



Laplacian Operator

Frequency is related to the speed of change of value

- Second derivative based operator

$$h(x, y) = \nabla^2 f(x, y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$H(\mu, \nu) = -4\pi^2(\mu^2 + \nu^2)F(\mu, \nu)$$

(= 0 at $\mu = \nu = 0$, and scales up higher frequencies)

- Second order finite difference

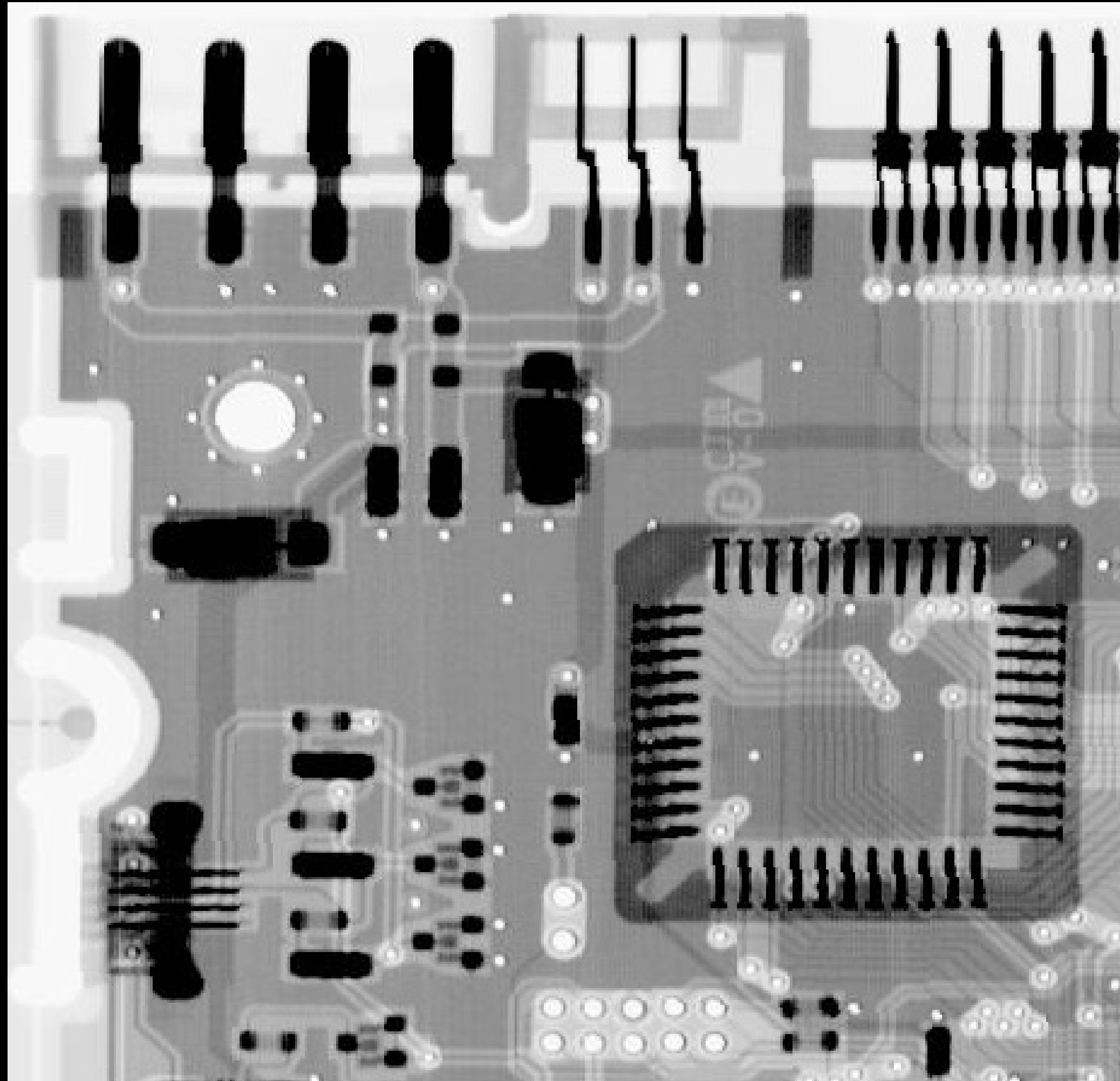
$$\begin{aligned}\nabla^2 f(x, y) &= [f(x+1, y) - 2f(x, y) + f(x-1, y)] + [f(x, y+1) - 2f(x, y) + f(x, y-1)] \\ &= f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)\end{aligned}$$

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

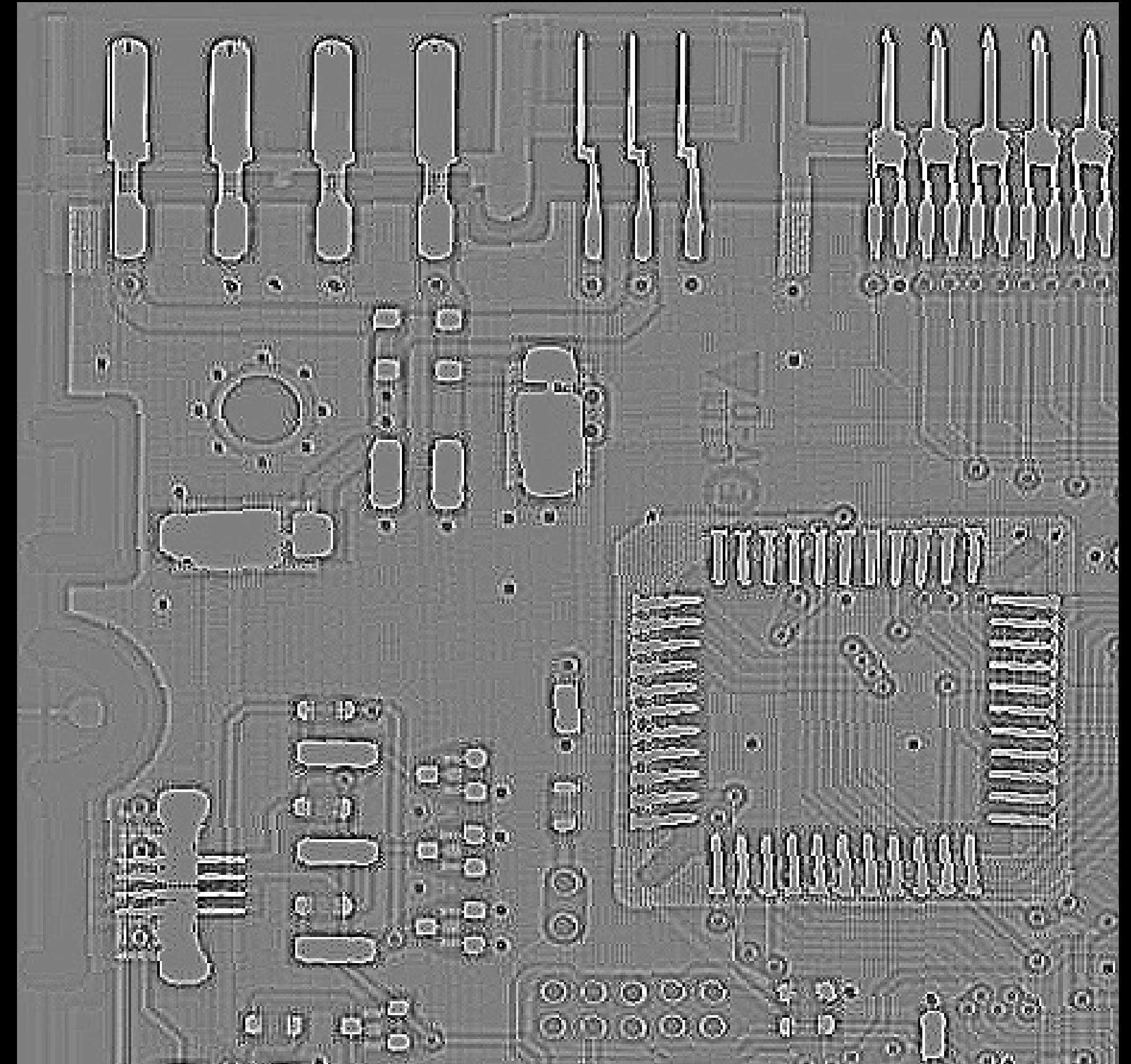
(Note weight sum == 0)

- Effectively an edge-detector

Second Derivatives



Original image f

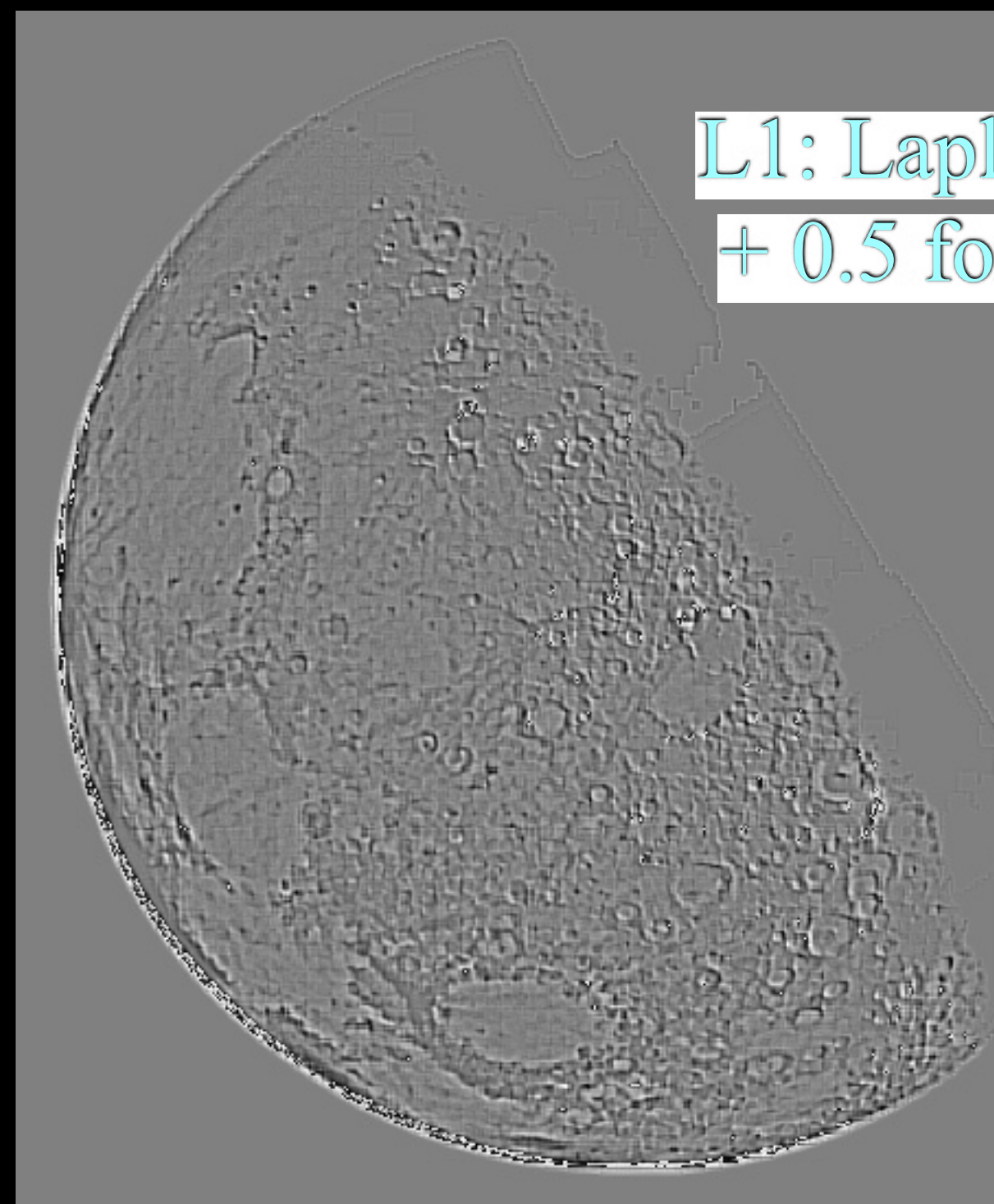


Laplacian $g = \nabla^2 f$ (+ 0.5 for display)

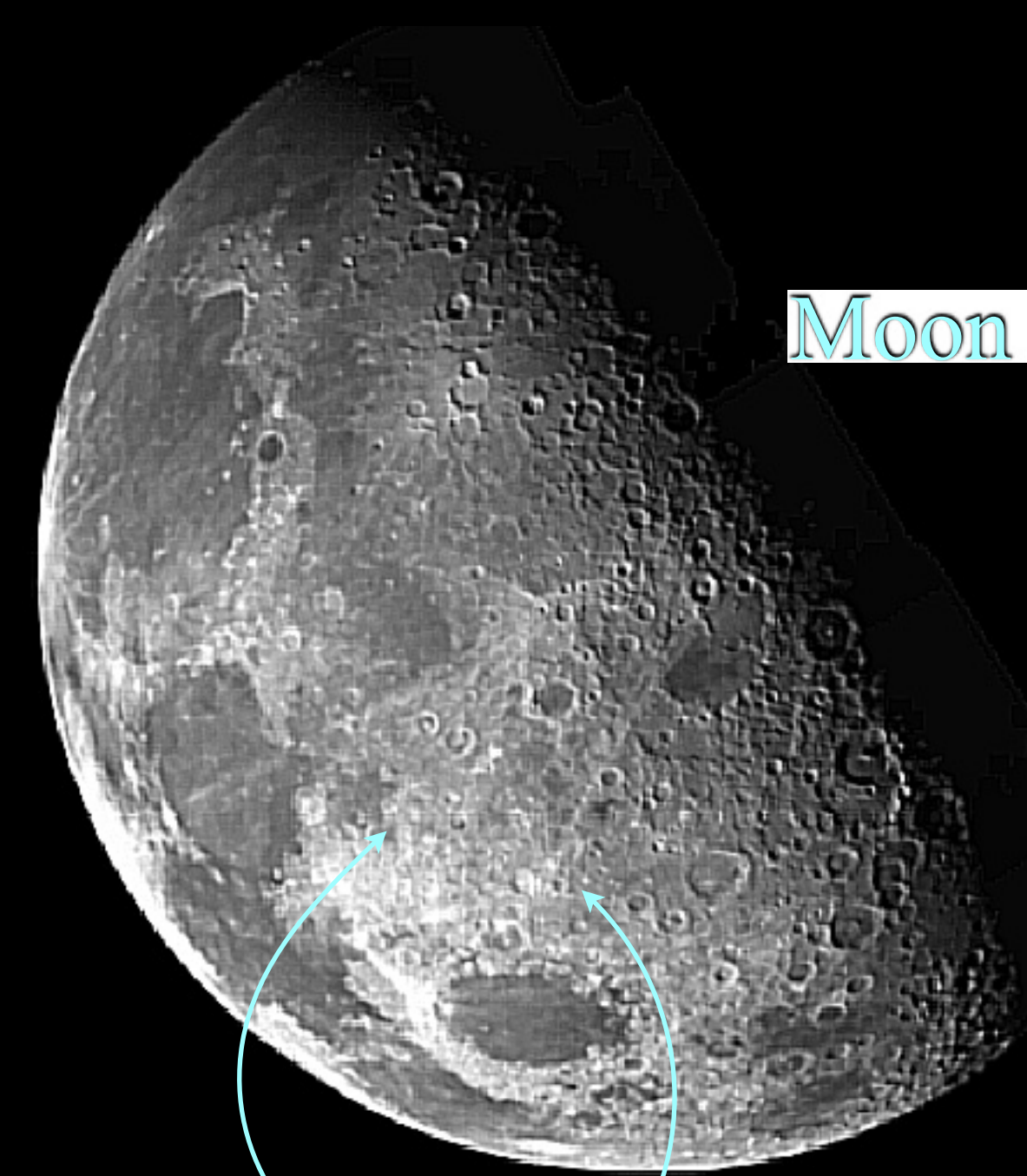
Sharpen



Moon



$L1: \text{Laplacian}(\text{Moon})$
 $+ 0.5 \text{ for display}$



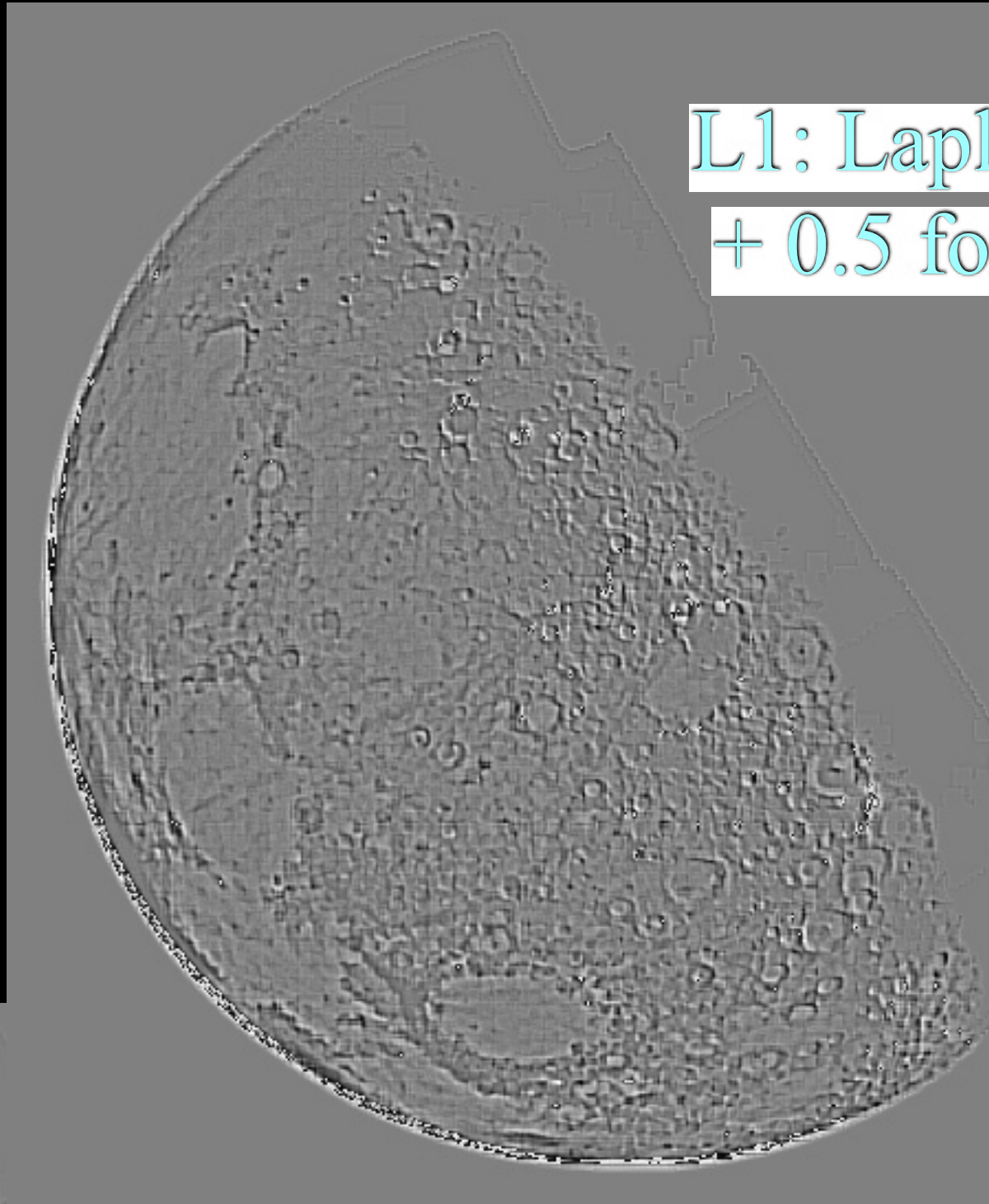
Moon - L1

Noisy

Sharpen



Moon

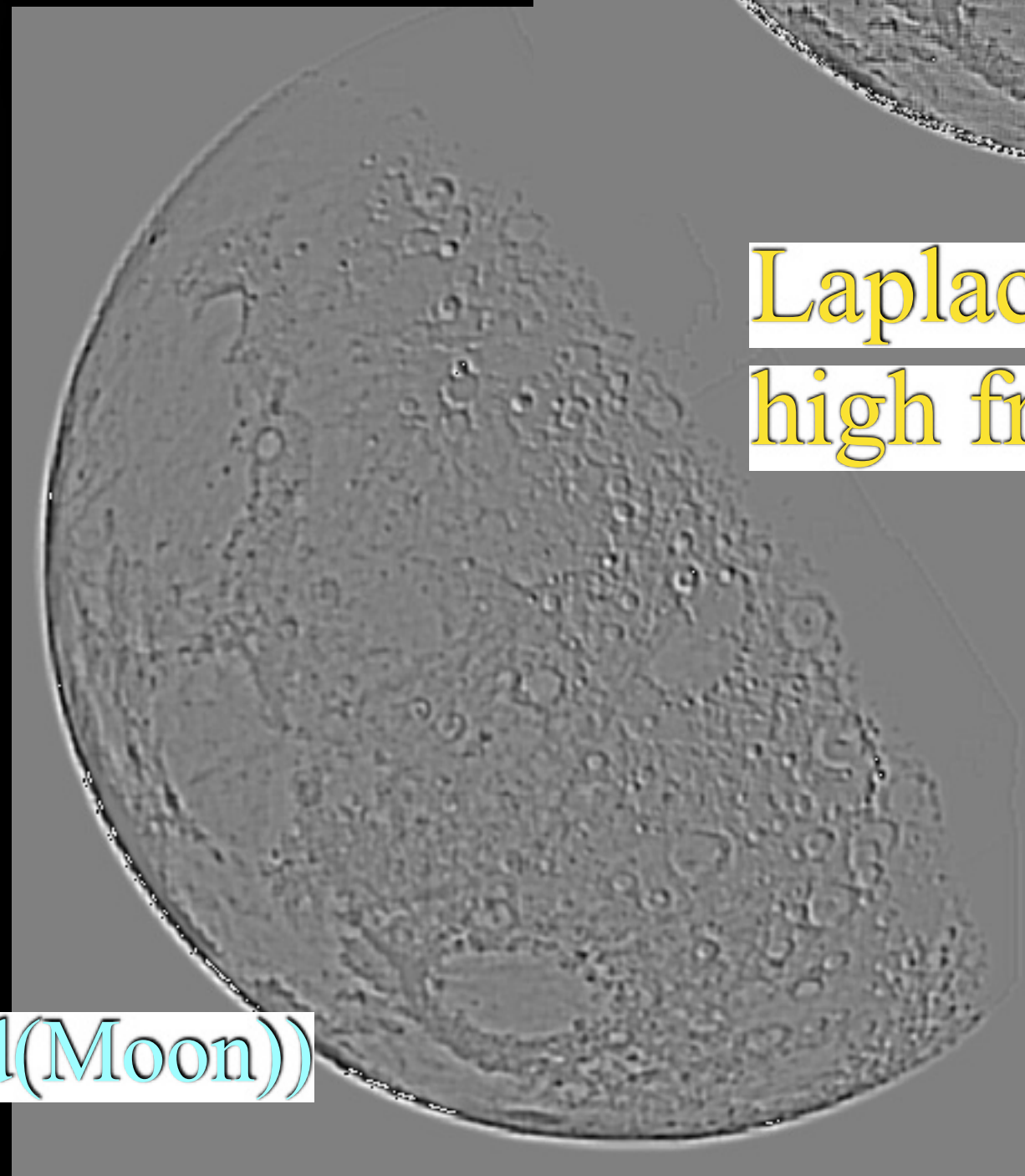


$L1: \text{Laplacian}(\text{Moon})$
 $+ 0.5 \text{ for display}$

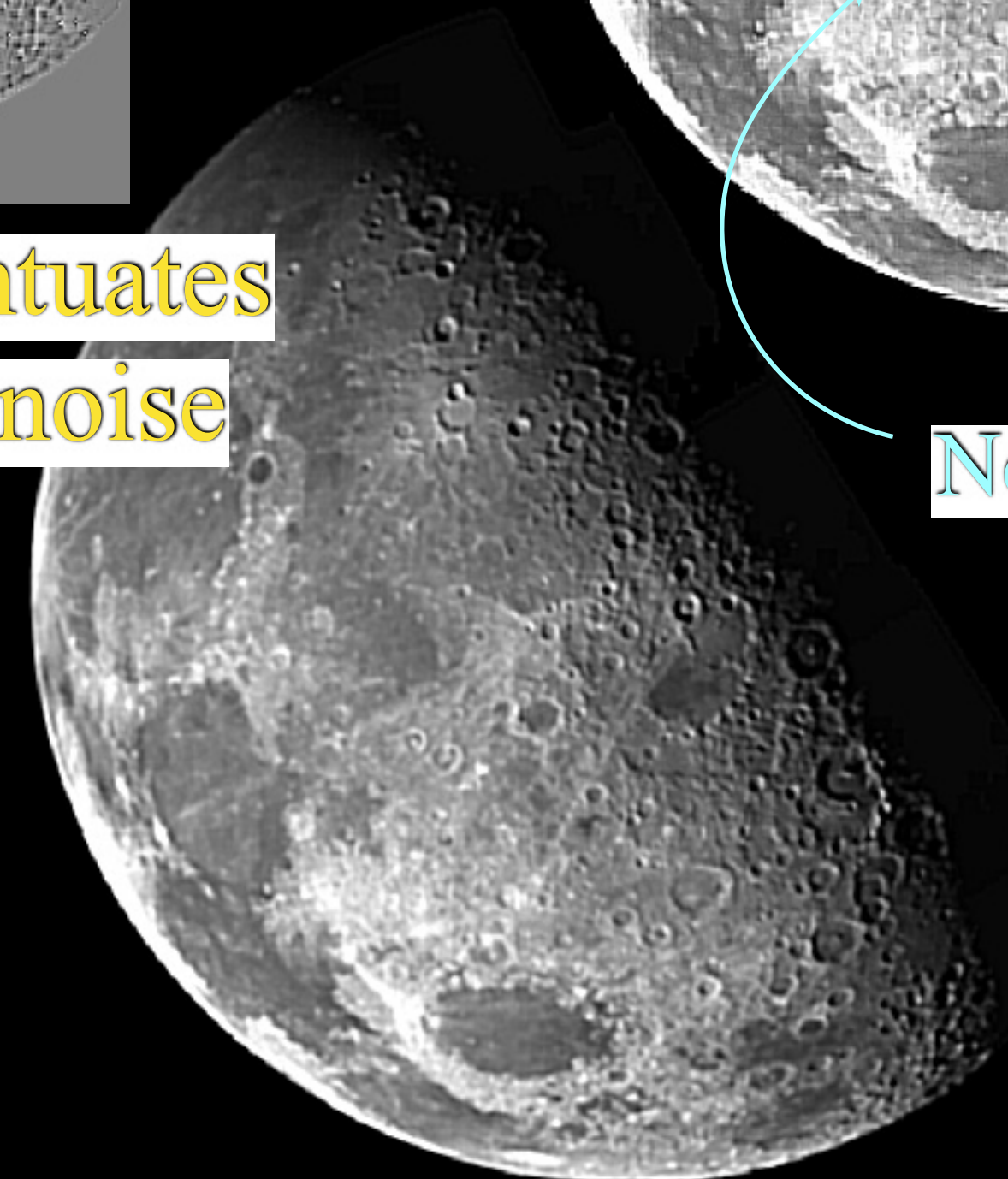


Moon - L1

Noisy



Laplacian accentuates
high frequency noise

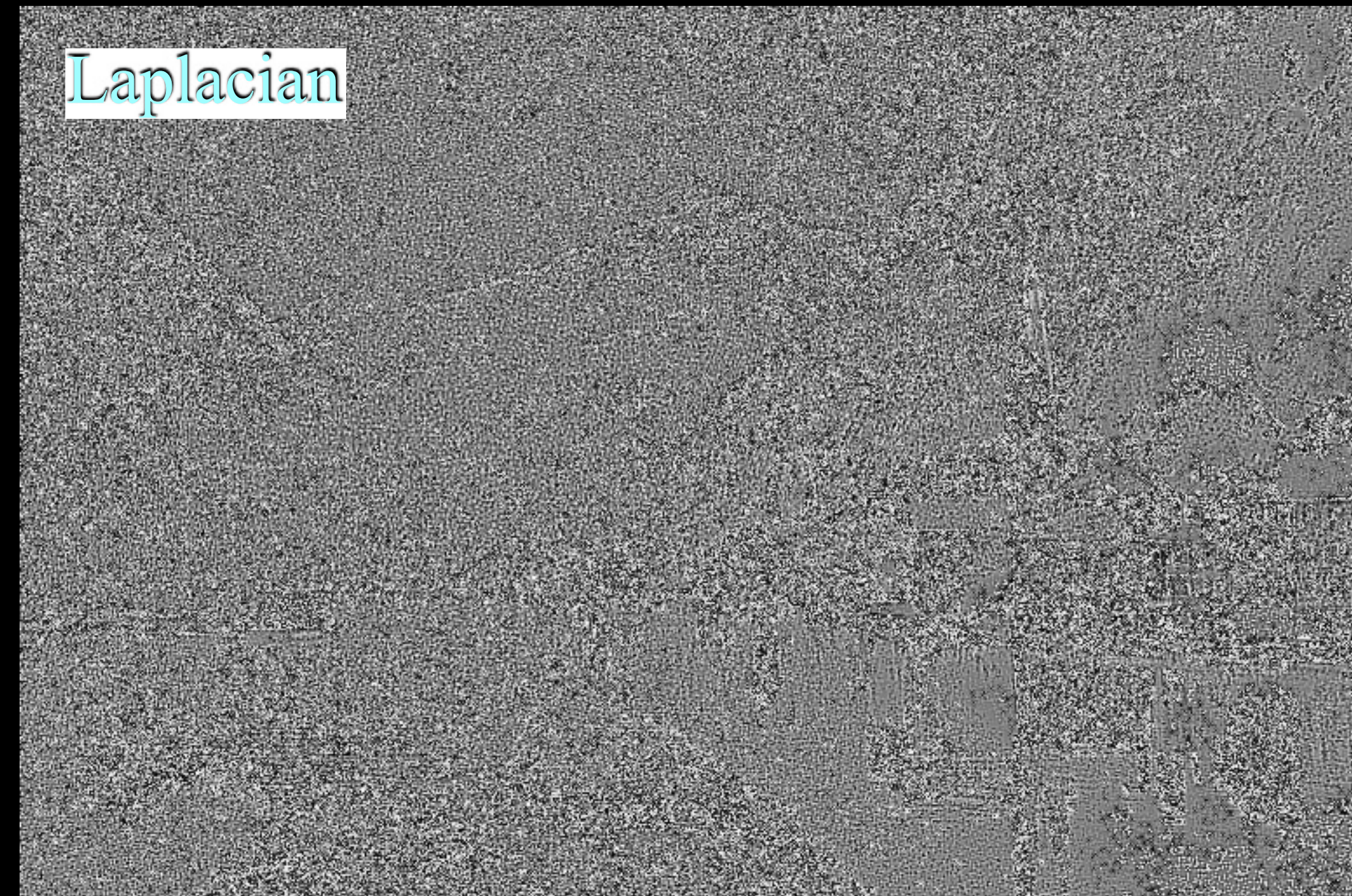


Moon - L2

$L2: \text{Laplacian}(\text{Blurred}(\text{Moon}))$
 $+ 0.5 \text{ for display}$

Noisy Laplacian

Sensitive to noise

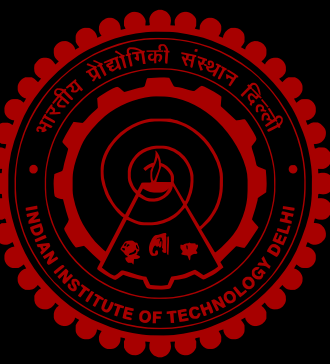




First Derivative

$$\text{Gradient } \nabla f = \begin{bmatrix} \partial_x f \\ \partial_y f \end{bmatrix} = \begin{bmatrix} \partial f / \partial x \\ \partial f / \partial y \end{bmatrix}$$

- Directional derivatives, gradient components are linear but not isotropic
- Magnitude $|\nabla f|$ is isotropic but not linear
- Divided difference: $[f(x+1) - f(x-1)]/2$



Gradient Operators

- Finite difference

$$\nabla_y \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

$$\nabla_x \begin{bmatrix} -1 & 1 \end{bmatrix}$$

- Roberts cross-gradient operators

$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

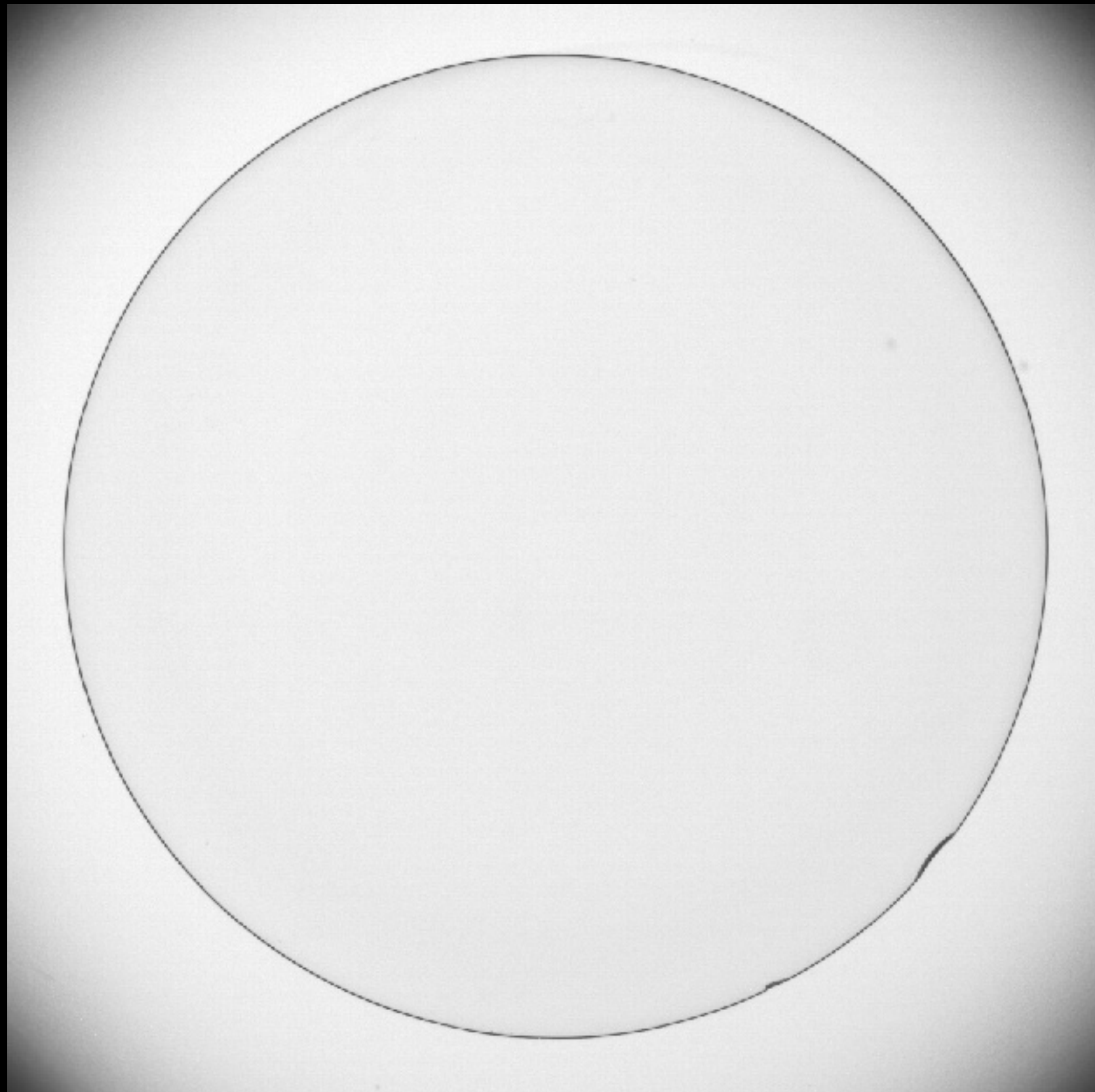
- Sobel operators (Divided difference * Gaussian)

$$\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

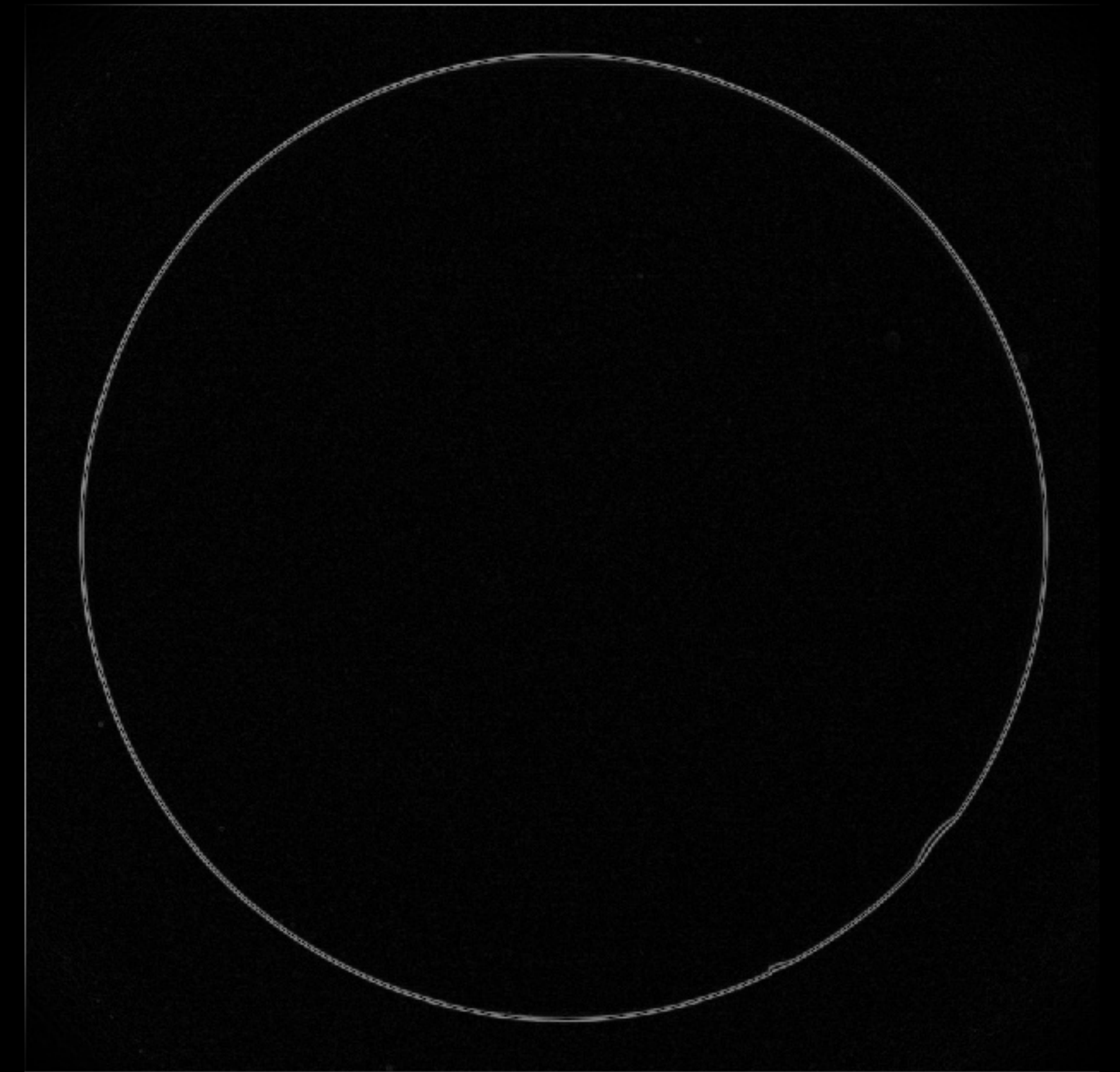
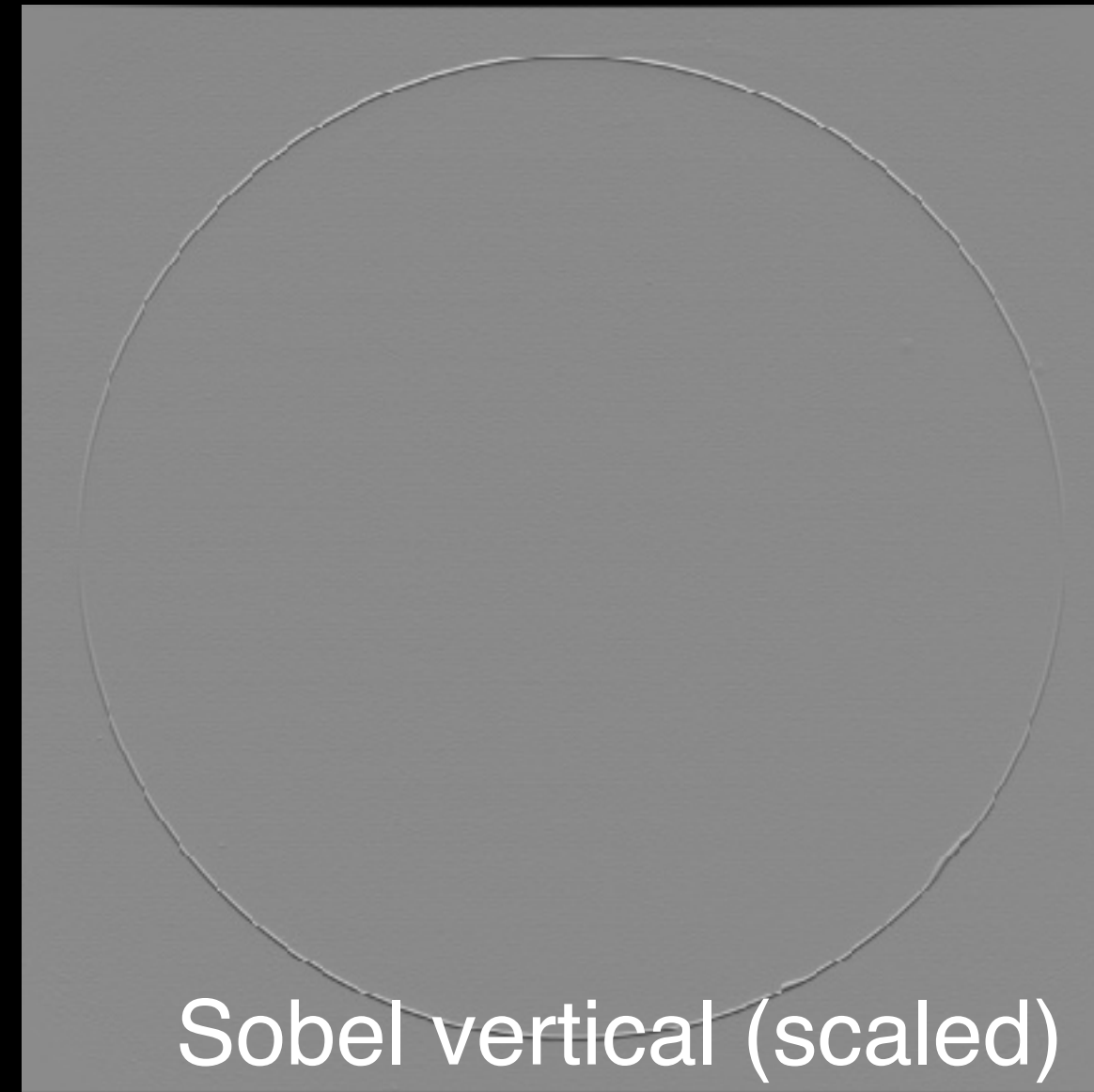
$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

Sobel Operator

Edge detection



Original image

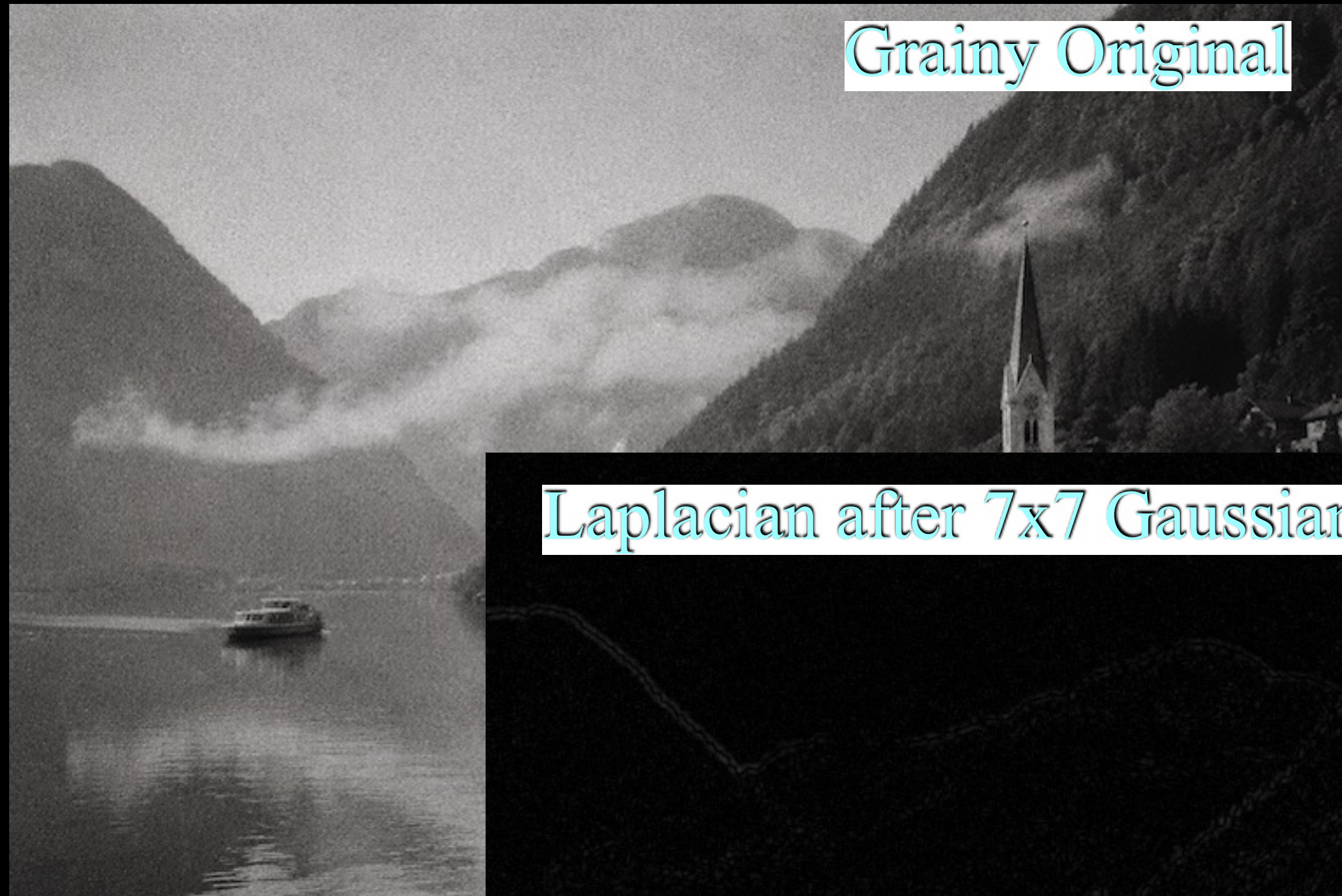
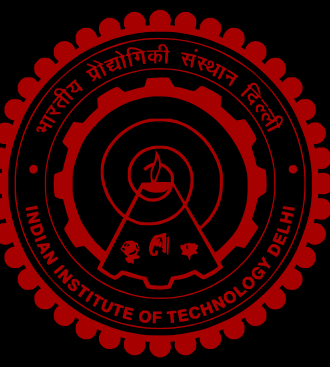


Sobel gradient magnitude

Sobel with Noise



Sobel with Noise



Sobel vs Laplacian

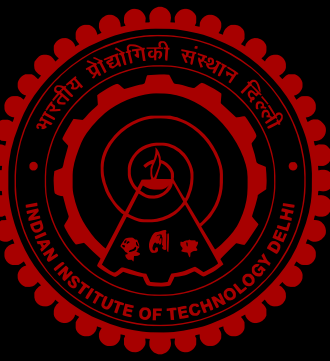


Sobel(barbara)



Laplacian (Barbara)
normalized

- Sobel has more steps, gets thick edges
- Laplacian more susceptible to noise, gets thin edges, good for sharpening and fine detail enhancement



Non-linear Filters

Not weighted average

- Max
- Min
- Median
- Morphological composition
- Bilateral
- Diffusion
- Anisotropic diffusion

Order-based Non-linear Filters

Take k^{th} highest among neighbors

- Median: Noise removal
- Min: Erosion
- Max: Dilation



Original



Median



Minimum



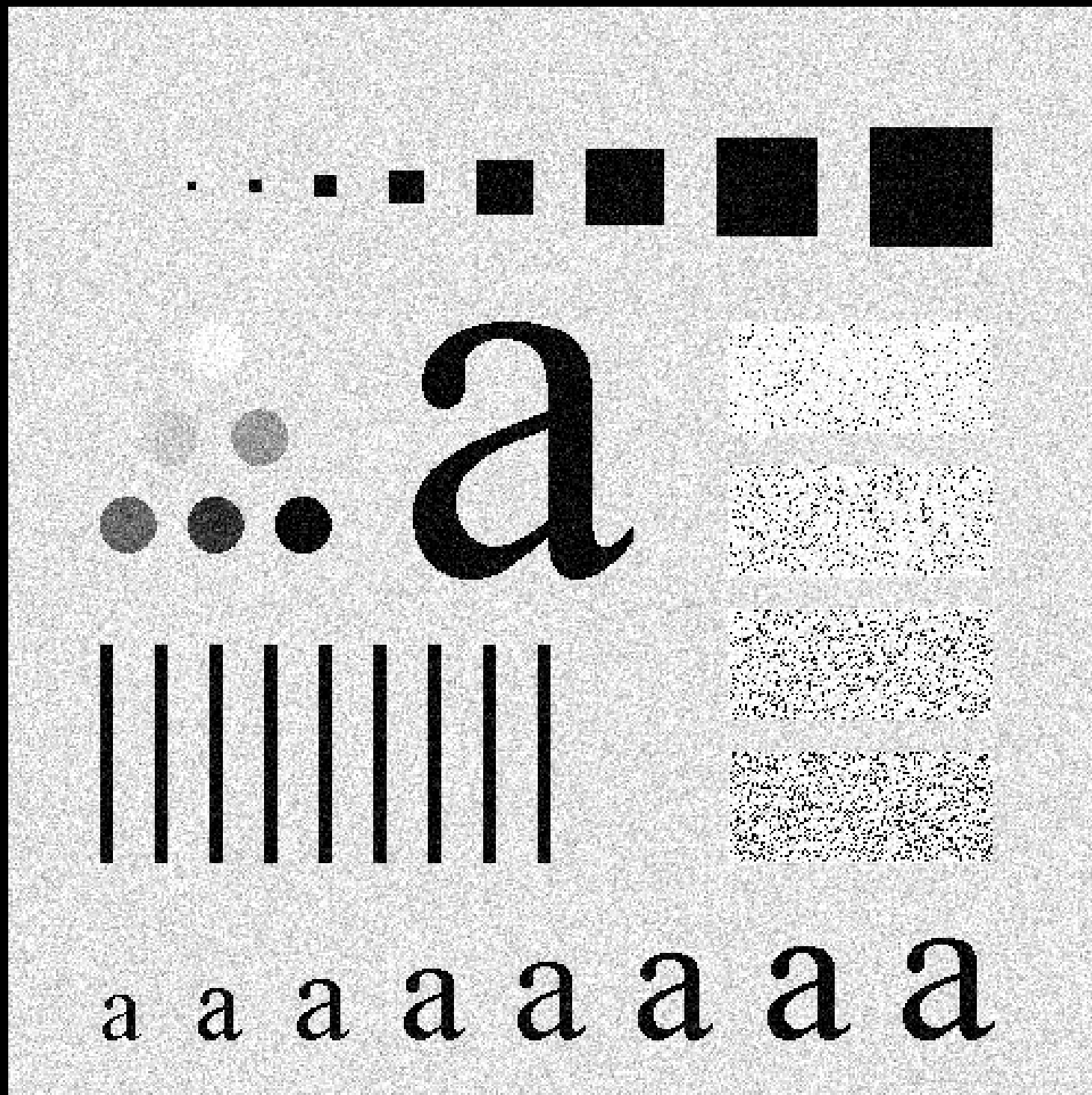
Maximum

Median Filter

Preserves edges, rounds corners

- Replace with median value in an $m \times n$ neighborhood (sort)
 - Removes extreme values, does not smear them by averaging

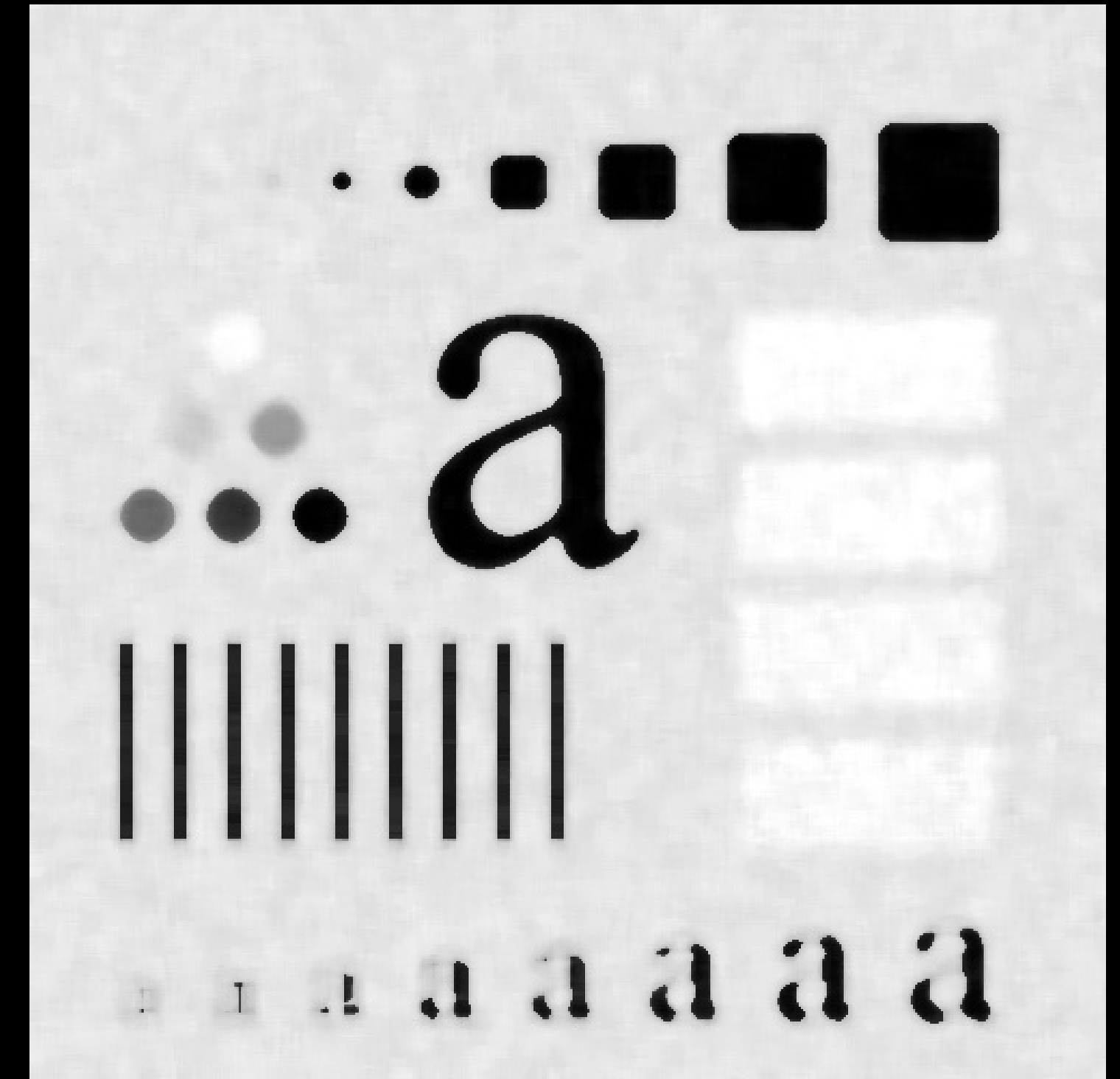
Small features can vanish



Original image



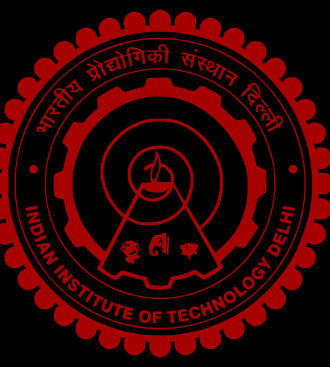
Gaussian filtered



Median filtered

Median Filter

Remove Salt-and-Pepper noise



Gaussian filter with $\sigma=1$



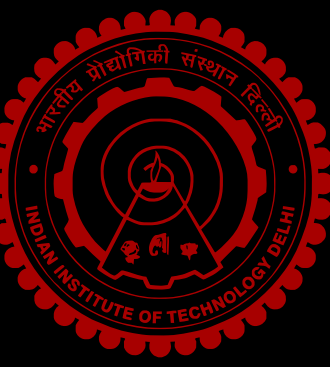
With Salt-and-pepper Noise



5x5 median filter

Median Filter

Remove Salt-and-Pepper noise



Gaussian filter with $\sigma=1$



With Salt-and-pepper Noise



5x5 median filter



5x5 median filter, with pre-smoothing

Example: removing “salt-and-pepper” noise using median filter



With Salt-and-pepper Noise



Gaussian filter with $\sigma = 2$

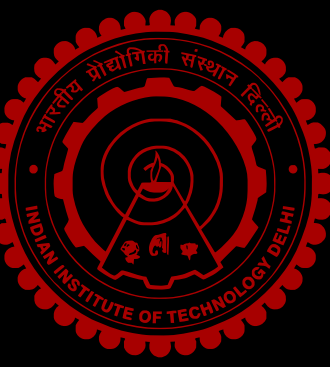


3x3 median filter



Min, Max Filters

- Min removes bright noise (salt)
 - Max removes dark noise (pepper)
- Min erodes (shrinks inward) bright features, separating such features
 - Dilates dark areas, filling gaps
- Max dilates (expands) bright features
 - Erodes dark features
- Subtracting the eroded mask from the original can highlight boundaries



Erosion and Dilation

PRESERVE
THIS TEXT

Original

PRESERVE
THIS TEXT

Erode: Min, 5x5

PRESERVE
THIS TEXT

Dilate back: Min, 5x5, Max 5x5

Erosion and Dilation



1400x1100 Original image

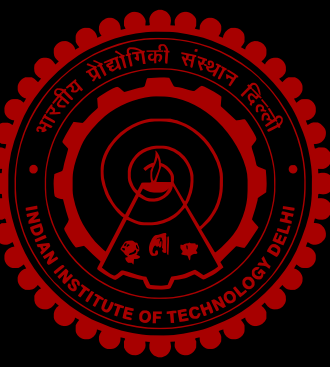


5x5 min filter



5x5 min filter, 5x5 max filter

Erosion and Dilation



21x21 Gaussian Blur



1400x1100 Original image



5x5 min filter



5x5 min filter, 5x5 max filter



Adaptive Median Filtering

Adaptive window size for denoising

- How to choose window size?
 - Trade-off between smoothing and noise reduction
- For highly noisy data, the median of neighbors could be noise (particularly for small window)
- Increase the size of neighborhood until the median is “not noise” and retry
- Estimate if the pixel is actual data or noise:
 - *Values equal to (or near) the extreme values in the neighborhood could be noise*
- Replace the value at a pixel by neighborhood median, only if it itself is noisy

Adaptive Median Filtering

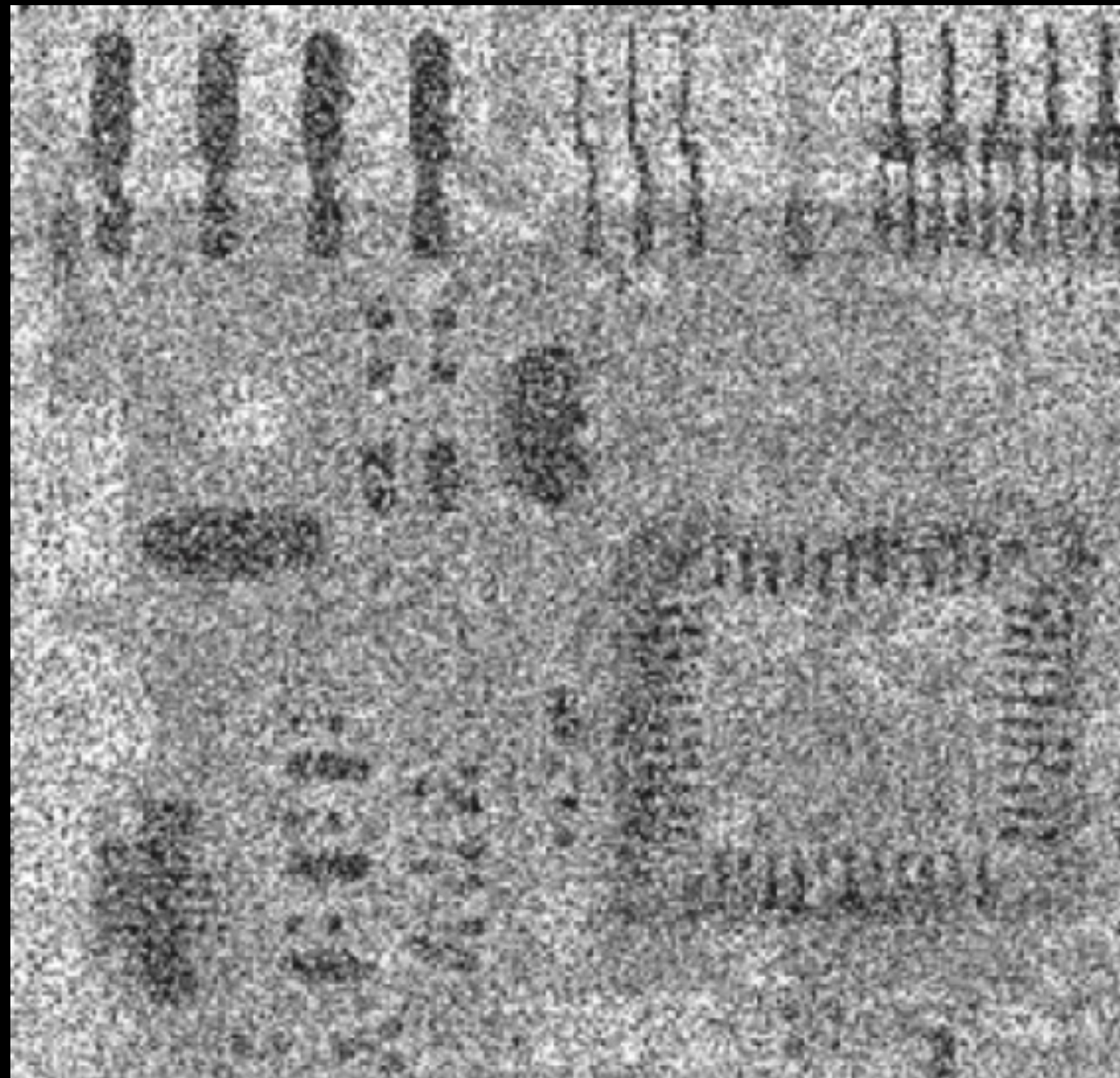
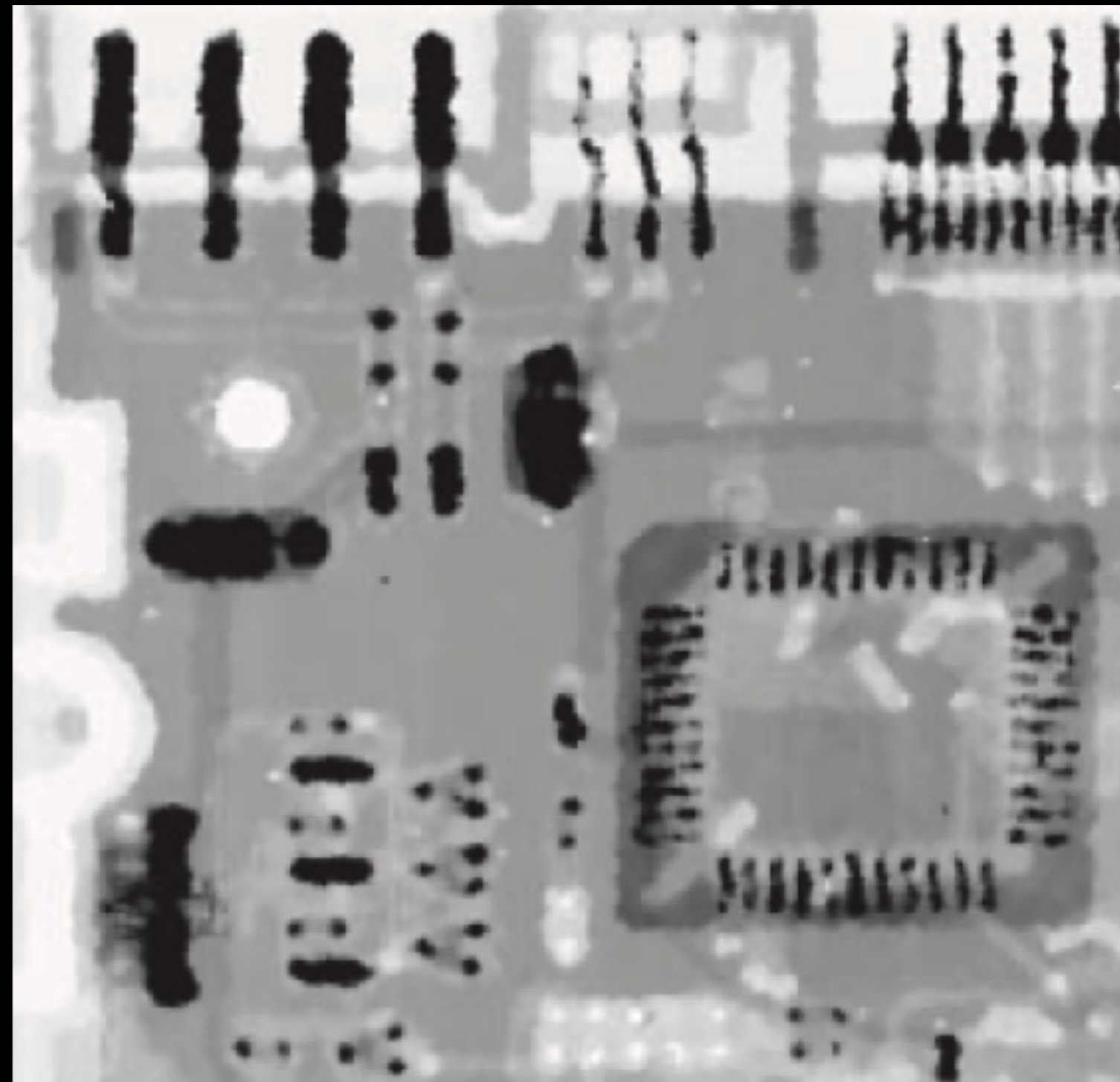
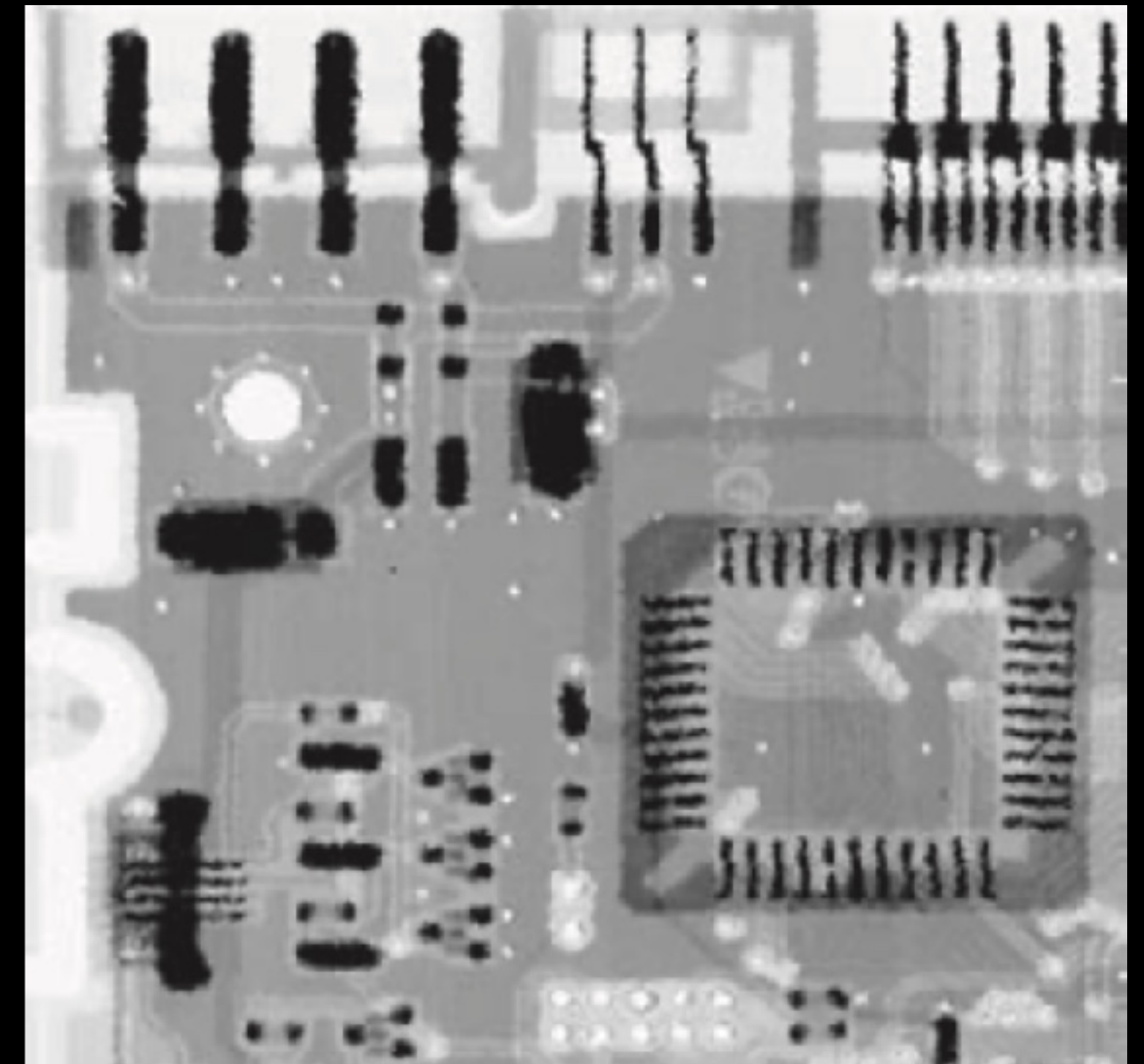


Image with salt-and-pepper noise



7x7 median filter



Adaptive median filter



Adaptive Mean Filtering

Adaptive Smoothing/Denoising

- Instead of increasing window size, adapt the amount of smoothing
 - By blending between the ‘unsmooth’ input and the smoothed values
- Adapt the amount of smoothing to local features
 - Edges (less), corners (less), noise (more), etc.
 - Need to find features, and estimate noise

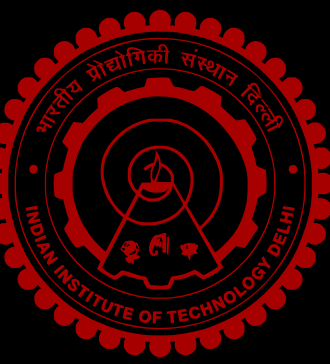
Linear interpolation between c and $\hat{c}$

- **Adaptive Mean Filter:** $\hat{\hat{c}}(x, y) = w(x, y) \hat{c}(x, y) + (1 - w(x, y)) c(x, y)$, where

$$w(x, y) = \sigma_{\eta}^2 / \text{var}_c(x, y)$$

(variance of noise
relative to that of data)

- “High frequency” noise estimation: high-pass filter



Noise Estimator

Immerkær Laplacian Mask

- Convolve c with $\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$
 - Lower spike at edges compared to Laplacian and Sobel filters
 - Yields 0 even at linear gradients and many edges
- Taking the inner $(H-2) \times (W-2)$ pixels of $\hat{c}$ (for an $H \times W$ input image):

Product of 1D 2nd derivative mask:

$$\begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} [1 \quad -2 \quad 1]$$

$$\sigma_{\eta} = \frac{\sum \hat{c}}{6(W-2)(H-2)} \sqrt{\pi/2}$$

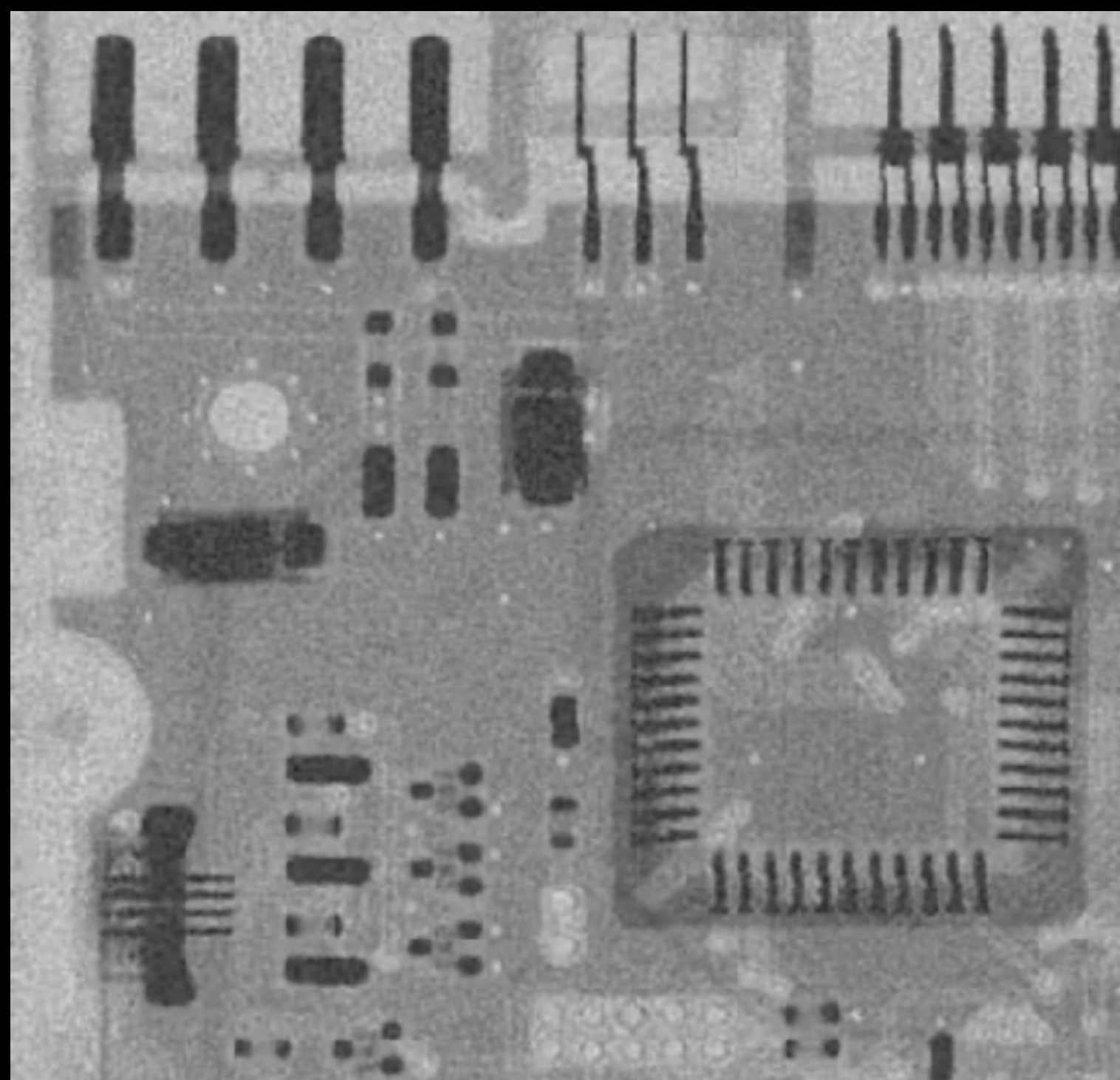
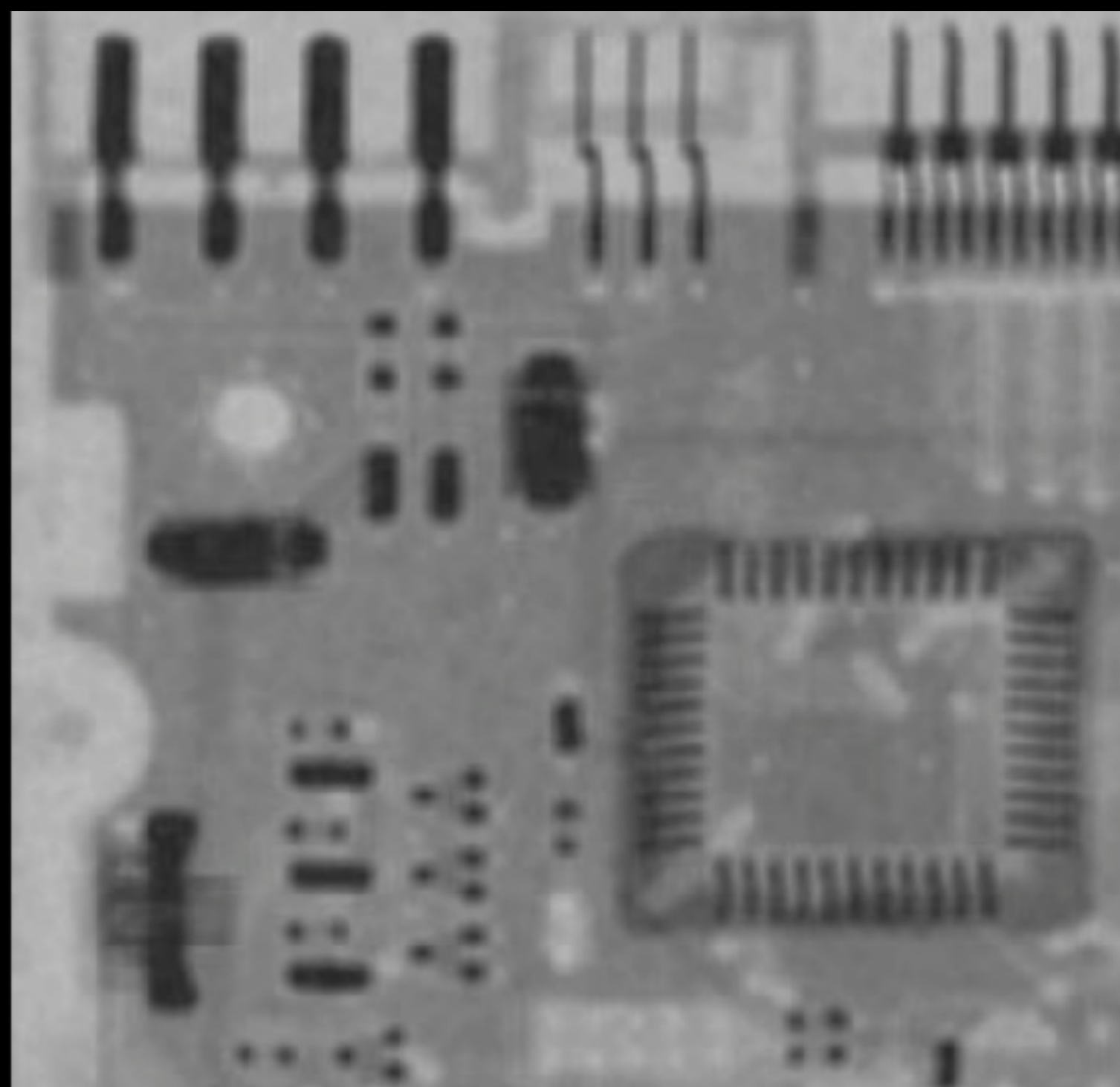
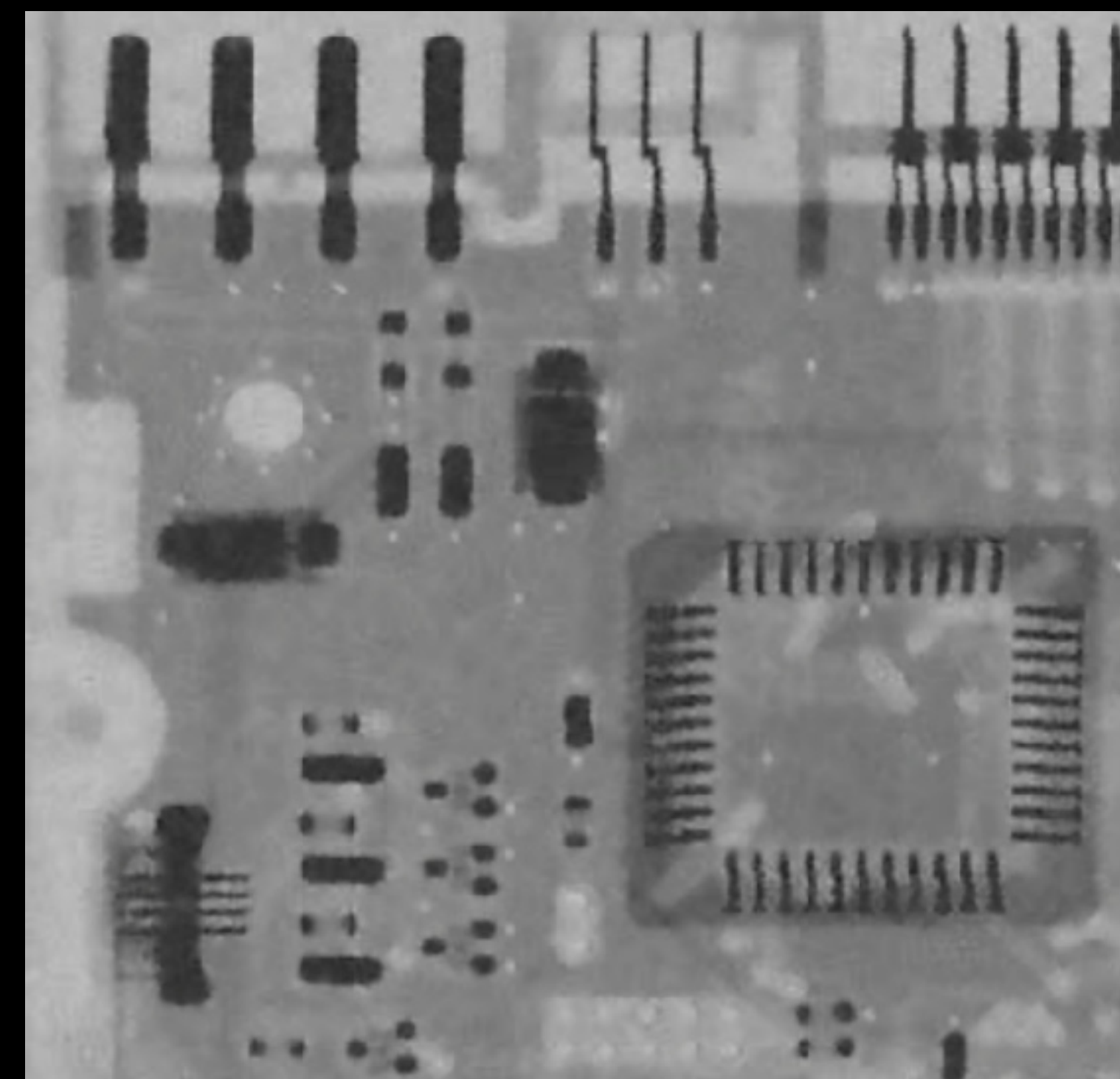


Image with additive Gaussian noise



Arithmetic mean filtering



Adaptive mean filtering