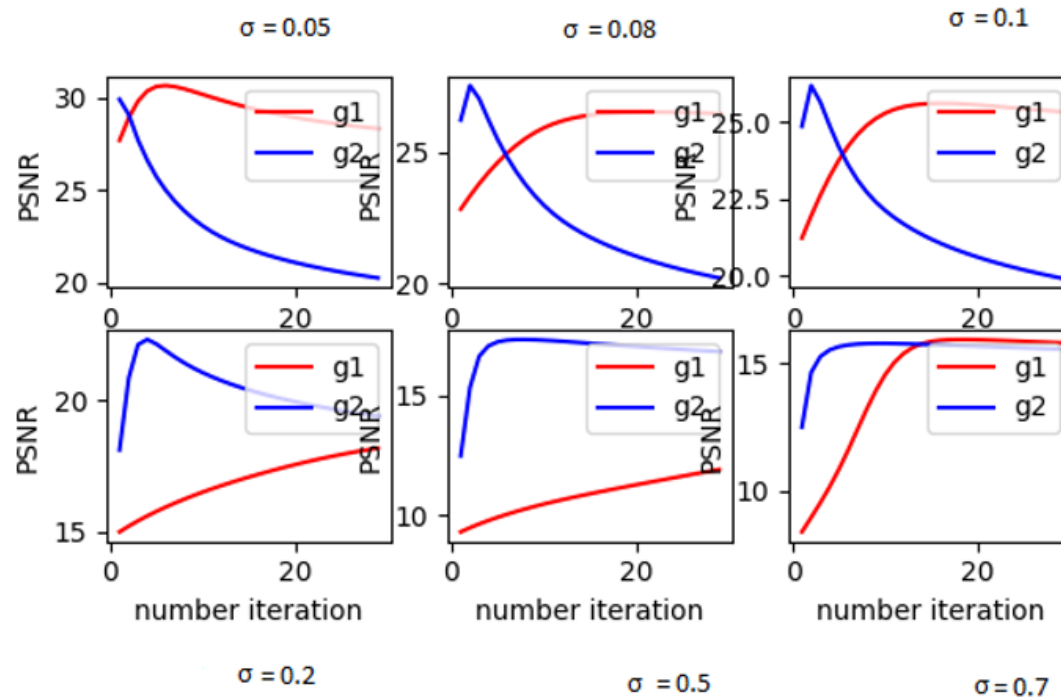
# Anisotropic filter

---

It is a filtering technique that aims to reduce noise in images without removing significant parts of the image content, typically edges, lines, or other details that are important for image interpretation.

$$Img^n = Img^{n-1} + \frac{1}{\eta} \sum_0^c g(\delta_i(Img^{n-1}), K) * \delta_i(Img^{n-1})$$

# Evaluation of conductivity functions



- $g_1 = e^{-\left(\frac{|\delta_i|}{K}\right)^2}$
- $g_2 = 1/(1 + \frac{|\delta_i|^2}{K^2})$



# Automatic calculation of the gradient threshold

---

- Canny noise estimator
- MAD estimator

$$K = 1.4826 * MAD(\nabla \text{Img} - \text{median}(\nabla \text{Img}))$$

- Morphological estimator

$$K = \sum_{i,j \in \text{Img}} \text{Img} \circ \frac{\text{st}}{\text{row}} * \text{col} - \sum_{i,j \in \text{Img}} \text{Img} \bullet \frac{\text{st}}{\text{row}} * \text{col}$$

# Automatic calculation of the number of iterations

---

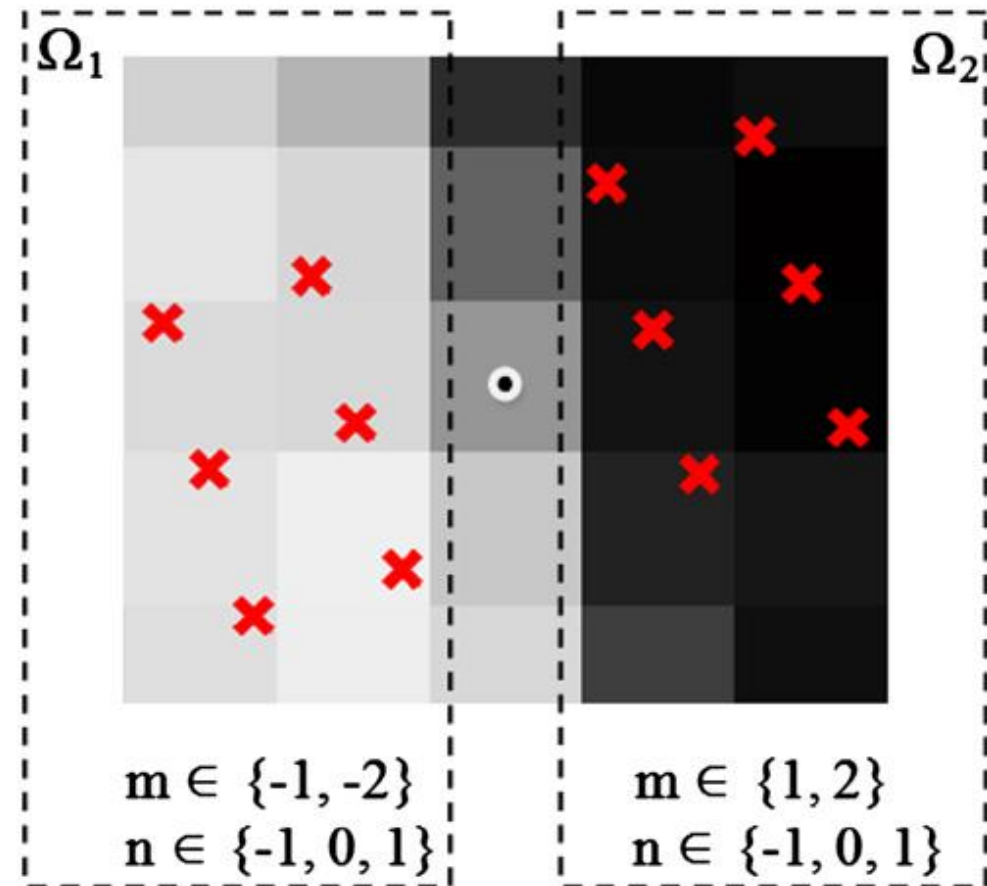
The algorithm consists of several steps:

1. Identification of the N edges

2. For each edge, a local area characterizing the edge is defined, in which the interpixel points are identified

$$\begin{aligned}x_{m,n} &= x_k + m * \cos(\theta) - n * \sin(\theta) & per\ m &= \{-1, -1, 1, 2\}, n = \{-1, 0, 1\} \\y_{m,n} &= y_k - m * \sin(\theta) - n * \cos(\theta) & per\ m &= \{-2, -1, 1, 2\}, n = \{-1, 0, 1\}\end{aligned}$$

### 3. Identification of interpixel regions



4. Calculation of the edge quality function  $Q$  for each edge identified in step 1

$$Q = |\mu_1 - \mu_2| - \alpha * |\sigma_1 + \sigma_2|$$

$$\alpha = 10 * \sigma / \mu_0, \mu_0 = \left(\frac{1}{N}\right) \sum_1^N |\mu_1(t) - \mu_2(t)|$$

5. Calculating the image quality function  $Q_m$  at each iteration

$$Q_m(t) = (1/N) \sum_{i=1}^N Q_i(t)$$

6. Estimation of the iteration instant

$$T = \arg \max_t Q_m(t)$$

# Filter behaviour b&w

Original



Gaussian noise



Morpho



- MSE: 0.01  
- PSNR: 68.65  
- SSIM: 0.69

MAD



- MSE: 0.02  
- PSNR: 66.33  
- SSIM: 0.65

Canny noise estimator



- MSE: 0.01  
- PSNR: 70.53  
- SSIM: 0.66

# Filter behaviour RGB

---

**Original**



**Added Noise: gaussian**



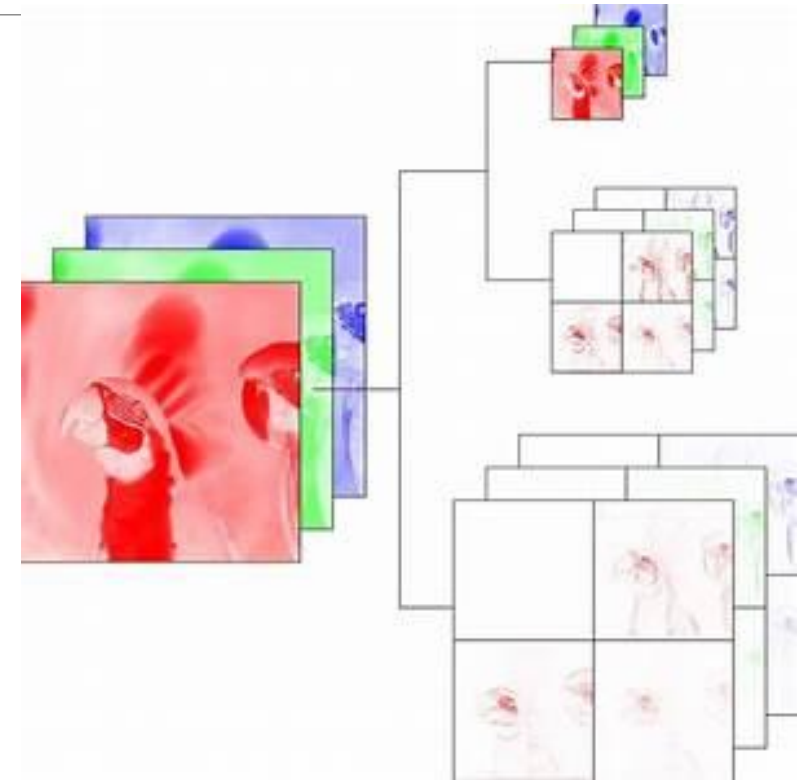
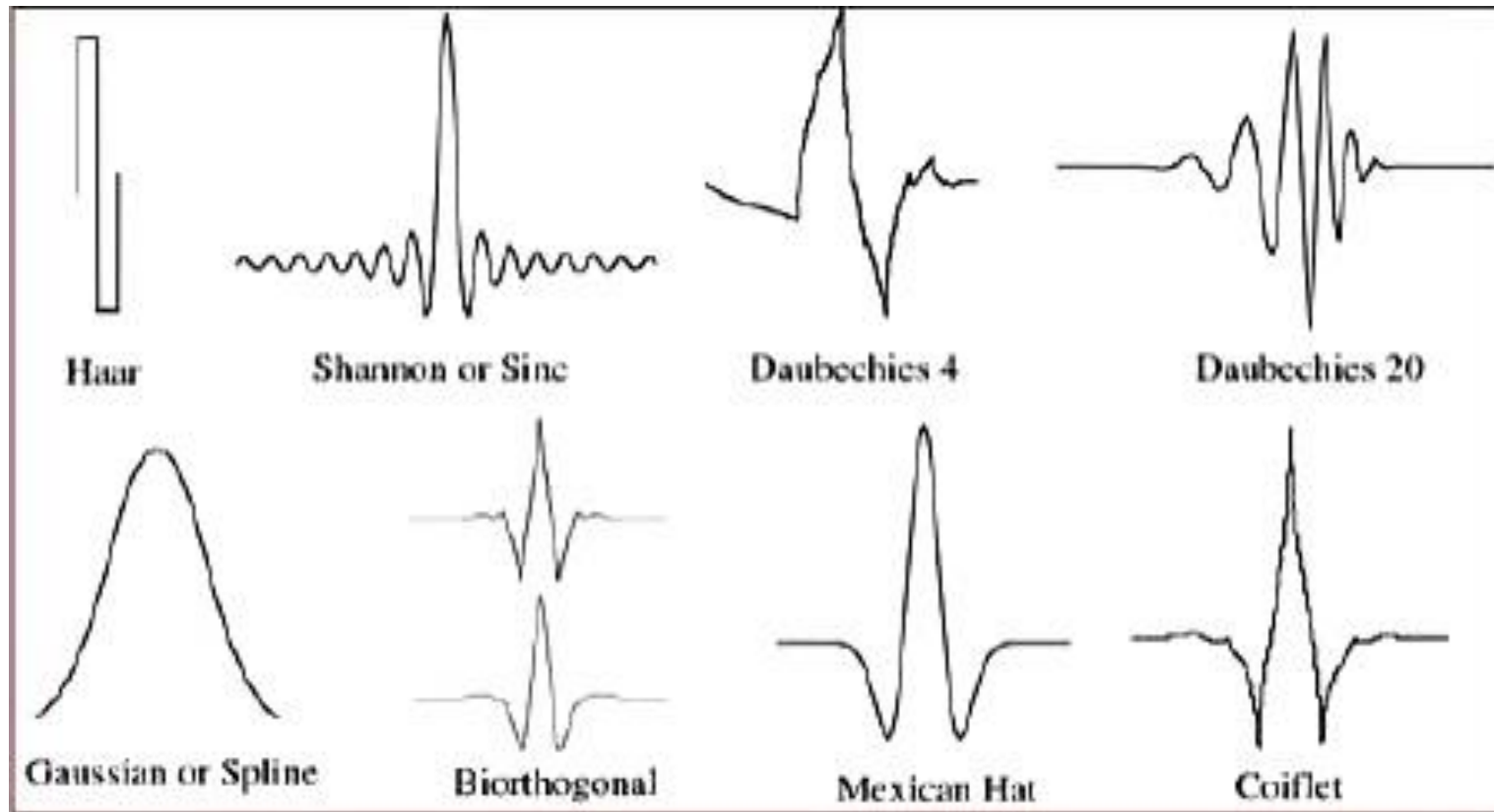
**Non Linear**



- MSE: 0.00  
- PSNR: 73.18  
- SSIM: 0.75



# Wavelet Denoising



# Wavelet Denoising

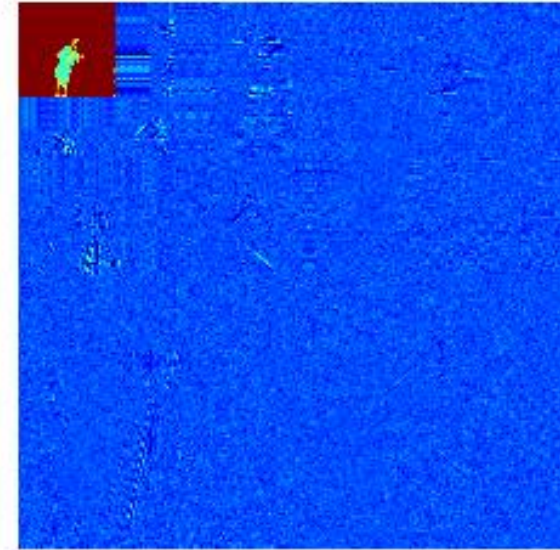
- Universal Threshold

$$\lambda = \sigma * \sqrt{2 * \log n^2}$$

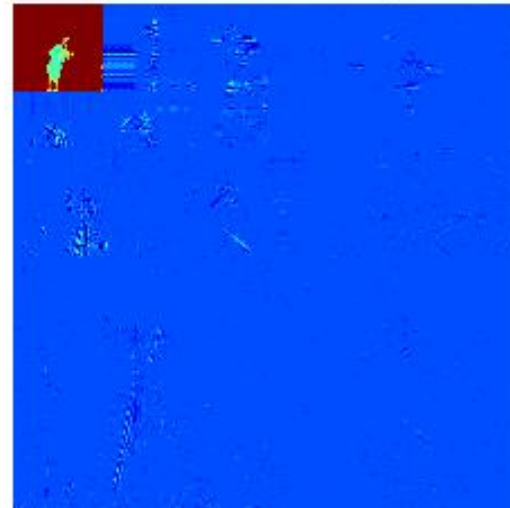
- NeighShrink

$$d_{j,k} = d_{j,k} * \left(1 - \frac{\lambda^2}{\sum_{(i,j) \in B_{j,k}} d_{i,l}^2}\right)$$

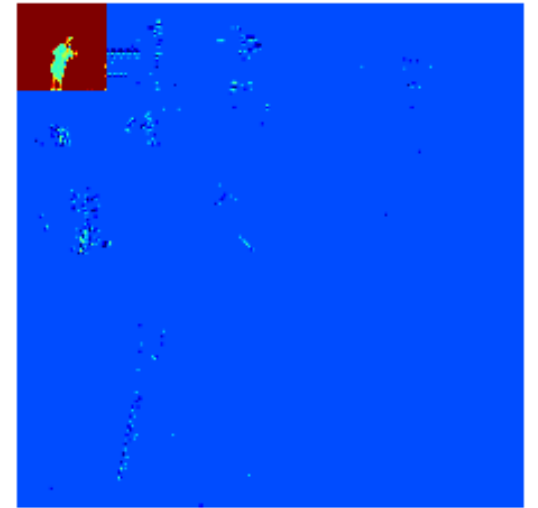
Pre Denoising



Universal Threshold

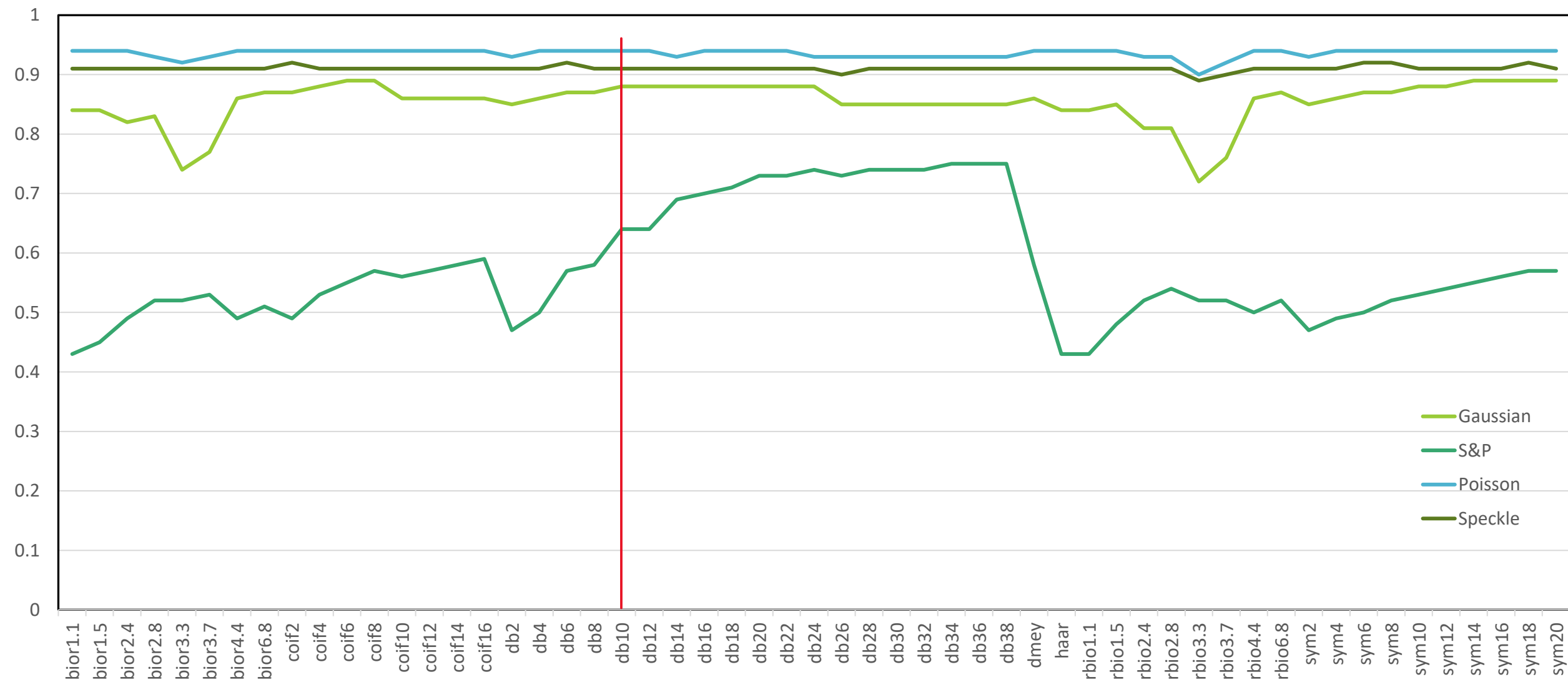


NeighShrink Threshold





Comparing Wavelets Performance on different types of noise



# Wavelet Denoising

**Originale**



**Rumore gaussian**



**univ**



- MSE: 0.01  
- PSNR: 67.62  
- SSIM: 0.68

**Neigh**



- MSE: 0.00  
- PSNR: 72.55  
- SSIM: 0.79

**Originale**



**Rumore gaussian**



**univ**



- MSE: 0.00  
- PSNR: 72.57  
- SSIM: 0.85

**Neigh**



- MSE: 0.00  
- PSNR: 77.52  
- SSIM: 0.88

**Original**



**Added Noise: gaussian**



**Linear**



- MSE: 0.01  
- PSNR: 69.05  
- SSIM: 0.54

**Non Linear**



- MSE: 0.00  
- PSNR: 73.74  
- SSIM: 0.84

**Transformed**



- MSE: 0.00  
- PSNR: 72.62  
- SSIM: 0.79

**Original**



**Added Noise: gaussian**



**Linear**



- MSE: 0.01  
- PSNR: 69.18  
- SSIM: 0.50

**Non Linear**



- MSE: 0.00  
- PSNR: 77.24  
- SSIM: 0.89

**Transformed**



- MSE: 0.00  
- PSNR: 76.96  
- SSIM: 0.88

# Filters comparison – Gaussian

**Original**



**Added Noise: s&p**



**Linear**



- MSE: 0.01  
- PSNR: 66.39  
- SSIM: 0.49

**Non Linear**



- MSE: 0.01  
- PSNR: 67.77  
- SSIM: 0.59

**Transformed**



- MSE: 0.01  
- PSNR: 67.88  
- SSIM: 0.65

**Original**



**Added Noise: s&p**



**Linear**



- MSE: 0.02  
- PSNR: 65.72  
- SSIM: 0.42

**Non Linear**



- MSE: 0.01  
- PSNR: 66.58  
- SSIM: 0.49

**Transformed**



- MSE: 0.01  
- PSNR: 68.63  
- SSIM: 0.64

# Filters comparation – Salt & Pepper



**Original**



**Added Noise: poisson**



**Linear**



- MSE: 0.00  
- PSNR: 74.33  
- SSIM: 0.78

**Non Linear**



- MSE: 0.00  
- PSNR: 76.39  
- SSIM: 0.90

**Transformed**



- MSE: 0.00  
- PSNR: 77.13  
- SSIM: 0.88

**Original**



**Added Noise: poisson**



**Linear**



- MSE: 0.00  
- PSNR: 72.54  
- SSIM: 0.81

**Non Linear**



- MSE: 0.00  
- PSNR: 81.85  
- SSIM: 0.97

**Transformed**



- MSE: 0.00  
- PSNR: 80.15  
- SSIM: 0.94

# Filters comparation – Poisson

**Original**



**Added Noise: speckle**



**Linear**



- MSE: 0.00  
- PSNR: 72.78  
- SSIM: 0.71

**Non Linear**



- MSE: 0.00  
- PSNR: 76.99  
- SSIM: 0.90

**Transformed**



- MSE: 0.00  
- PSNR: 76.95  
- SSIM: 0.87

**Original**



**Added Noise: speckle**



**Linear**



- MSE: 0.00  
- PSNR: 72.15  
- SSIM: 0.81

**Non Linear**



- MSE: 0.00  
- PSNR: 80.59  
- SSIM: 0.96

**Transformed**



- MSE: 0.00  
- PSNR: 77.93  
- SSIM: 0.91

# Filters comparison – Speckle