# Image Enhancement (Spatial Filtering)

# Image Enhancement Revisited

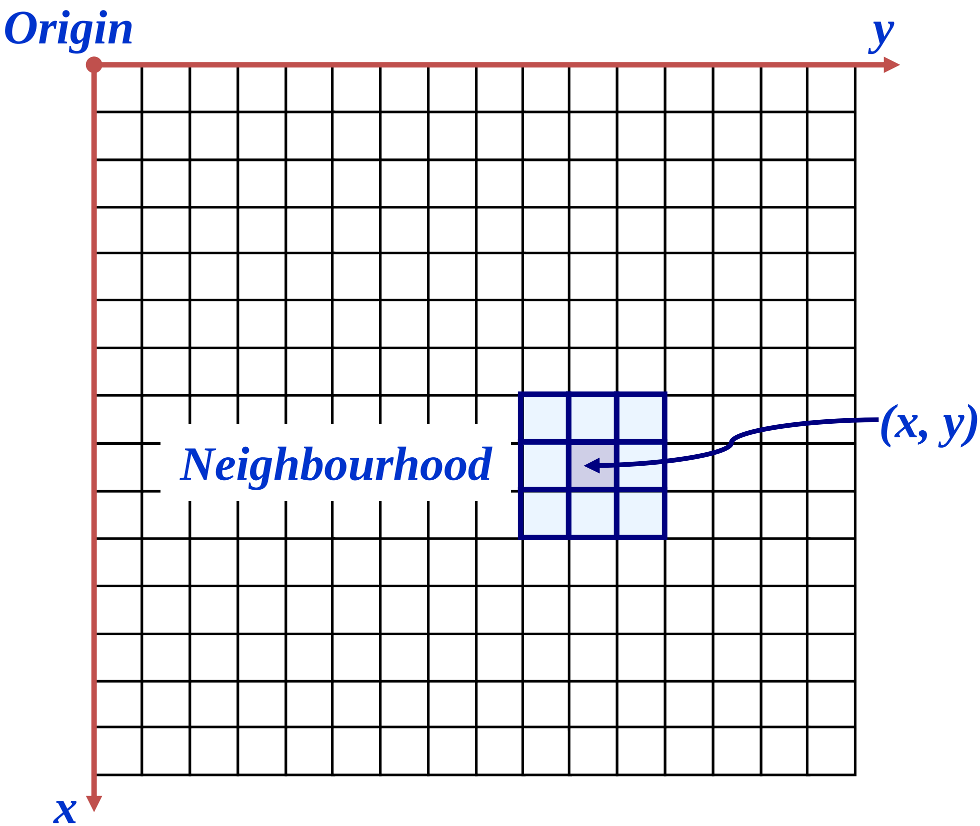
- Spatial domain methods:

  Operate directly on pixels
- Frequency domain methods:

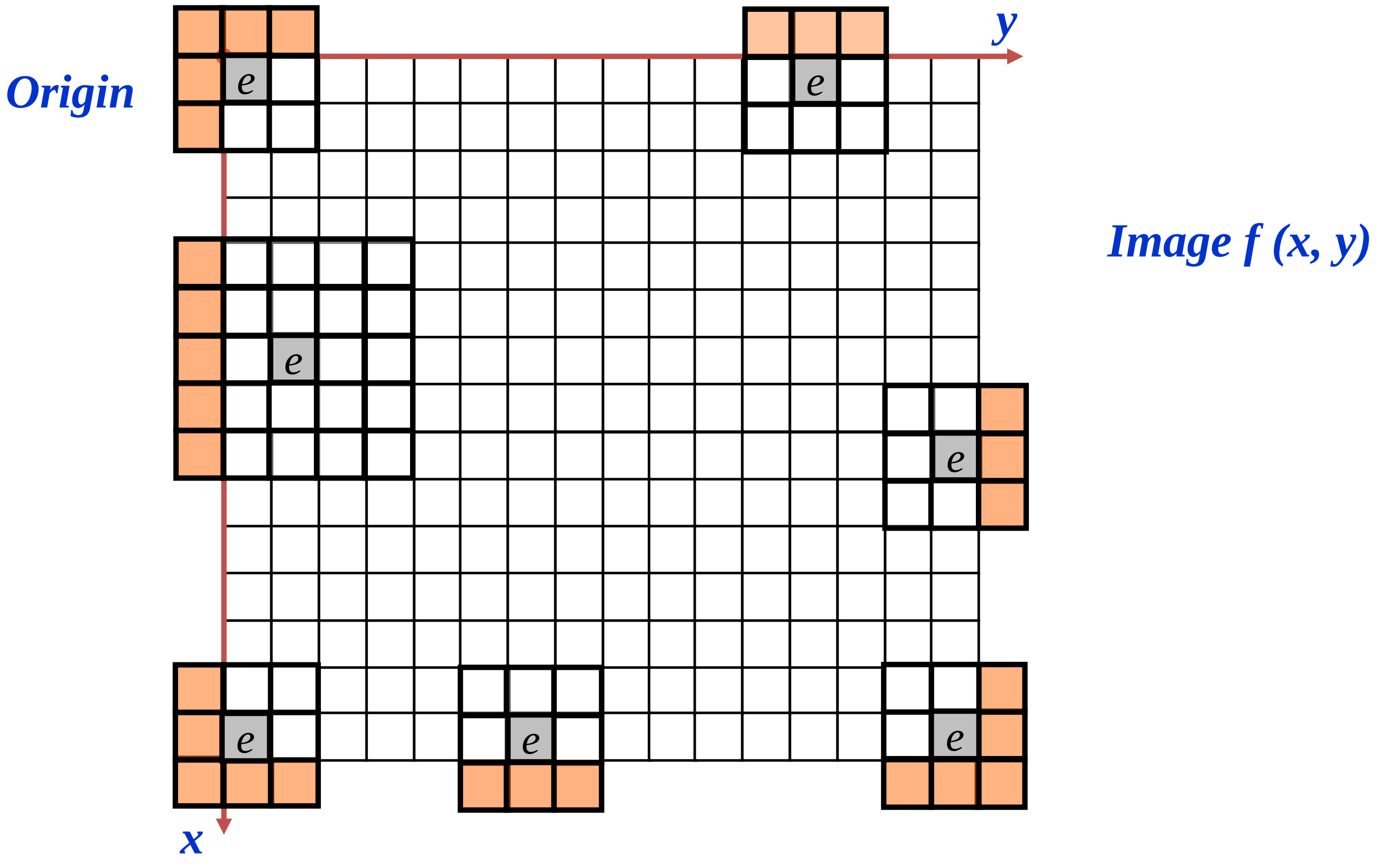
  Operate on the transform of an image

# Neighbourhood Operations

- Operate on a neighbourhood of pixels



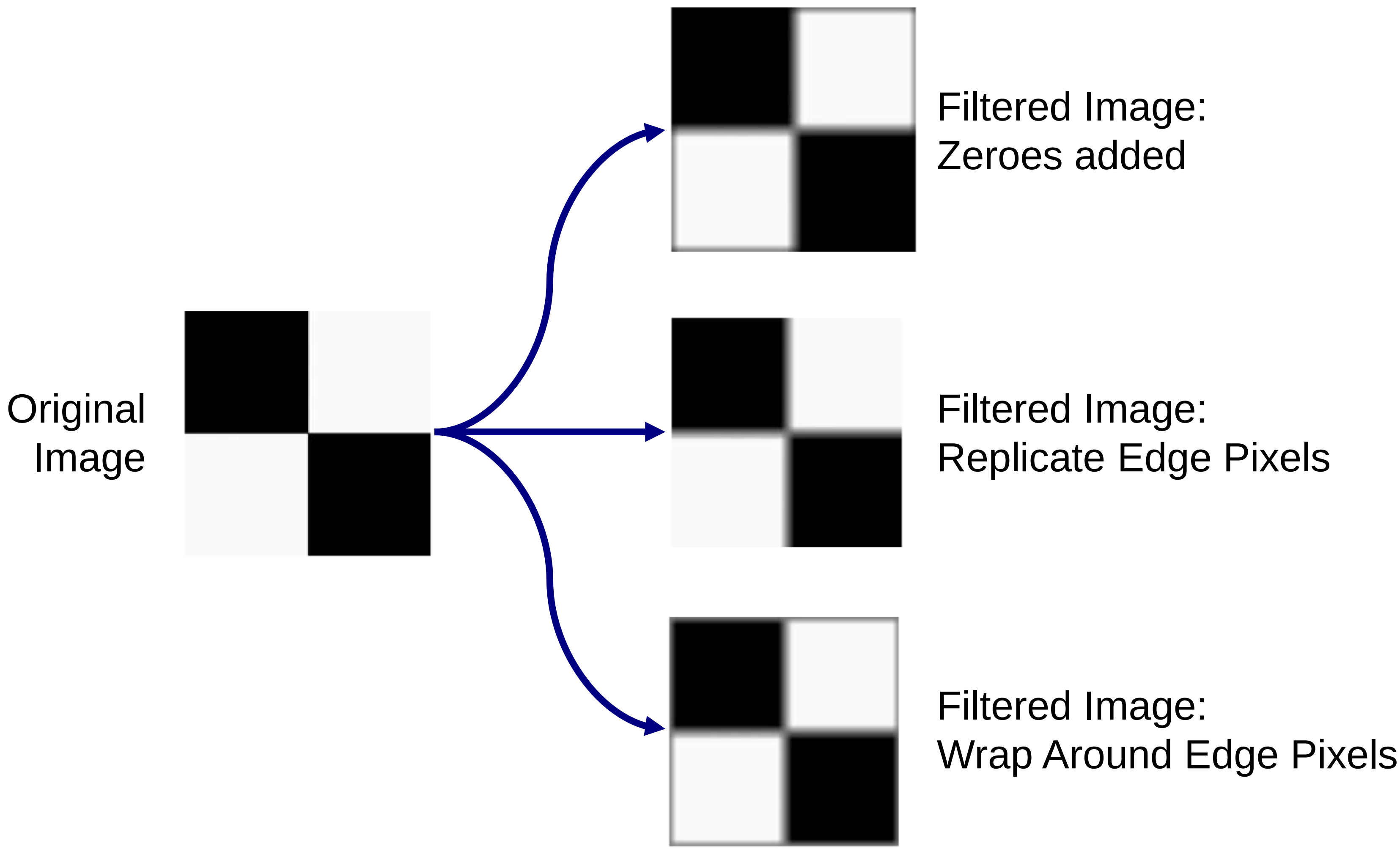
# Processing of pixels at the edges



# Approaches to process pixels at the edges

- Add pixels at corners with either all white or all black pixels
- Replicate border pixels
- Truncate the image
- Allow pixels wrap around the image

# Examples: modified image at the edges



# Neighbourhood Operations Example

***Original Image***

| | | | | | |
|---|---|---|---|---|---|
| 123 | 127 | 128 | 119 | 115 | 130 |
| 140 | 145 | 148 | 153 | 167 | 172 |
| 133 | 154 | 183 | 192 | 194 | 191 |
| 194 | 199 | 207 | 210 | 198 | 195 |
| 164 | 170 | 175 | 162 | 173 | 151 |

***Enhanced Image***

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| | 142 | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

# 2-D Convolution for Enhancement

Image is convolved with filter Convolution

| | | |
|---|---|---|
| *a* | *b* | *c* |
| *d* | *e* | *e* |
| *f* | *g* | *h* |

**Original Image Pixels**

$*$

**Convolution**

| | | |
|---|---|---|
| *r* | *s* | *t* |
| *u* | *v* | *w* |
| *x* | *y* | *z* |

**Filter**

For convolution, swap filter and multiply it with image

| | | |
|---|---|---|
| *z* | *y* | *x* |
| *w* | *v* | *u* |
| *t* | *s* | *r* |

**Swapped Filter**

$$e_{\text{convolution}} = (v \times e) + (z \times a) + (y \times b) + (x \times c) + (w \times d) + (u \times e) + (t \times f) + (s \times g) + (r \times h)$$

| | | |
|---|---|---|
| *a* | *b* | *c* |
| *d* | *e* | *e* |
| *f* | *g* | *h* |

**Original Image Pixels**

$\times$

| | | |
|---|---|---|
| *z* | *y* | *x* |
| *w* | *v* | *u* |
| *t* | *s* | *r* |

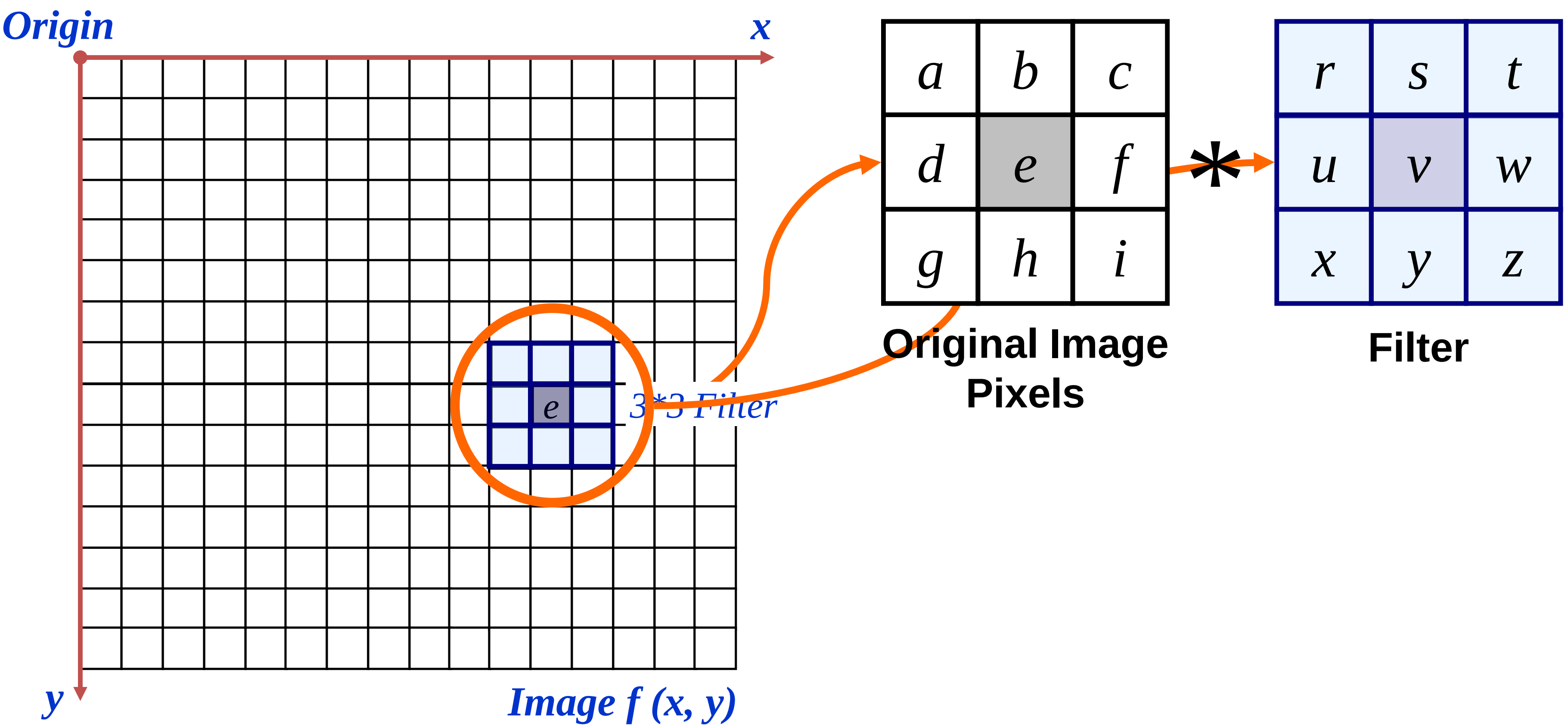
**Swapped Filter**

$=$

| | | |
|---|---|---|
| *a* | *b* | *c* |
| *d* | $e_{con}$ | *e* |
| *f* | *g* | *h* |

**Filtered Image Pixels**

# The Spatial Filtering Process




Procedure is repeated for every pixel in the original image to generate the filtered image

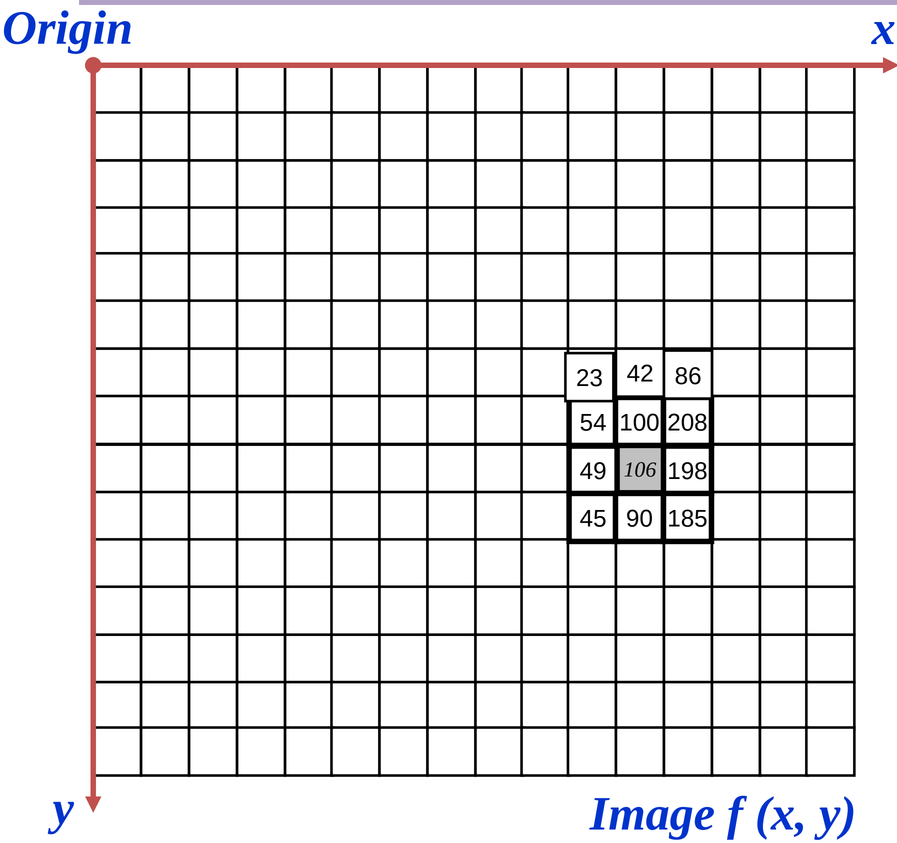
# Smoothing Spatial Filters

- Averages all of the pixels in a neighbourhood around a central value
- Useful in reducing noise from images and for highlighting gross detail

Averaging filter

$$A = (1/9)\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

# Smoothing Spatial Filtering




| | | |
|---|---|---|
| *54* | *100* | *208* |
| *49* | *106* | *198* |
| *45* | *90* | *185* |

**Original Image Pixels**

$*$ 1/9

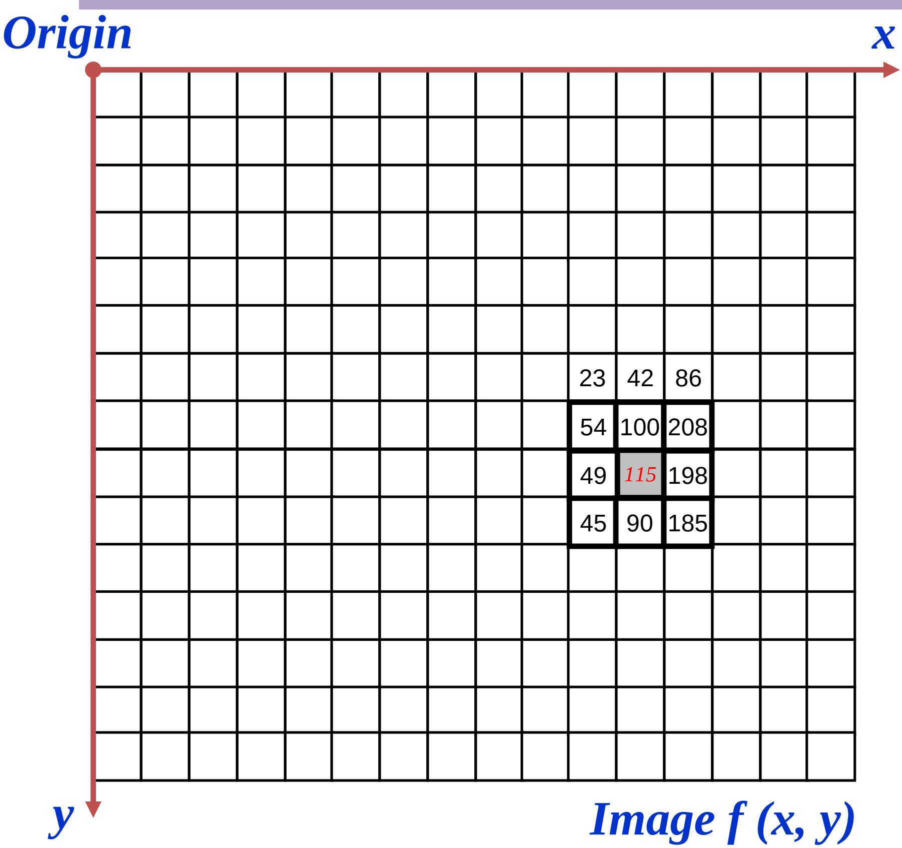
| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

**Filter**

*3*3 Smoothing Filter*

$$e = {}^{1}/_{9}[106 + 54 + 100 + 208 + 49 + 198 + 45 + 90 + 185] = 115$$

# Smoothing Spatial Filtering




| | | |
|---|---|---|
| *54* | *100* | *208* |
| *49* | *106* | *198* |
| *45* | *90* | *185* |

**Original Image Pixels**

$*$ 1/9

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

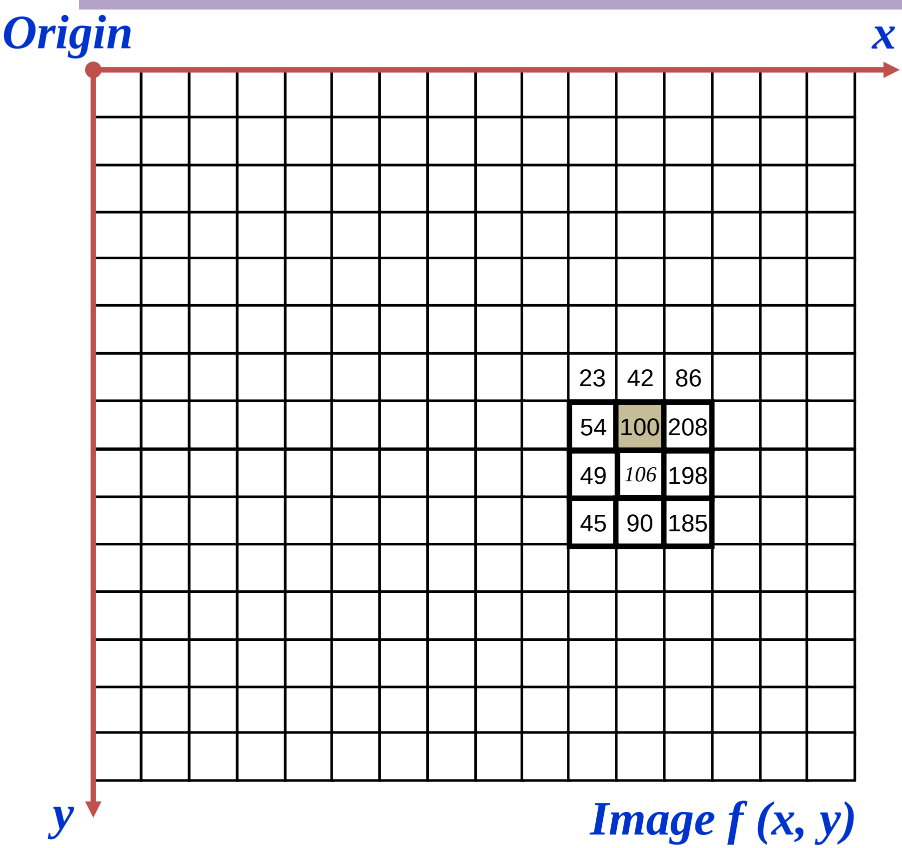
**Filter**

*3*3 Smoothing Filter*

$$e = {}^1/_9[106 + 54 + 100 + 208 + 49 + 198 + 45 + 90 + 185]$$

$$= 115$$

# Smoothing Spatial Filtering

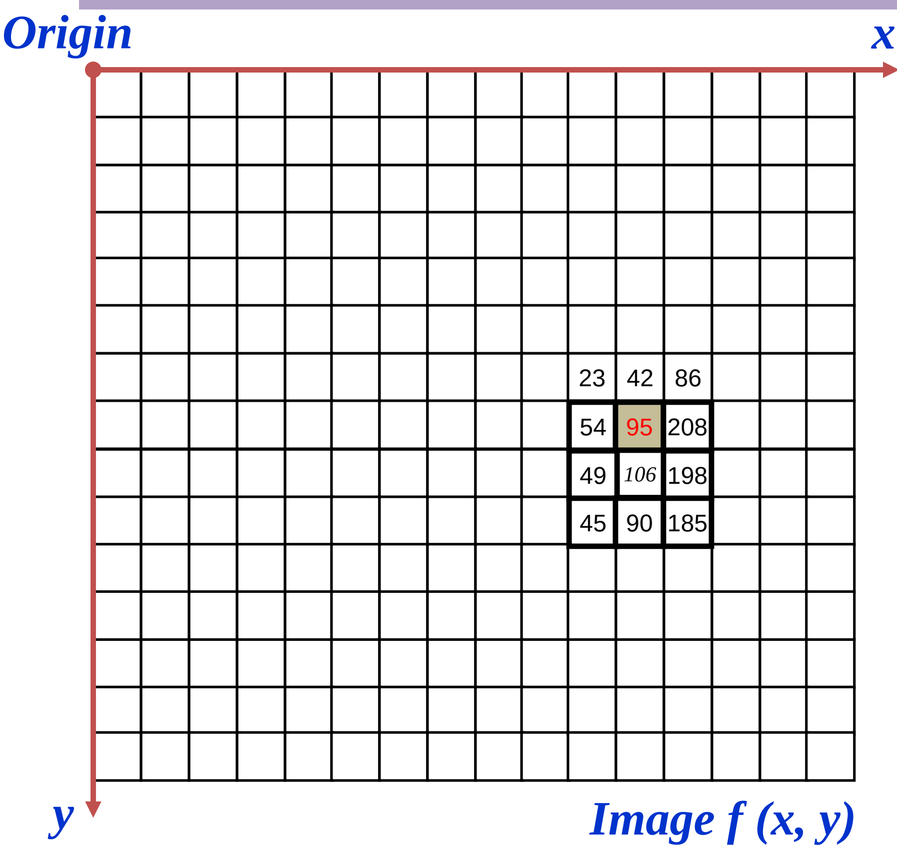



$$\begin{bmatrix} 23 & 42 & 86 \\ 54 & 100 & 208 \\ 49 & 106 & 185 \end{bmatrix} * \begin{bmatrix} 1/9 & {}^{1}/_{9} & {}^{1}/_{9} \\ {}^{1}/_{9} & {}^{1}/_{9} & {}^{1}/_{9} \\ {}^{1}/_{9} & {}^{1}/_{9} & {}^{1}/_{9} \end{bmatrix}$$

*3*3 Smoothing Filter*

$100 \rightarrow 94.77 \rightarrow 95$

# Smoothing Spatial Filtering




$$\begin{bmatrix} 23 & 42 & 86 \\ 54 & 100 & 208 \\ 49 & 106 & 185 \end{bmatrix} * \begin{bmatrix} 1/9 & {}^{1}/_{9} & {}^{1}/_{9} \\ {}^{1}/_{9} & {}^{1}/_{9} & {}^{1}/_{9} \\ {}^{1}/_{9} & {}^{1}/_{9} & {}^{1}/_{9} \end{bmatrix}$$

*3*3 Smoothing Filter*

$$e = 94.77 \rightarrow 95$$

# Gaussian Averaging/ Low Pass Filter

- Gaussian function with mean 0 and standard deviation, σ is

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

3×3 filter with σ = 1

(1/16)

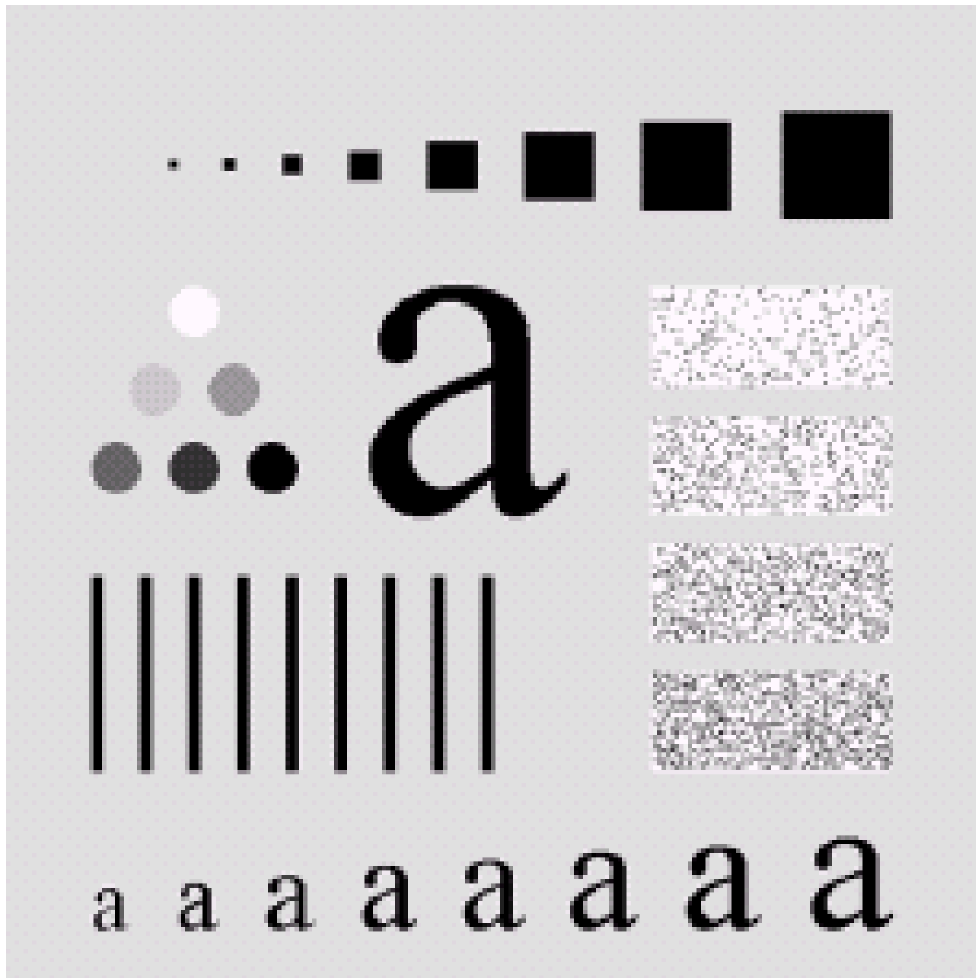
| | | |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 4 | 2 |
| 1 | 2 | 1 |

5×5 filter with σ = 1

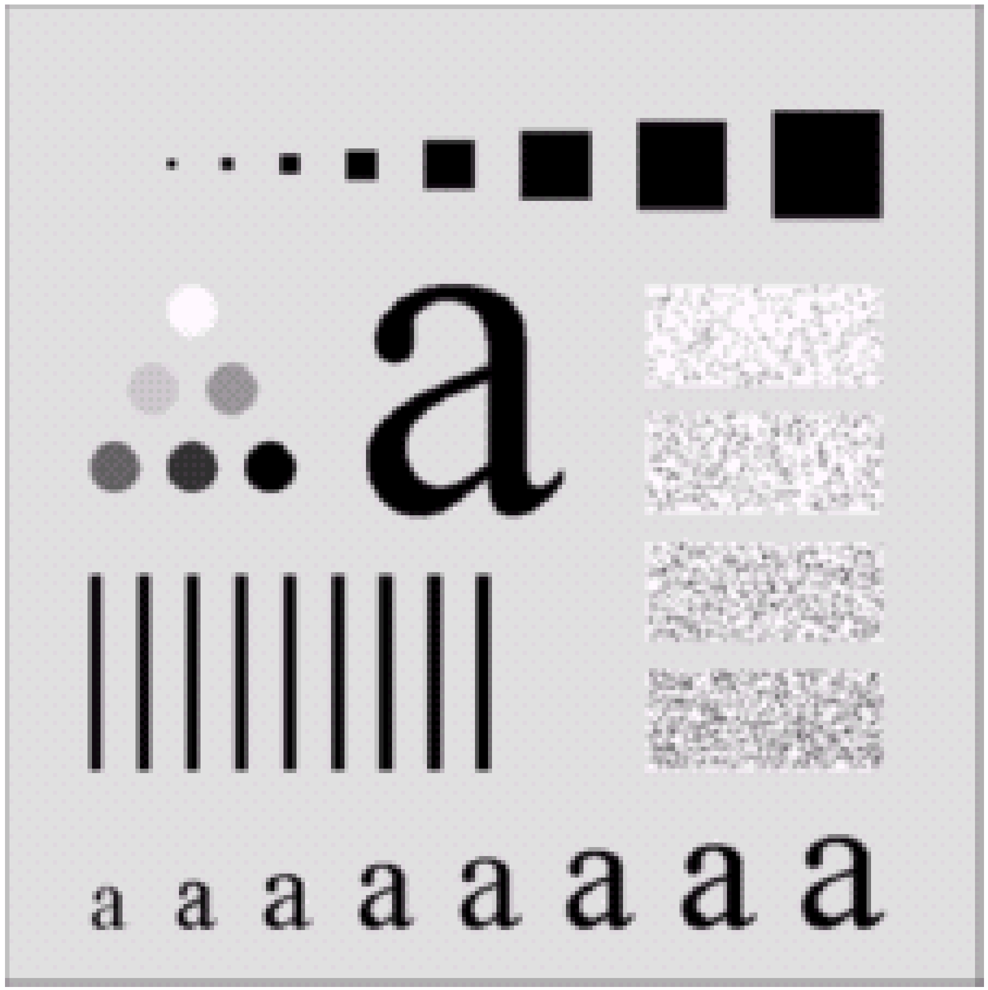
$\frac{1}{273}$

| | | | | |
|---|---|---|---|---|
| 1 | 4 | 7 | 4 | 1 |
| 4 | 16 | 26 | 16 | 4 |
| 7 | 26 | 41 | 26 | 7 |
| 4 | 16 | 26 | 16 | 4 |
| 1 | 4 | 7 | 4 | 1 |

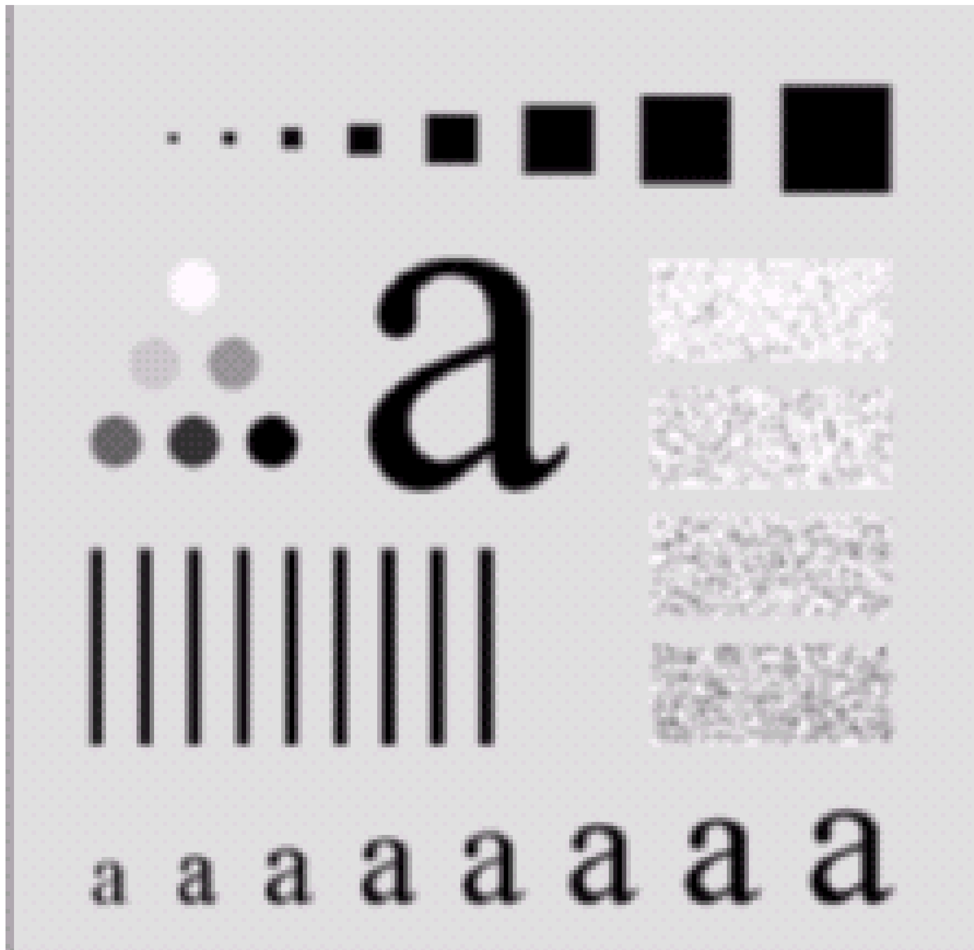
# Original Image



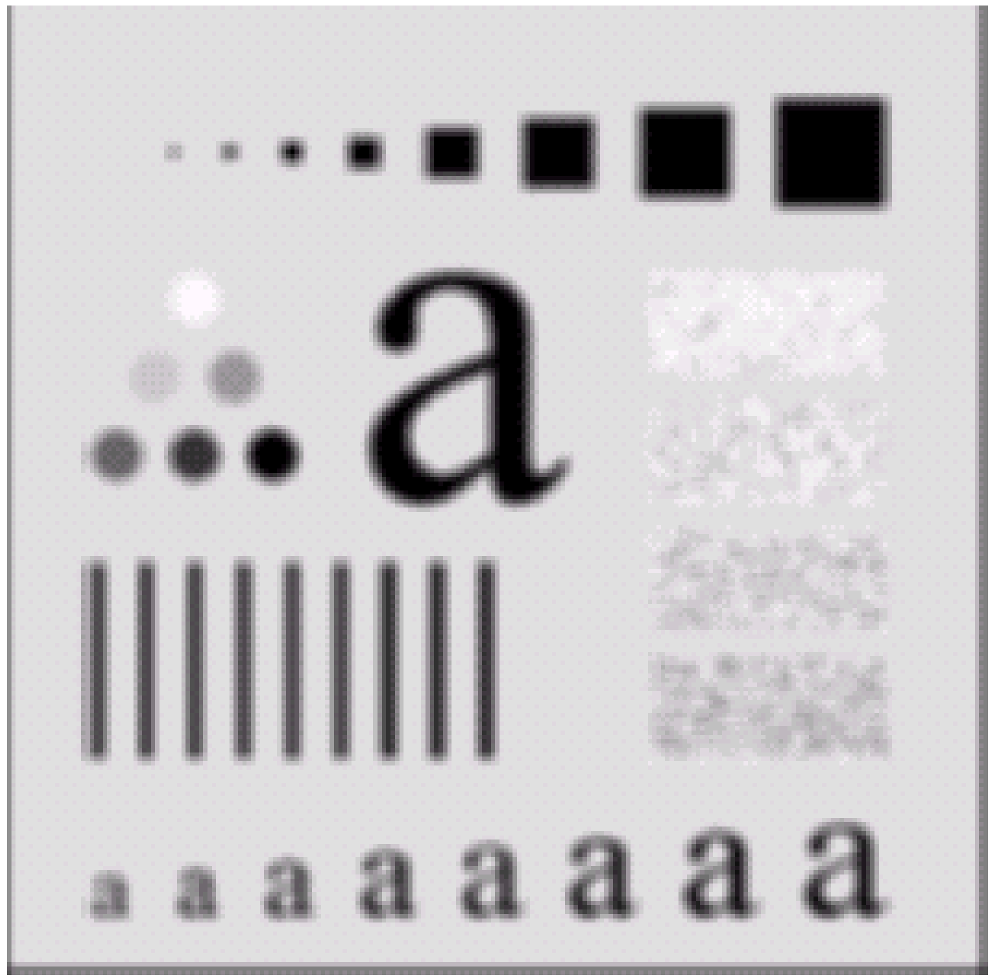
# 3x3 mask



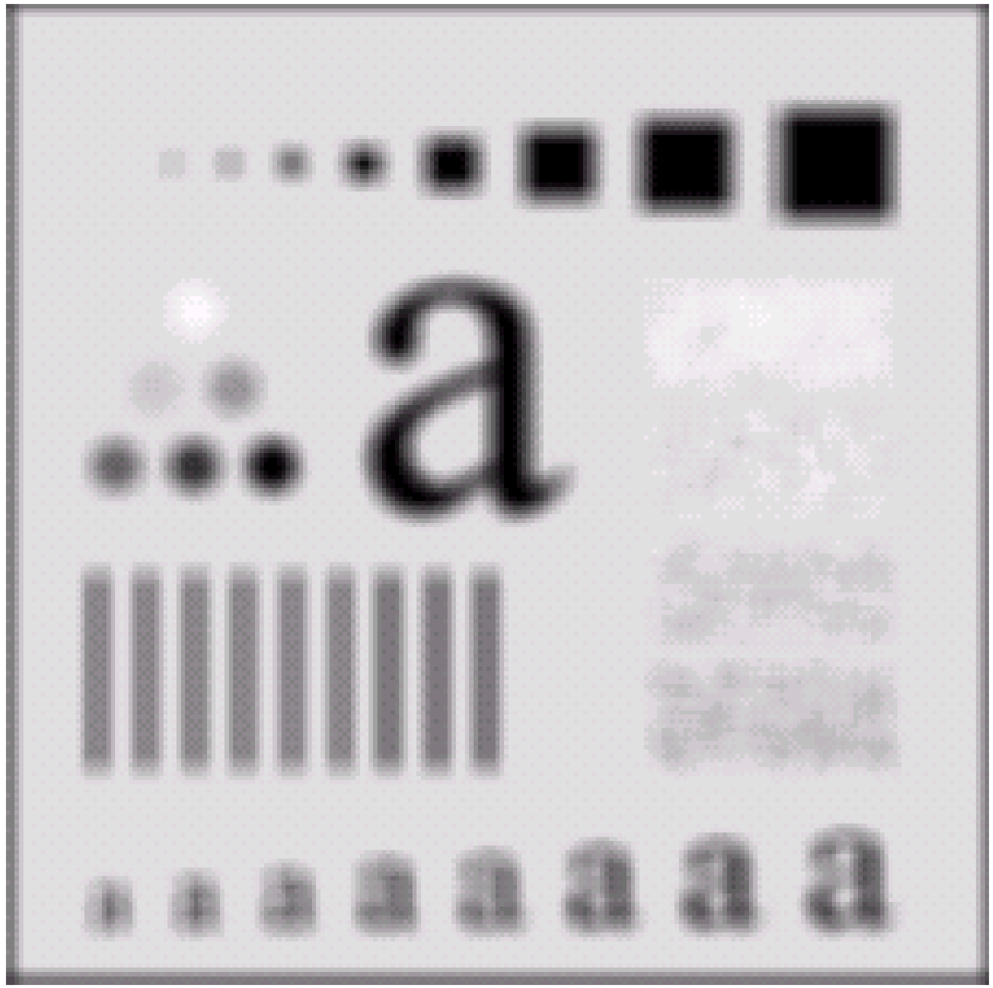
# 5x5 mask



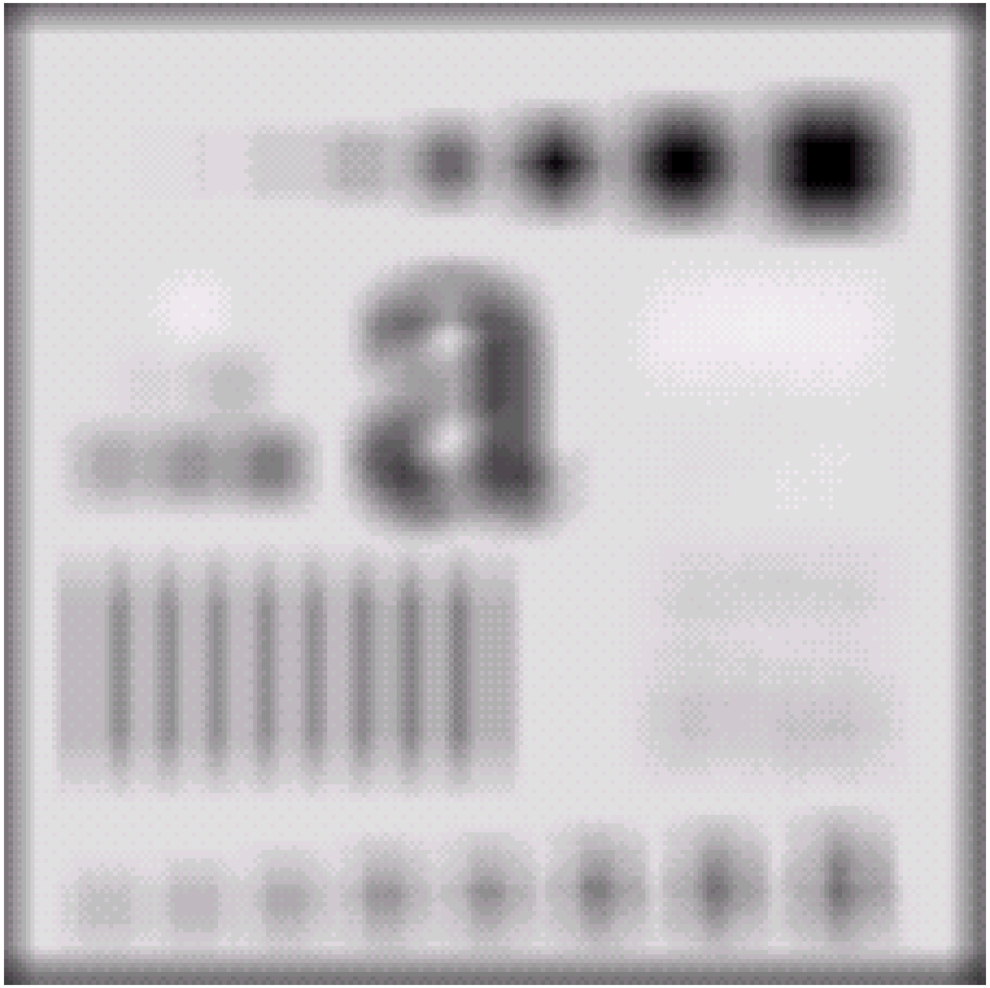
# 9x9 mask



# 15x15 mask

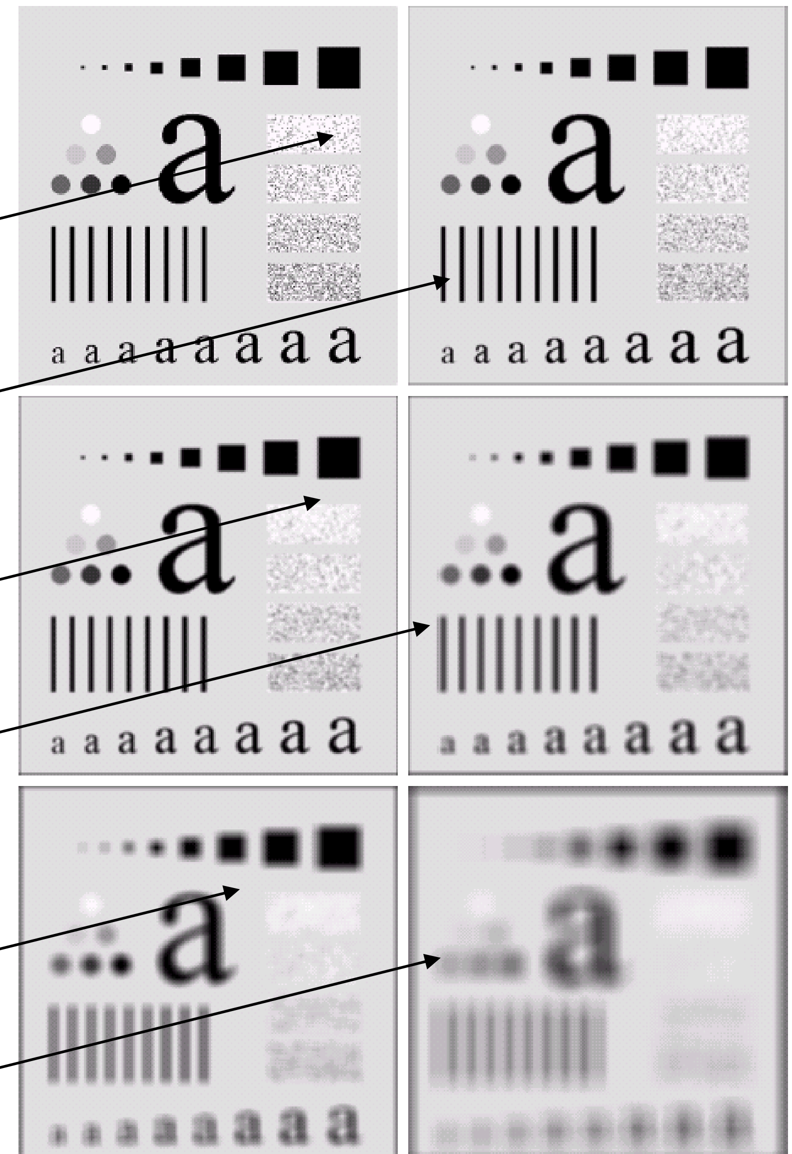


# 35x35 mask



# Image Smoothing Example

Sharpness/detail begins to disappear with increase in size of mask



Original image

3x3

5x5

9x9

15x15

35x35

# Gaussian Averaging/ Low Pass Filter

- The effect of Gaussian smoothing is to blur an image, in a similar fashion to the average filter
- The degree of smoothing is determined by the standard deviation of the Gaussian
- Larger standard deviation Gaussians, more is blurring
- Center pixel is given more weight than neighboring pixels
- This is in contrast to the average filter's uniformly weighted average
- Because of this, a Gaussian provides gentler smoothing and preserves edges better than a similarly sized mean filter

# Weighted Average Filters

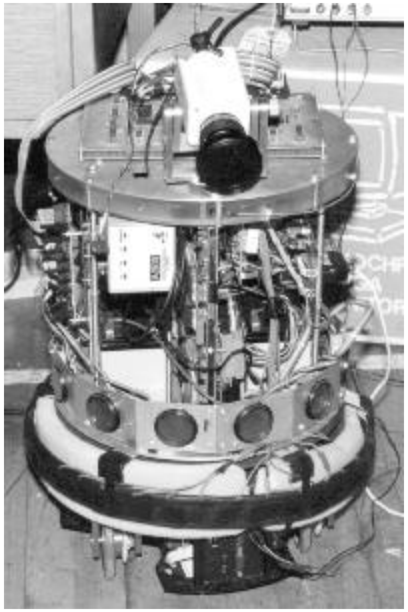
- Gaussian filters are also called weighted average filters
- Elements of mask have different weights
- Provides more effective smoothing
- Pixels closer to the central pixel are more important
- Often referred to as a weighted averaging

$$A = (1/16) \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Weighted averaging filter

# Image filtered with Gaussian filter

Filtered images



Original image

σ = 1

σ = 2

σ = 4

# Gaussian filter to denoise image

Image corrupted by
Gaussian noise with σ = 8

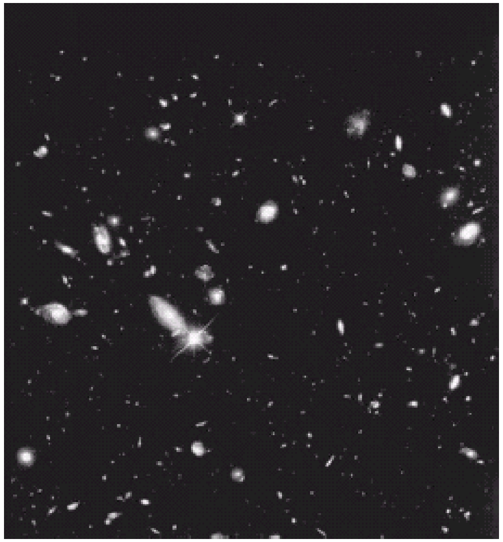
Smoothing with $5 \times 5$
Gaussian filter, σ = 1

Smoothing with $5 \times 5$
average filter

# Limitations of averaging filter

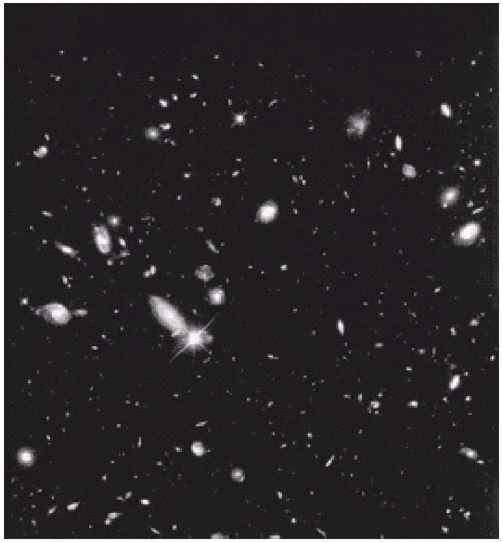
- Leads to blurring of image
- Attenuates impulse noise
- Does not remove impulse noise

# Application of average filter (smoothening & thresholding)

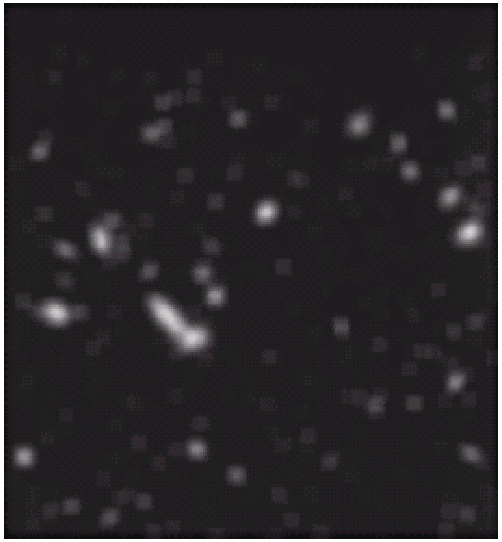


**Original Image**

# Application of average filter (smoothening & thresholding)

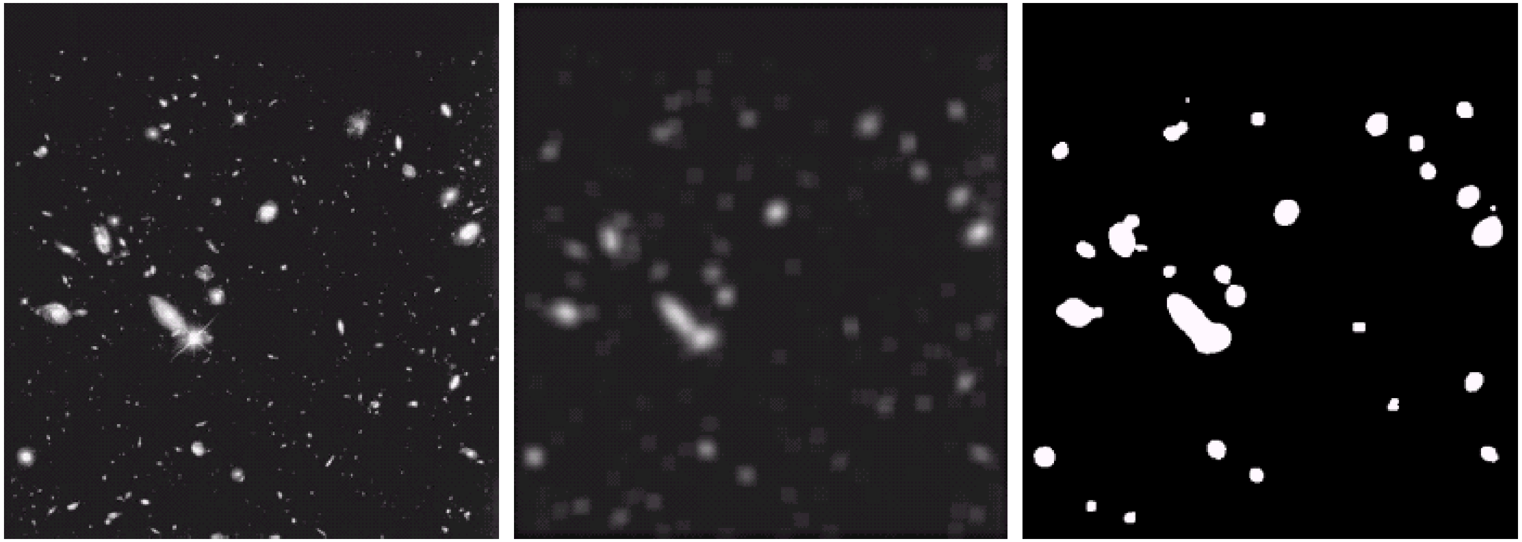


Original Image



Smoothed Image

# Application of average filter (smoothening & thresholding)



Original Image

Smoothed Image
$5 \times 5$ mask

Thresholded Image
$G(x,y) = 255$, if $f(x,y) > 120$
$= 0$, otherwise

- Removes finer details
- Thresholding separates gross details

# Simple Neighbourhood Operations

- **Min:** minimum in the neighbourhood
- **Max:** maximum in the neighbourhood
- **Median:** midpoint value of the set

# Minimum and Maximum Filter

| | | |
|---|---|---|
| 2 | 3 | 6 |
| 1 | 2 | 8 |
| 7 | 4 | 5 |

Image Before filter

→

| | | |
|---|---|---|
| 2 | 3 | 6 |
| 1 | 1 | 8 |
| 7 | 4 | 5 |

Image after minimum filter

| | | |
|---|---|---|
| 2 | 3 | 6 |
| 1 | 8 | 8 |
| 7 | 4 | 5 |

Image after Maximum filter

# Median filter

| | | |
|---|---|---|
| 2 | 3 | 6 |
| 1 | 2 | 8 |
| 7 | 4 | 5 |

Image Before filter

→ [2 3 6 1 2 8 7 4 5]

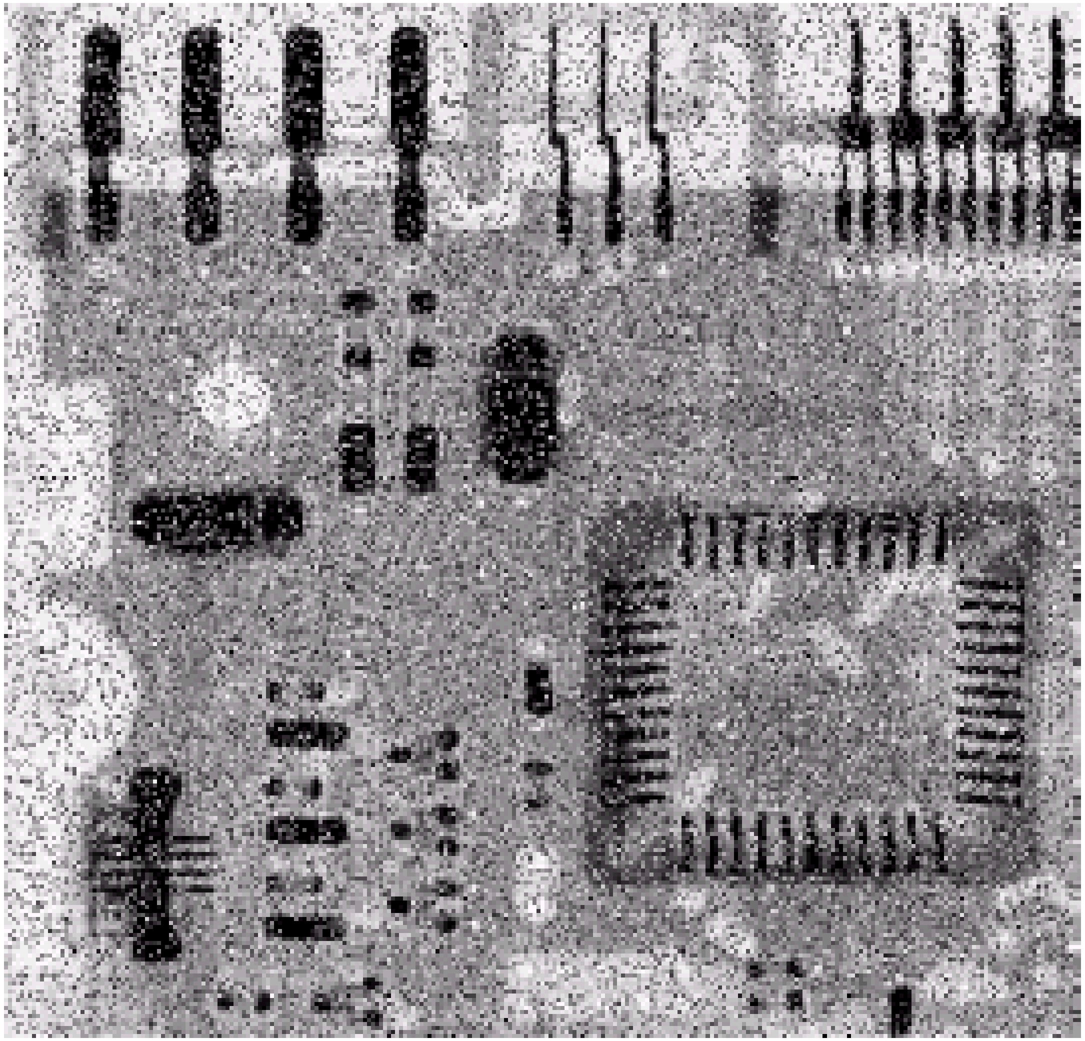
Sort it in ascending

[1 2 2 3 4 5 6 7 8]

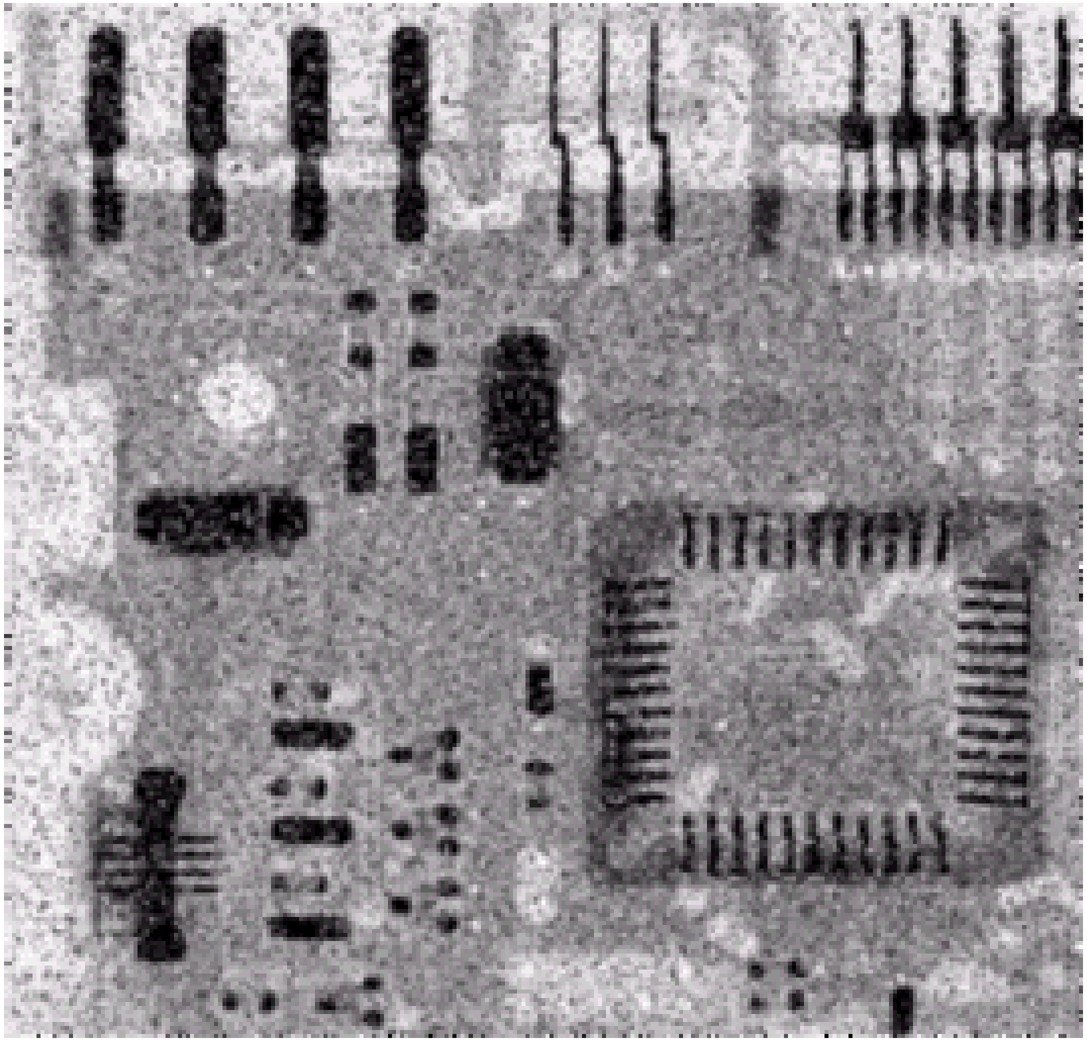
| | | |
|---|---|---|
| 2 | 3 | 6 |
| 1 | 4 | 8 |
| 7 | 4 | 5 |

After median filter

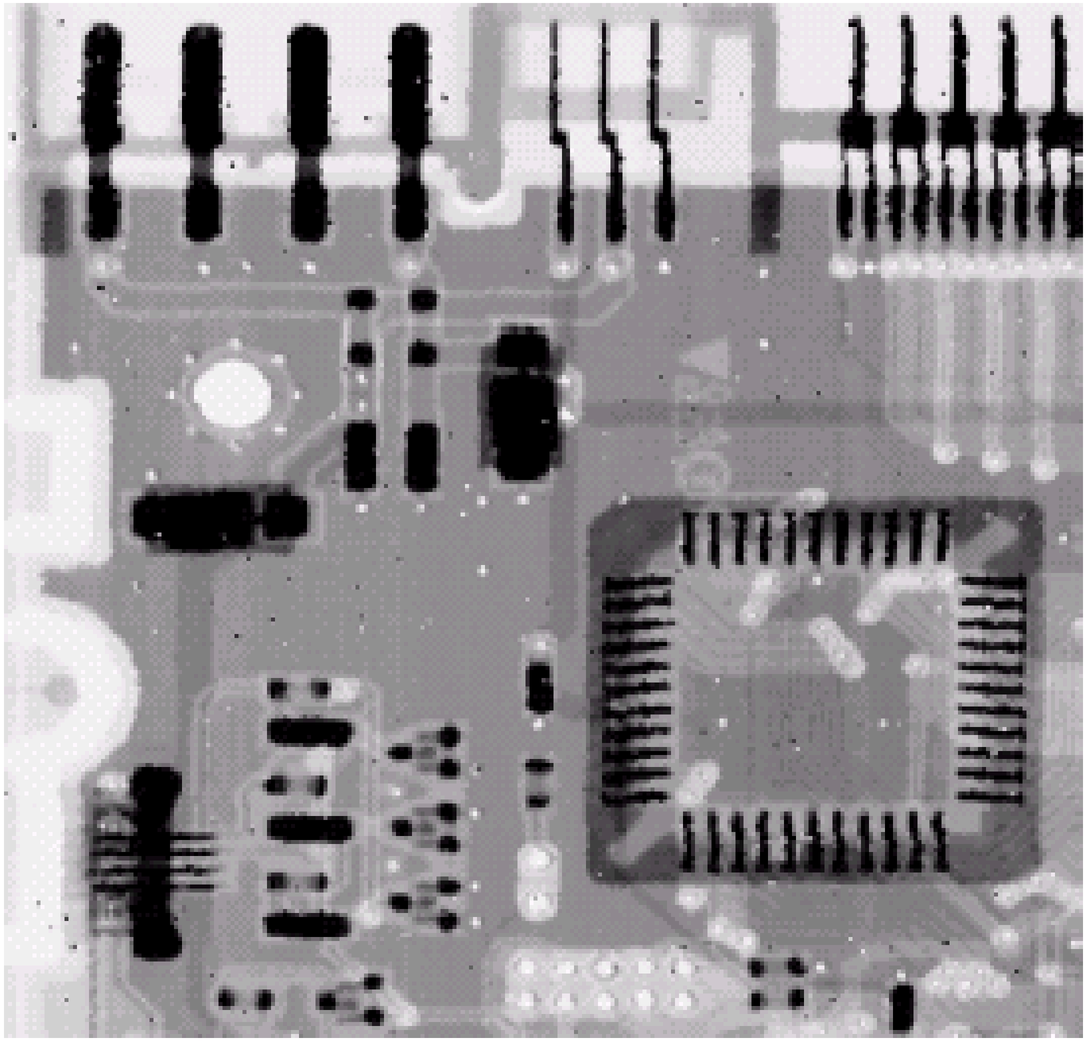
# Image with salt and pepper /impulse noise



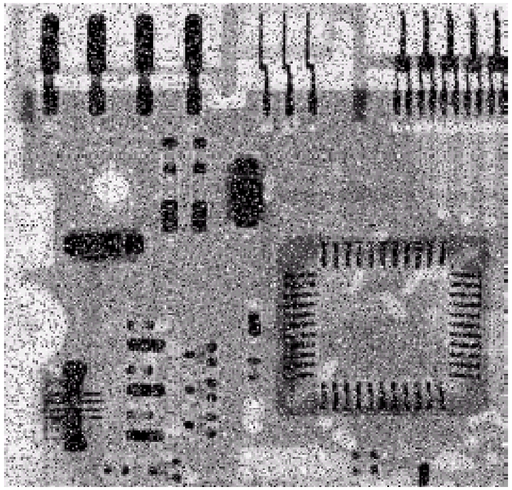
# After Averaging Filter (blurs)



# After Median Filter



# Averaging Filter Vs. Median Filter



**Original Image With Noise**

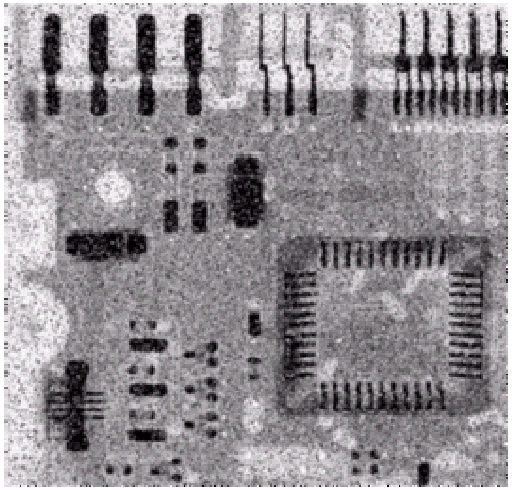


**Image After Averaging Filter**

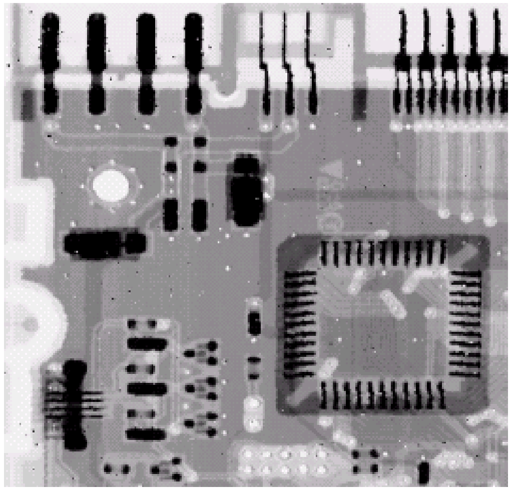


**Image After Median Filter**

Median filter works better than an averaging filter for salt and pepper noise

# Example: Spatial filters

Image matrix is given below. Determine the effect of

1. 3x3 and 5x5 averaging filters
2. 3x3 weighted averaging filter
3. 3x3 Minimum, maximum and median filters

| | | | | |
|---|---|---|---|---|
| 45 | 56 | 42 | 63 | 54 |
| 20 | 47 | 56 | 28 | 53 |
| 63 | 59 | 26 | 38 | 47 |
| 67 | 36 | 27 | 48 | 51 |
| 43 | 36 | 42 | 65 | 43 |

# Example: Spatial filters

Image matrix is given below. Determine the effect of

1. 3x3 and 5x5 averaging filters
2. 3x3 weighted averaging filter
3. 3x3 Minimum, maximum and median filters

| | | | | |
|---|---|---|---|---|
| 45 | 56 | 42 | 63 | 54 |
| 20 | 47 | 56 | 28 | 53 |
| 63 | 59 | 26 | 38 | 47 |
| 67 | 36 | 27 | 48 | 51 |
| 43 | 36 | 42 | 65 | 43 |

First location of filter for 3x3 filter

| | | | | |
|---|---|---|---|---|
| 45 | 56 | 42 | 63 | 54 |
| 20 | 47 | 56 | 28 | 53 |
| 63 | 59 | 26 | 38 | 47 |
| 67 | 36 | 27 | 48 | 51 |
| 43 | 36 | 42 | 65 | 43 |

Next location of mask for 3x3 filter

# Example: Spatial filters

Image matrix is given below. Determine the effect of

1. 3x3 and 5x5 averaging filters and minimum filter

| | | | | |
|---|---|---|---|---|
| 45 | 56 | 42 | 63 | 54 |
| 20 | 47 | 56 | 28 | 53 |
| 63 | 59 | 26 | 38 | 47 |
| 67 | 36 | 27 | 48 | 51 |
| 43 | 36 | 42 | 65 | 43 |

$*$(1/9)

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

=

| | | | | |
|---|---|---|---|---|
| 45 | 56 | 42 | 63 | 54 |
| 20 | 46 | 46 | 45 | 53 |
| 63 | 44 | 41 | 42 | 47 |
| 67 | 44 | 42 | 43 | 51 |
| 43 | 36 | 42 | 65 | 43 |

| | | | | |
|---|---|---|---|---|
| 45 | 56 | 42 | 63 | 54 |
| 20 | 47 | 56 | 28 | 53 |
| 63 | 59 | 26 | 38 | 47 |
| 67 | 36 | 27 | 48 | 51 |
| 43 | 36 | 42 | 65 | 43 |

$*$(1/25)

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |

=

| | | | | |
|---|---|---|---|---|
| 45 | 56 | 42 | 63 | 54 |
| 20 | 47 | 56 | 28 | 53 |
| 63 | 59 | 46 | 38 | 47 |
| 67 | 36 | 27 | 48 | 51 |
| 43 | 36 | 42 | 65 | 43 |

# Example: Spatial filters

Image matrix is given below. Determine the effect of

1. 3x3 weighted averaging filters

| | | | | |
|---|---|---|---|---|
| 45 | 56 | 42 | 63 | 54 |
| 20 | 47 | 56 | 28 | 53 |
| 63 | 59 | 26 | 38 | 47 |
| 67 | 36 | 27 | 48 | 51 |
| 43 | 36 | 42 | 65 | 43 |

$* (1/16)$

| | | |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 4 | 2 |
| 1 | 2 | 1 |

=

| | | | | |
|---|---|---|---|---|
| 45 | 56 | 42 | 63 | 54 |
| 20 | 47 | 45 | 44 | 53 |
| 63 | 47 | 39 | 40 | 47 |
| 67 | 43 | 38 | 44 | 51 |
| 43 | 36 | 42 | 65 | 43 |

| | | | | |
|---|---|---|---|---|
| 45 | 56 | 42 | 63 | 54 |
| 20 | 47 | 56 | 28 | 53 |
| 63 | 59 | 26 | 38 | 47 |
| 67 | 36 | 27 | 48 | 51 |
| 43 | 36 | 42 | 65 | 43 |

→min(3x3) =

| | | | | |
|---|---|---|---|---|
| 45 | 56 | 42 | 63 | 54 |
| 20 | 20 | 26 | 26 | 53 |
| 63 | 20 | 26 | 26 | 47 |
| 67 | 26 | 26 | 26 | 51 |
| 43 | 36 | 42 | 65 | 43 |

# Example: Spatial filters

Image matrix is given below. Determine the effect of

1. 3x3 weighted averaging filters

| | | | | |
|---|---|---|---|---|
| 45 | 56 | 42 | 63 | 54 |
| 20 | 47 | 56 | 28 | 53 |
| 63 | 59 | 26 | 38 | 47 |
| 67 | 36 | 27 | 48 | 51 |
| 43 | 36 | 42 | 65 | 43 |

→max(3x3) =

| | | | | |
|---|---|---|---|---|
| 45 | 56 | 42 | 63 | 54 |
| 20 | 63 | 63 | 63 | 53 |
| 63 | 67 | 59 | 56 | 47 |
| 67 | 67 | 65 | 65 | 51 |
| 43 | 36 | 42 | 65 | 43 |

| | | | | |
|---|---|---|---|---|
| 45 | 56 | 42 | 63 | 54 |
| 20 | 47 | 56 | 28 | 53 |
| 63 | 59 | 26 | 38 | 47 |
| 67 | 36 | 27 | 48 | 51 |
| 43 | 36 | 42 | 65 | 43 |

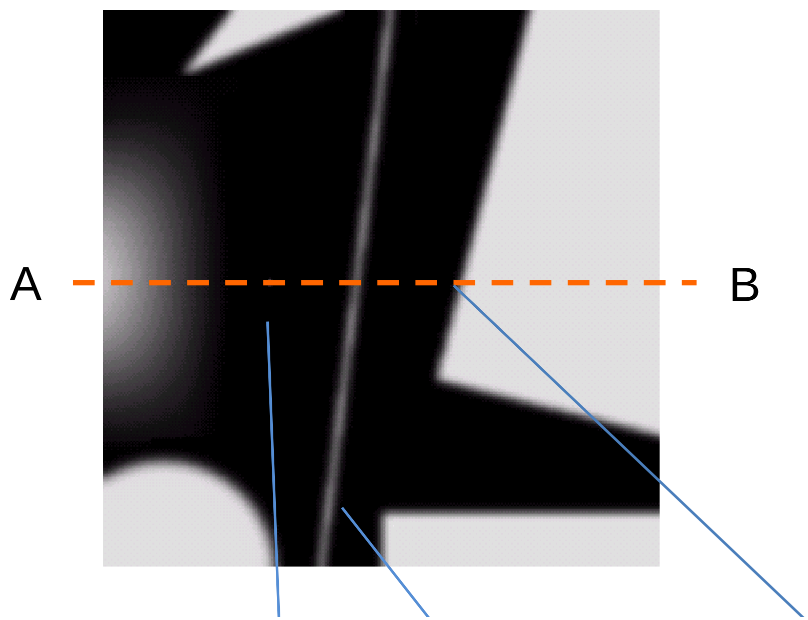
→median(3x3) =

| | | | | |
|---|---|---|---|---|
| 45 | 56 | 42 | 63 | 54 |
| 20 | 47 | 47 | 47 | 53 |
| 63 | 47 | 38 | 47 | 47 |
| 67 | 42 | 38 | 43 | 51 |
| 43 | 36 | 42 | 65 | 43 |

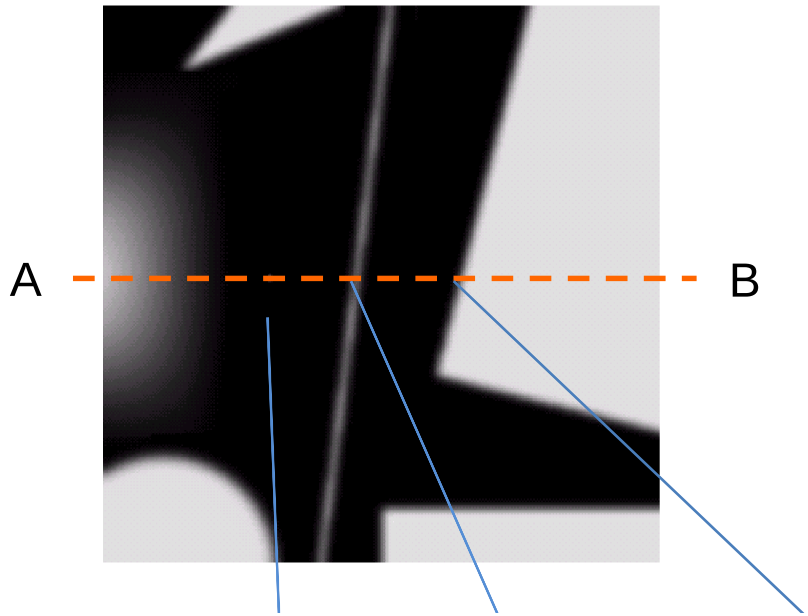
# Sharpening Spatial Filters

- Remove blurring in images
- Highlight transition in intensity (edges)
- Used in electronic printing, medical imaging, industrial inspection etc.
- Uses spatial differentiation
- Differentiation measures the rate of change of a function
- Differentiation are also called derivative

# First Derivative

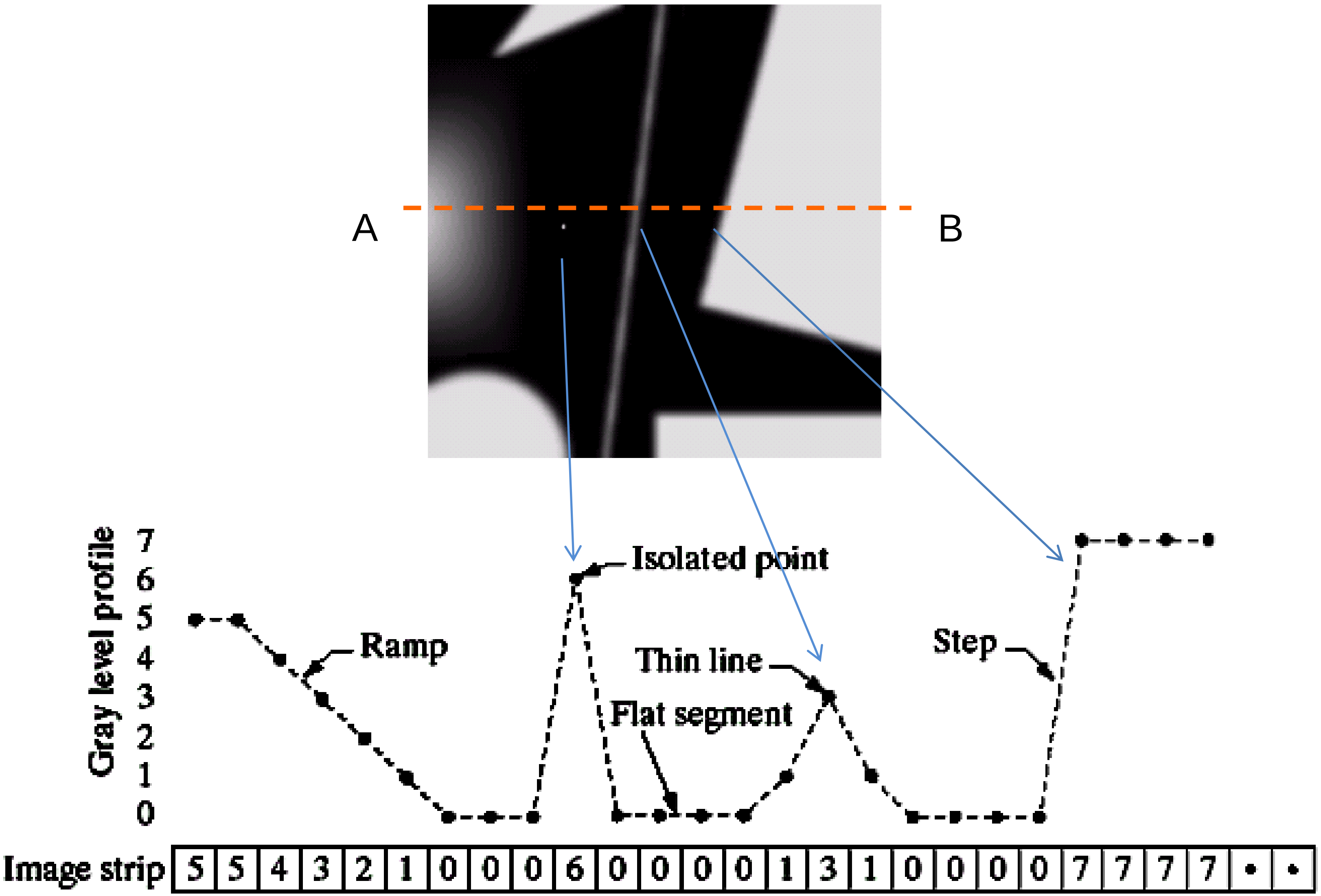


# First Derivative




| | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Image strip | 5 | 5 | 4 | 3 | 2 | 1 | 0 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 1 | 3 | 1 | 0 | 0 | 0 | 0 | 7 | 7 | 7 | 7 | • | • |

# First Derivative



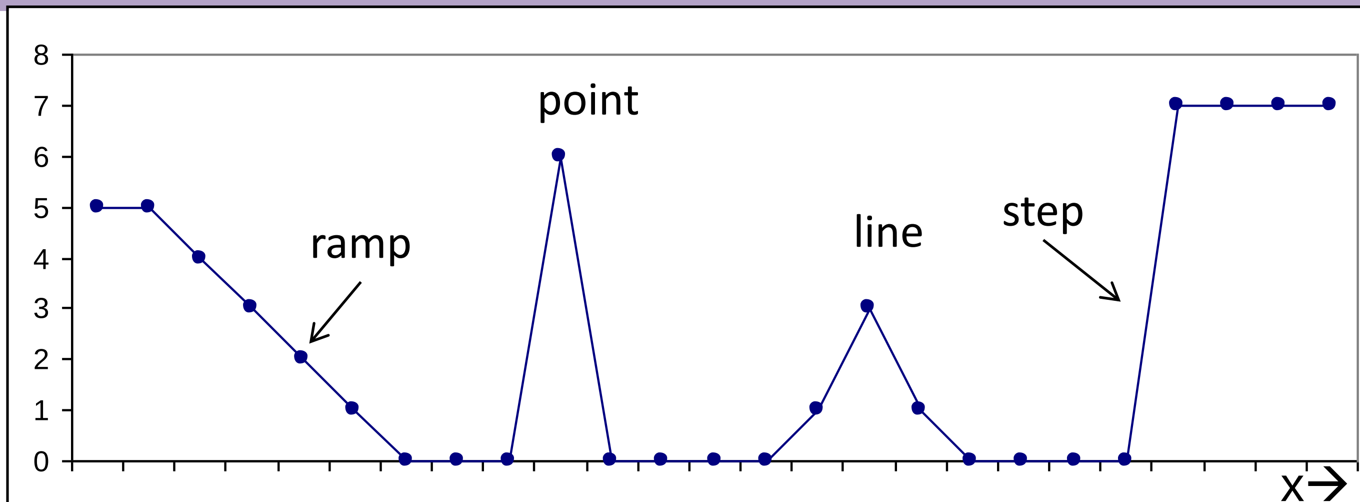
# First Derivative

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

- f(x) is pixel value at location, x
- Difference between consecutive values
- Measures the rate of change of the function

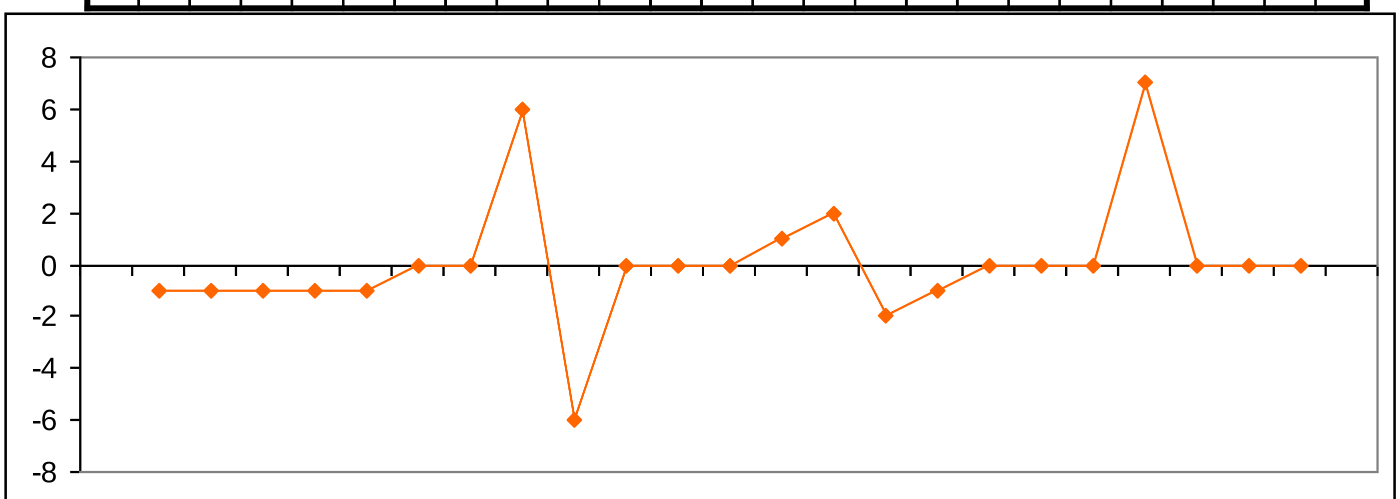
# First Derivative

Intensity profile, f(x)




| | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| f(x) | 5 | 5 | 4 | 3 | 2 | 1 | 0 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 1 | 3 | 1 | 0 | 0 | 0 | 0 | 7 | 7 | 7 | 7 |
| f'(x) | 0 | -1 | -1 | -1 | -1 | -1 | 0 | 0 | 6 | -6 | 0 | 0 | 0 | 1 | 2 | -2 | -1 | 0 | 0 | 0 | 7 | 0 | 0 | 0 | |

First derivative, f'(x)




- Non zero at the start and during ramp
- Non zero at the start and zero during the step

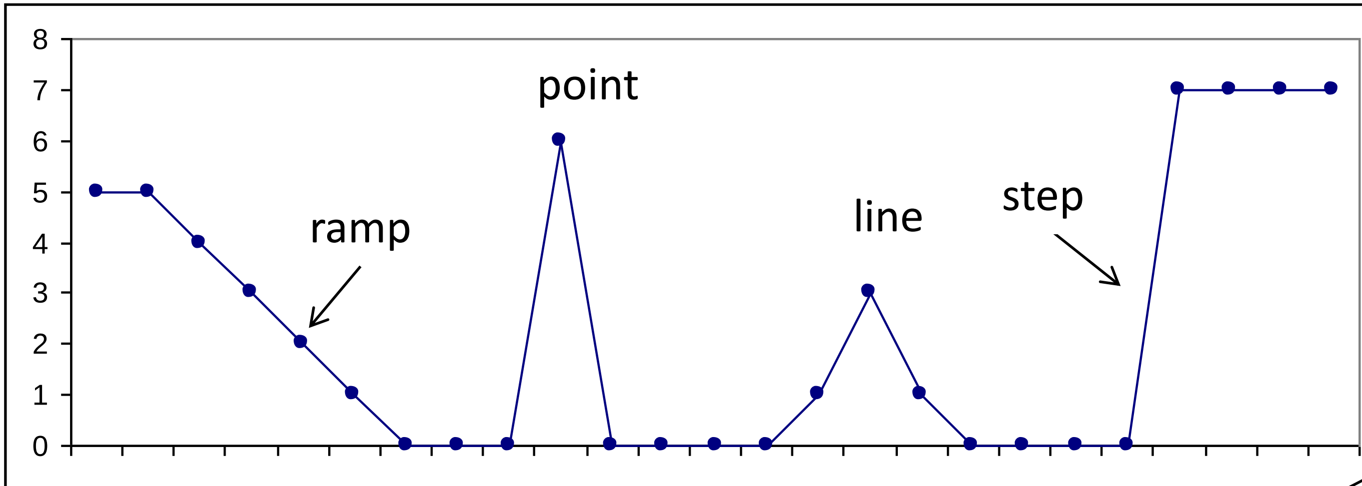
# Second Derivative

$$\frac{\partial^2 f}{\partial^2 x} = f(x+1) + f(x-1) - 2f(x)$$

Considers values both before and after the current value

# Second Derivative

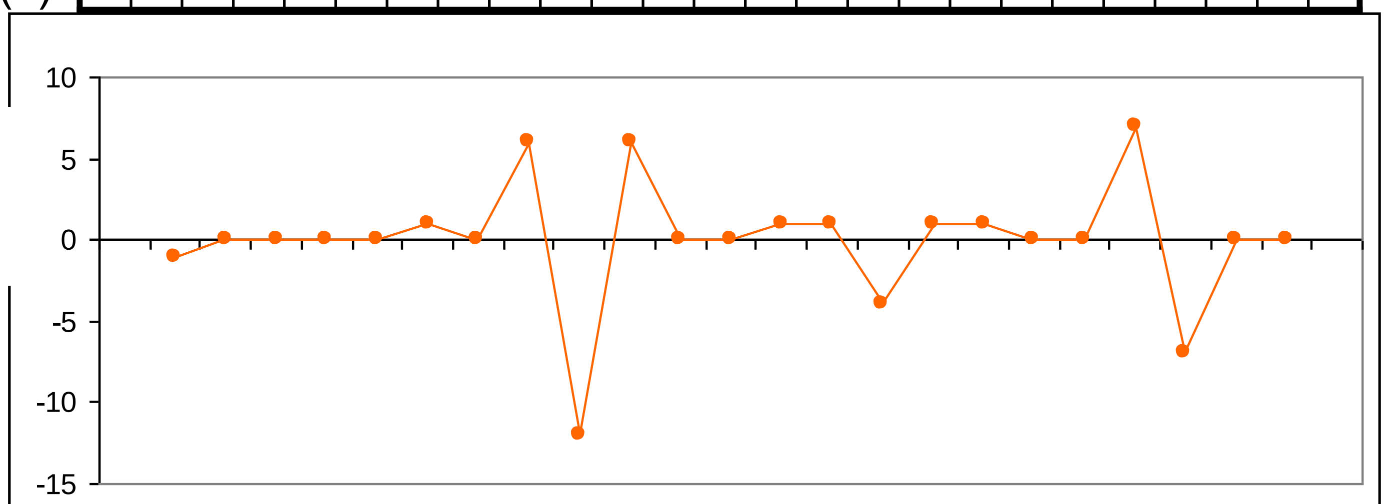
Intensity profile, f(x)




Sign changes at onset of step

| | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| f(x) | 5 | 5 | 4 | 3 | 2 | 1 | 0 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 1 | 3 | 1 | 0 | 0 | 0 | 0 | 7 | 7 | 7 | 7 |
| f''(x) | | -1 | 0 | 0 | 0 | 0 | 1 | 0 | 6 | -12 | 6 | 0 | 0 | 1 | 1 | -4 | 1 | 1 | 0 | 0 | 7 | -7 | 0 | 0 | |

Second derivative, f''(x)

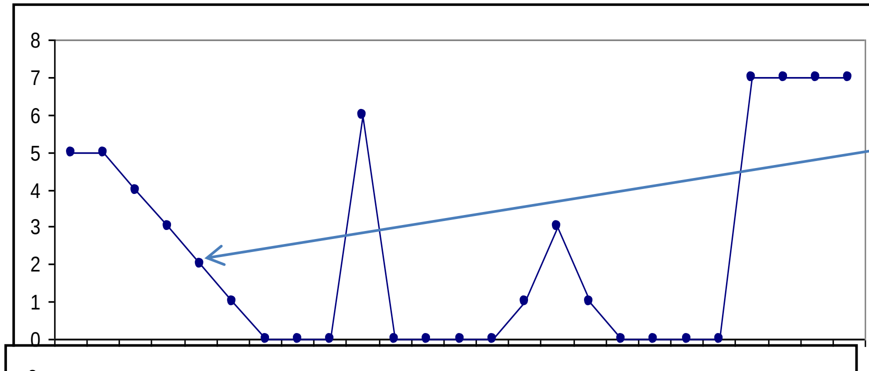



Non zero at the onset and end of ramp and step

Zero in the middle

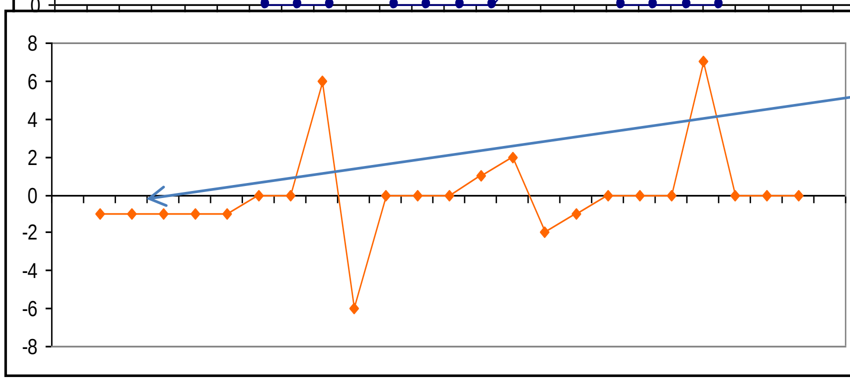
# 1$^{st}$ and 2$^{nd}$ Derivatives

f(x)



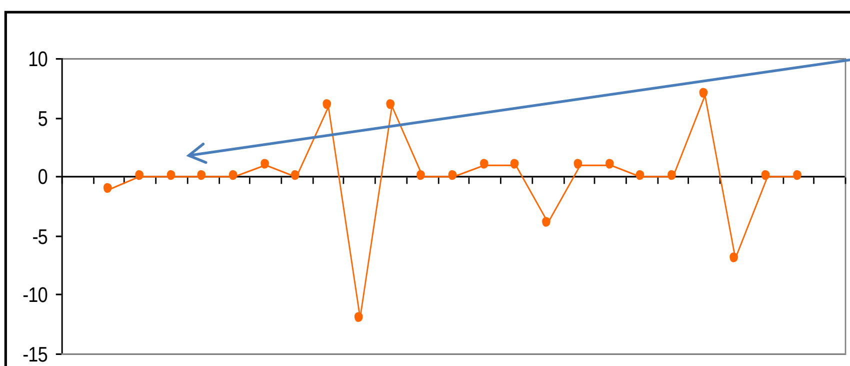

ramp like transition

f'(x)




first derivative results in thick edges because of nonzero value along a ramp

f''(x)




Second derivative produces a double edge one pixel thick, separated by zeros. Therefore sharpens fine details better than first derivative

# Example derivatives

- Intensity of pixels along a line in an image is

  f(x)=[1 4 1 0 0 7 7 7 7 0 0 0 1 2 3 4 5 4 3 2 1]

  Compute first order derivative
- $f'(x) = f(x+1)-f(x)$
- f'(x) = [3,-3,-1,0,7,0,0,0,-7,0,0,1,1,1,1,1,-1,-1,-1,-1]
- Compute second order derivative
- $f''(x) = f(x+1)+f(x-1)-2f(x)$
- f''(x)= [2,-6,2,1,7,-7,0,0,-7,7,0,1,0,0,0,-2,0,0,0,0]

# First Derivative(Gradient) of image

- Gradient is first order derivative
- Gradient of *f(x,y)*

$$\nabla \mathrm{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

# Gradient

- Magnitude of gradient (gradient image)

$$M(x,y) = mag(\nabla \mathrm{f})$$

$$= \left[G_x^2 + G_y^2\right]^{1/2}$$

$$= \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2\right]^{1/2}$$

- Size of *M(x,y)* is same as image.

$$M(x,y) \approx |G_x| + |G_y|$$

# Gradient using $3 \times 3$ filter

| | | |
|---|---|---|
| $z_1$ | $z_2$ | $z_3$ |
| $z_4$ | $z_5$ | $z_6$ |
| $z_7$ | $z_8$ | $z_9$ |

$G_x$

$$M(x,y) \approx |(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)|$$

$$+|(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)|$$

$G_y$

# Sobel Operators

| | | |
|---|---|---|
| $z_1$ | $z_2$ | $z_3$ |
| $z_4$ | $z_5$ | $z_6$ |
| $z_7$ | $z_8$ | $z_9$ |

→

$G_x$

| | | |
|---|---|---|
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| 1 | 2 | 1 |

+

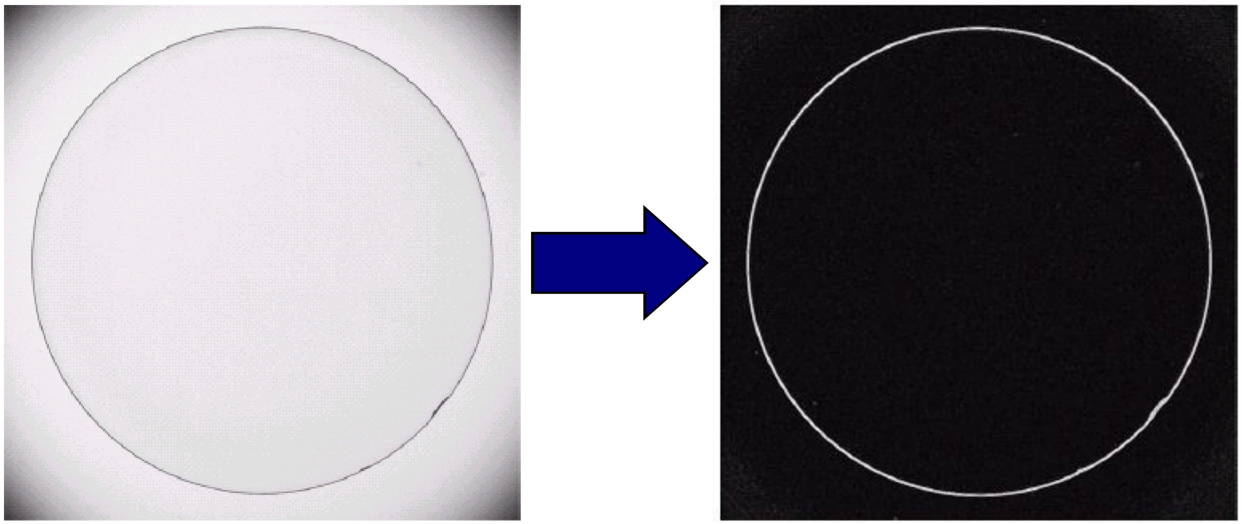
$G_y$

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

$$M(x,y) \approx |(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)|$$

$$+|(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)|$$

# Sobel Example



Sobel filters are typically used for edge detection

**An image of a contact lens which is enhanced in order to make defects more obvious**

# The Laplacian operator

$$\nabla^2 f = \frac{\partial^2 f}{\partial^2 x} + \frac{\partial^2 f}{\partial^2 y}$$

where the partial 2nd order derivative in the *x* and *y* direction is defined as

$$\frac{\partial^2 f}{\partial^2 x} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

$$\frac{\partial^2 f}{\partial^2 y} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

$$\nabla^2 f = [f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)]$$

- Laplacian filters are derivative filters used to extract the vertical as well as horizontal edges from an image
- Sobel filters are single derivative filters, they can only find edges in a single dimension

# The Laplacian operator

$$\nabla^2 f = [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)]$$

or

$$\nabla^2 f = -[f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) + 4f(x,y)]$$

Image

| | | |
|---|---|---|
| a | b | d |
| e | f | g |
| h | i | j |

# The Laplacian operator

$$\nabla^2 f = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

Negative Laplacian

or

$$\nabla^2 f = -[f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) + 4f(x,y)]$$

Positive Laplacian

Image

| | | |
|---|---|---|
| a | b | d |
| e | f | g |
| h | i | j |

Negative Laplacian

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | -4 | 1 |
| 0 | 1 | 0 |

# The Laplacian operator

$$\nabla^2 f = [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)]$$

$$\nabla^2 f = -[-f(x+1,y) - f(x-1,y) - f(x,y+1) - f(x,y-1) + 4f(x,y)]$$

Image

| | | |
|---|---|---|
| a | b | d |
| e | f | g |
| h | i | j |

Negative Laplacian

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | -4 | 1 |
| 0 | 1 | 0 |

or

Positive Laplacian

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 4 | -1 |
| 0 | -1 | 0 |

# The Laplacian

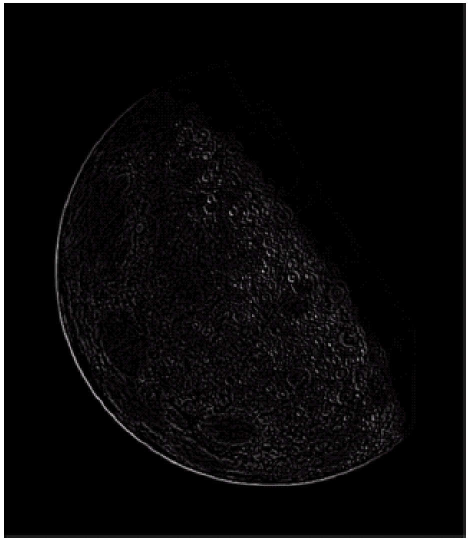
Highlights edges and other discontinuities

Original
Image

# The Laplacian

Highlights edges and other discontinuities

Original
Image

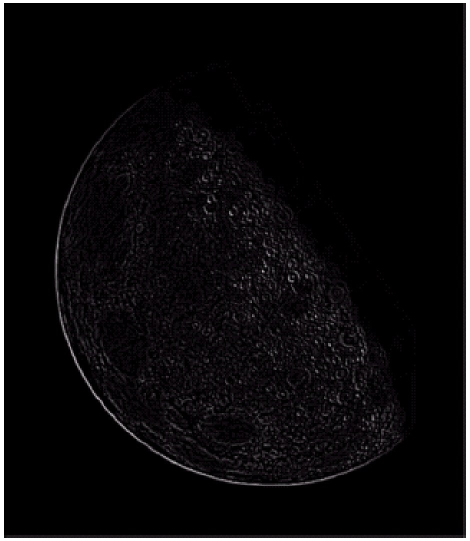


Laplacian
Filtered Image

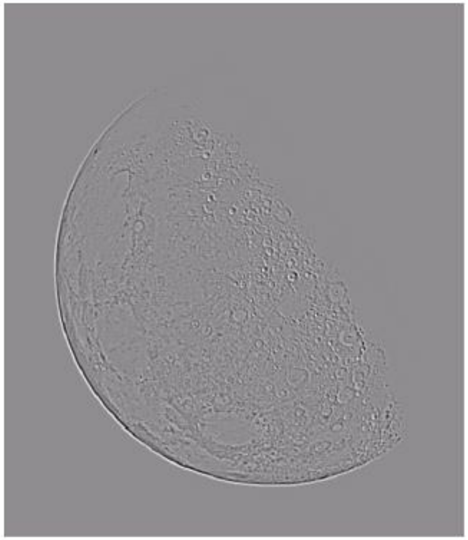
# The Laplacian

Highlights edges and other discontinuities

Original
Image



Laplacian
Filtered Image



Laplacian
Filtered Image
Scaled for Display

# The Laplacian operator

$$\nabla^2 f = [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)]$$

$$\nabla^2 f = -[-f(x+1,y) - f(x-1,y) - f(x,y+1) - f(x,y-1) + 4f(x,y)]$$

- Laplacian detects the edges
- For sharpening the images
- If negative Laplacian is applied on the image, subtract the resultant image from the original image
- If positive Laplacian is applied then add the resultant image to original image

# Image Enhancement using Laplacian

Original Image

-

negative Laplacian Filtered Image

=

Sharpened Image

Sharpened image has enhanced edges and fine details

# Image Enhancement using Laplacian

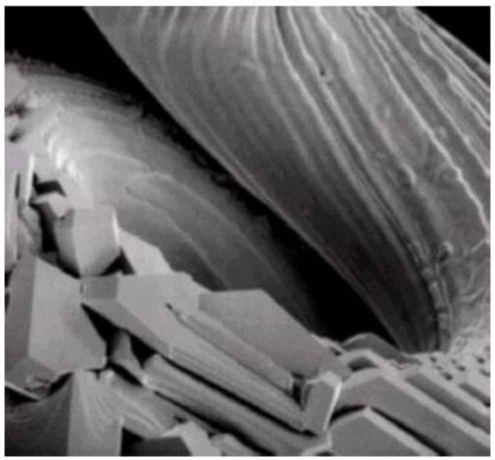
# Simplified Image Enhancement

The entire enhancement can be combined into a single filtering operation

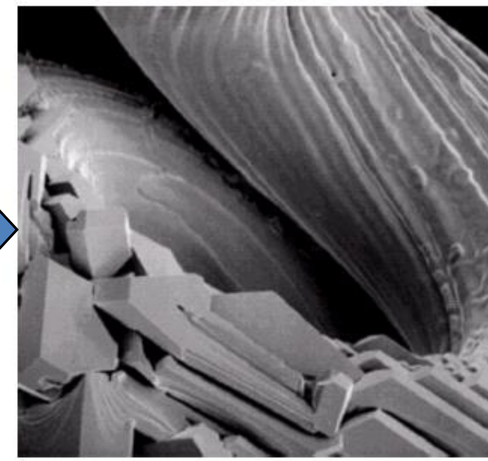
$$
\begin{aligned}
g(x, y) &= f(x, y) - \nabla^2 f \\
&= f(x, y) - [f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)] \\
&= 5f(x, y) - [f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1)]
\end{aligned}
$$

# Image Enhancement using the Lapacian

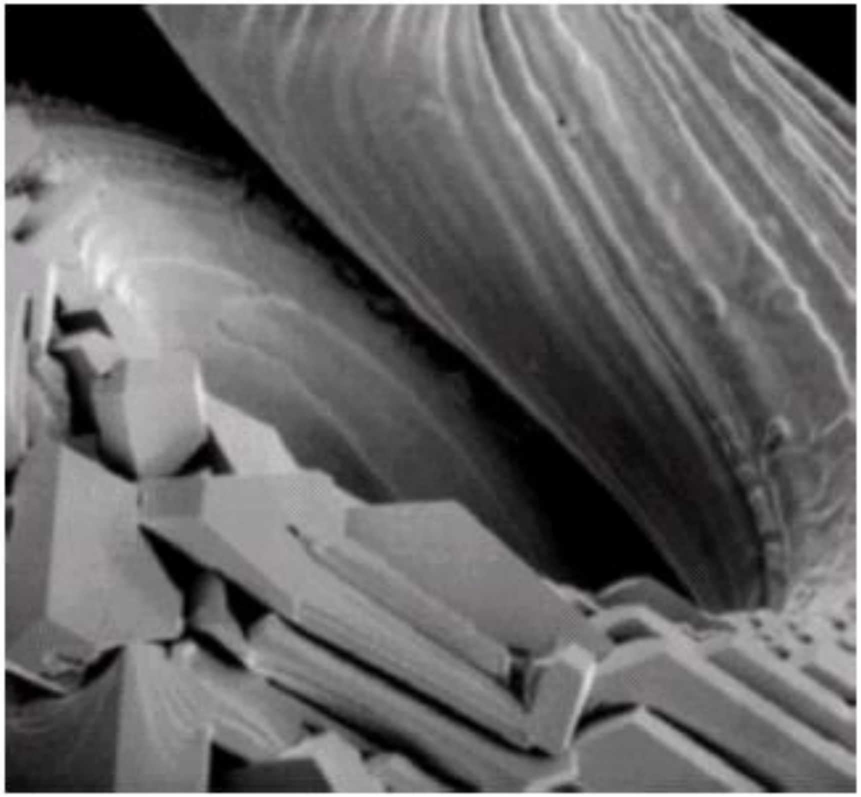
$$g(x,y) = 5f(x,y) - f(x+1,y) - f(x-1,y) - f(x,y-1) - f(x,y+1)$$



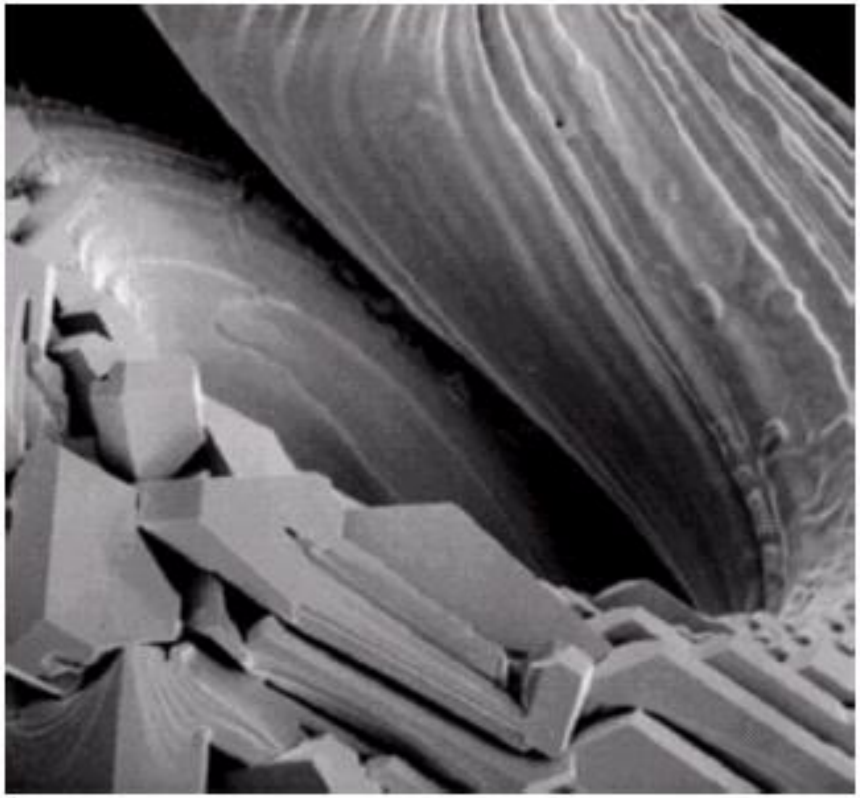
| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |



# Simplified Image Enhancement



original



enhanced

# Variants of Laplacian

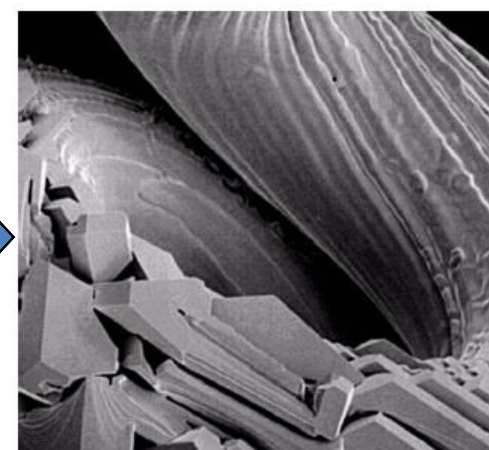
| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | -4 | 1 |
| 0 | 1 | 0 |

Simple Laplacian

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | -8 | 1 |
| 1 | 1 | 1 |

Variant of Laplacian

| | | |
|---|---|---|
| -1 | -1 | -1 |
| -1 | 8 | -1 |
| -1 | -1 | -1 |



# Example Variant of Laplacian

| | | | | |
|---|---|---|---|---|
| 1 | 4 | 5 | 2 | 7 |
| 0 | 4 | 0 | 6 | 2 |
| 3 | 2 | 1 | 0 | 2 |
| 7 | 5 | 2 | 3 | 1 |
| 4 | 3 | 2 | 5 | 1 |

*

| | | |
|---|---|---|
| -1 | -1 | -1 |
| -1 | 8 | -1 |
| -1 | -1 | -1 |

=

| | | | | |
|---|---|---|---|---|
| 1 | 4 | 5 | 2 | 7 |
| 0 | 16 | -24 | 29 | 2 |
| 3 | -6 | -14 | -17 | 2 |
| 7 | -16 | -5 | 10 | 1 |
| 4 | 3 | 2 | 5 | 1 |

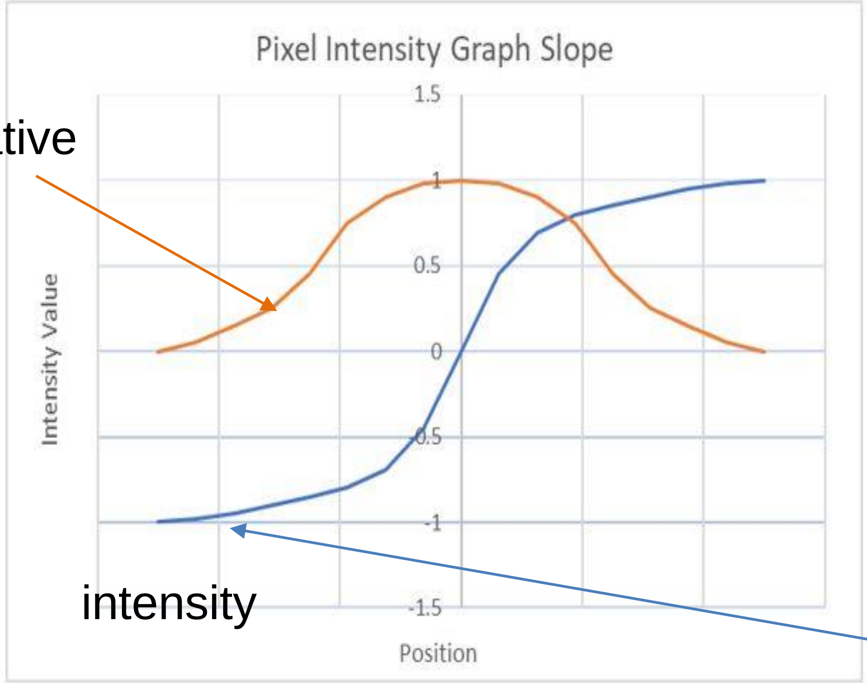
# Laplacian for Edge Detection

- Edge detection is an important part of image processing and computer vision applications
- It is used to detect objects, locate boundaries, and extract features
- Edge detection is about identifying sudden, local changes in the intensity values of the pixels in an image
- Edge detection algorithms like the Sobel Operator work on the first derivative of an image
- It checks where the slope of the graph of the intensity reaches a peak, and that peak is marked as an edge

# First Derivative for Edge Detection

- Limitation: first derivative of an image might be subject to noise
- Local peaks in the slope of the intensity values might be due to shadows or tiny color changes that are not edges

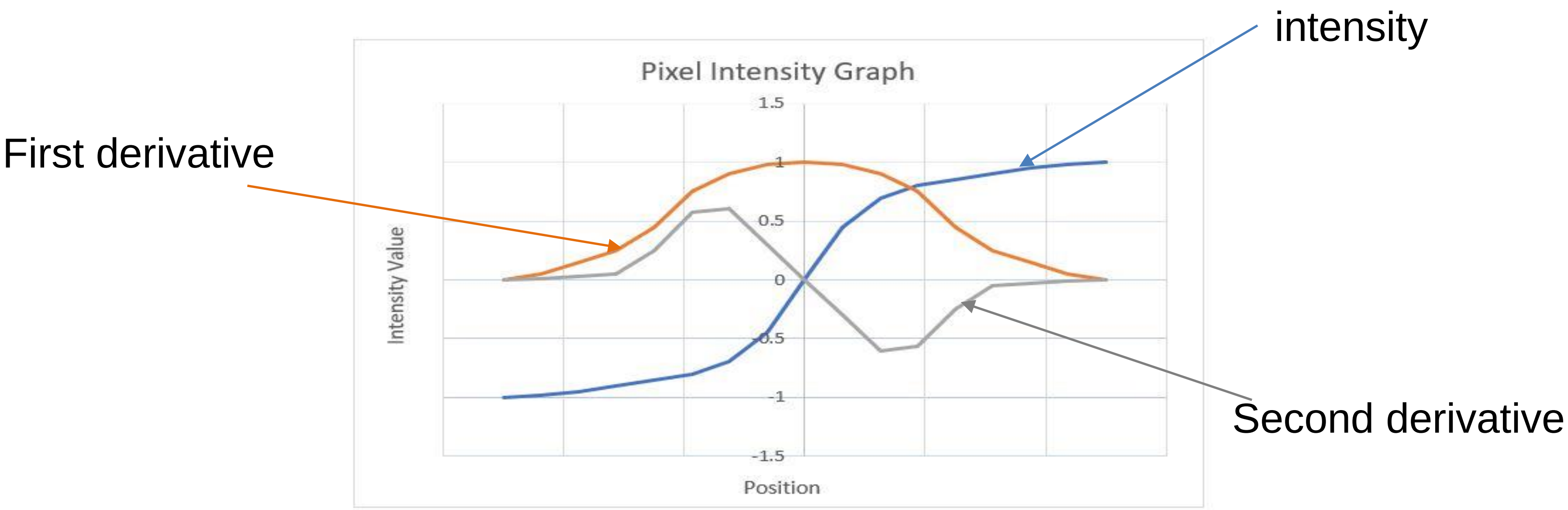
First derivative




intensity

# Second Derivative for Edge Detection

- Second derivative is the slope of the first derivative curve
- An edge occurs where the graph of the second derivative crosses zero
- Second derivative-based method is called the Laplacian algorithm
- The Laplacian algorithm is also subject to noise



# Laplacian of Gaussian (LoG)

- Defined as the Laplace operator applied to a Gaussian kernel
- Laplace operator is

$$\nabla^2 f = \frac{\partial^2 f}{\partial^2 x} + \frac{\partial^2 f}{\partial^2 y}$$

- Since second order derivatives are sensitive to noise
- Image is blurred using Gaussian filter before applying Laplace operator
- Laplace operator is

$$\Delta(I * G) = I * \Delta G$$

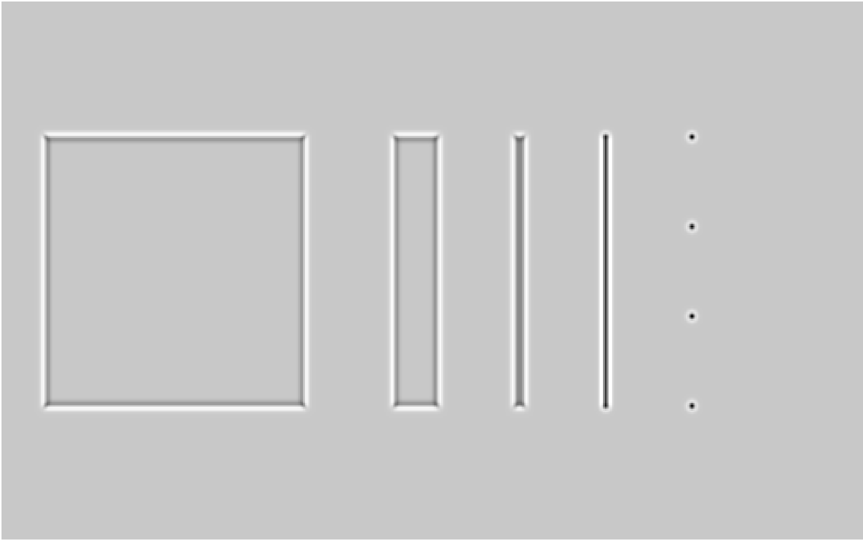
- I is image, G is Gaussian filter and * is convolution operator
- Laplacian of the image smoothed by a Gaussian kernel is identical to the image convolved with the Laplacian of the Gaussian kernel

# Laplacian of Gaussian (LoG)

- Detects edges at zero crossings unlike first derivative which detects edges at maxima/minima
- Can be used to construct an edge detector

Original image

LoG of image

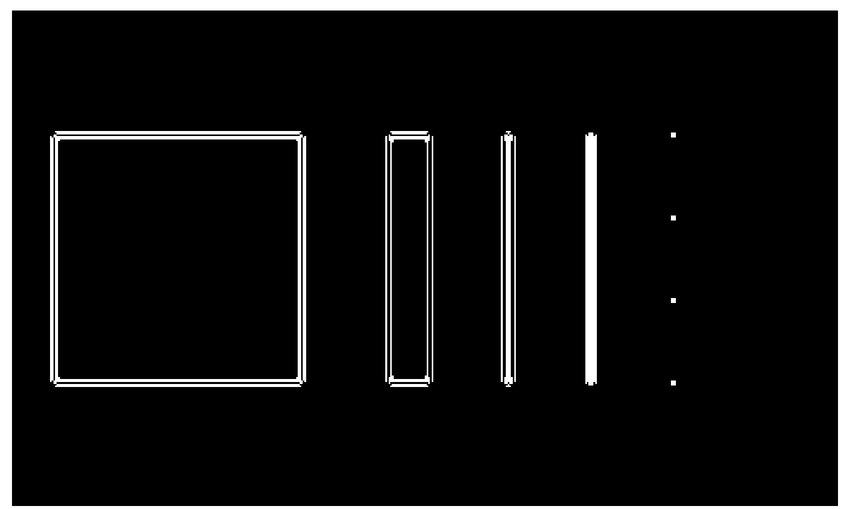


# Laplacian of Gaussian (LoG)

LoG of image

Threshold (t1) LoG of image

Threshold (t2, t2<t1) LoG of image



- Strong (negative) response along the thin line and on the small dots
- It also has medium responses around the edges of the wider objects (negative inside the edge, positive outside)
- Zero crossings are close to where the edges are

# Laplacian of Gaussian (LoG)

- If size of Gaussian filter is too small image may remain noisy
- If size is too large image be blurry and edge detection may not be effective
- Points of zero crossing are edge points

# Example of a LoG approximation

- On filtered image-
  - Set a threshold for zero crossings and retain only those zero crossings that exceed the threshold
  - Strong zero crossings are ones that have a big difference between the positive maximum and the negative minimum on either size of the zero crossing
  - Weak zero crossings are most likely noise, so they are ignored due to the optimum thresholding

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 0 |
| 1 | 2 | 4 | 5 | 5 | 5 | 4 | 2 | 1 |
| 1 | 4 | 5 | 3 | 0 | 3 | 5 | 4 | 1 |
| 2 | 5 | 3 | -12 | -24 | -12 | 3 | 5 | 2 |
| 2 | 5 | 0 | -24 | -40 | -24 | 0 | 5 | 2 |
| 2 | 5 | 3 | -12 | -24 | -12 | 3 | 5 | 2 |
| 1 | 4 | 5 | 3 | 0 | 3 | 5 | 4 | 1 |
| 1 | 2 | 4 | 5 | 5 | 5 | 4 | 2 | 1 |
| 0 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 0 |

LoG filter for $\sigma = 1.4$

# Difference of Gaussians (DoG)

- Useful for enhancing edges in noisy digital images
- Is difference between two smoothed versions of an image obtained by applying two Gaussian kernels of different standard deviations (sigma) on that image
- DoG transformation of an image requires subtracting one highly blurred version of an original image from another less blurred version
- Acts as a band-pass filter that preserves a specific spatial frequency
- Generally serves as an edge enhancement algorithm that delineates the high-frequency content of the image free from noise

# Difference of Gaussian (DoG)

- Is similar to the LoG in its uses
- Can be considered an approximation to the LoG
- DoG is a tunable band-pass filter where both the center frequency and the bandwidth can be tuned separately
- Whereas the LoG has a single parameter that affects both the center frequency and the bandwidth simultaneously
- DoG of image, I is

$$I * G_1 - I * G_2 = I * (G_1 - G_2)$$

# Difference of Gaussian (DoG)


# Unsharp (smoothed) Masking and Highboost Filtering

- Apply averaging mask to blur the original image
- Subtract the blurred image from the original image
- Difference is called the mask
- Add weighted mask to the original

# Steps for image sharpening

- $g_{mask}(x,y) = f(x,y) - f_b(x,y)$

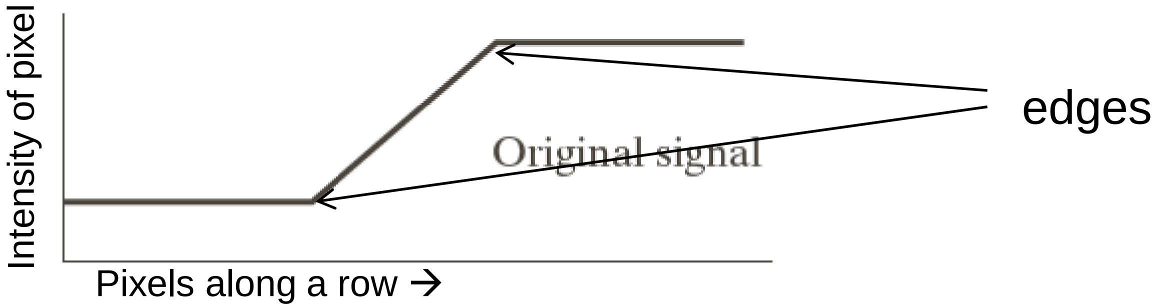
  $f_b$ represents blurred image
- $g(x,y) = f(x,y) + k*g_{mask}(x,y)$, where, $k \geq 0$

  k=1, unsharp masking

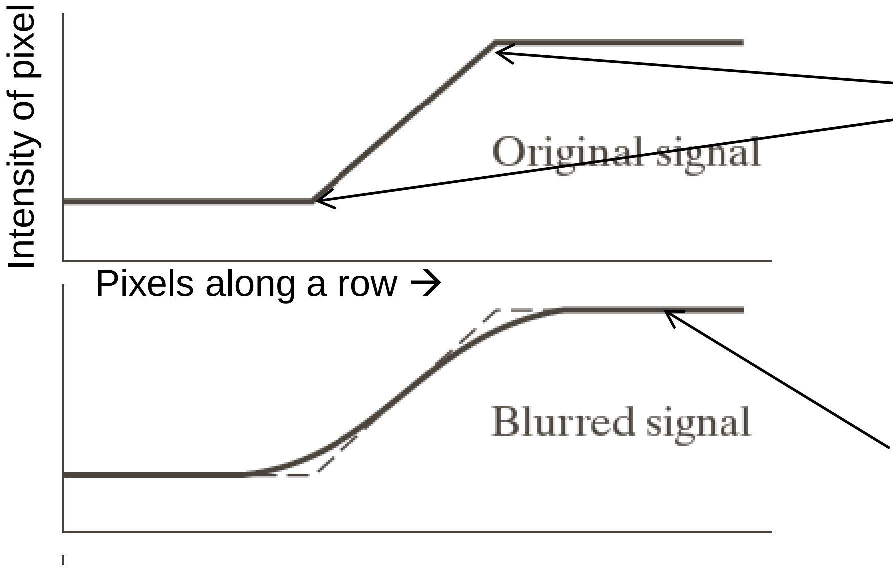
  k>1 , highboost filtering

  k<1 , reduces effect of unsharp mask
- Pixels of g(x,y) can be <0 or >255
- Scale it accordingly

# Highboost Filtering



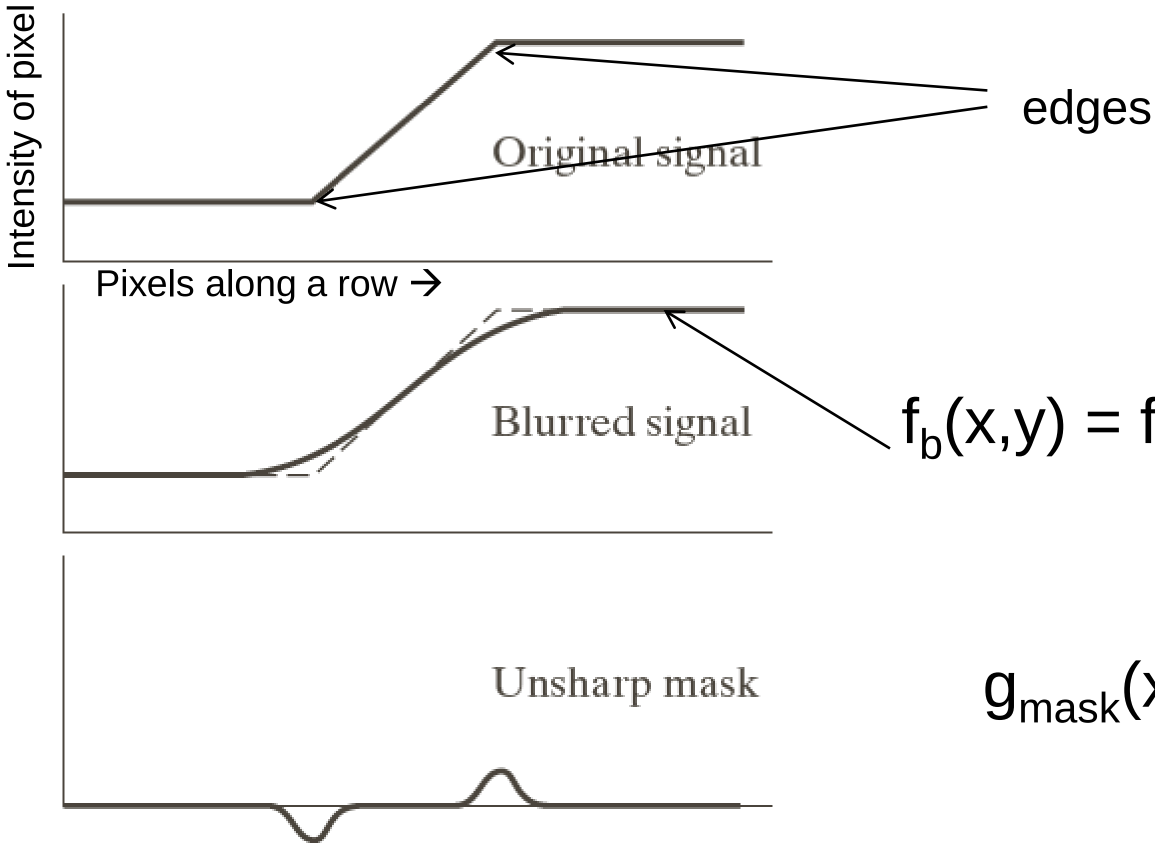
# Highboost Filtering




edges

$f_b(x,y) = f(x,y) * \text{averaging filter}$

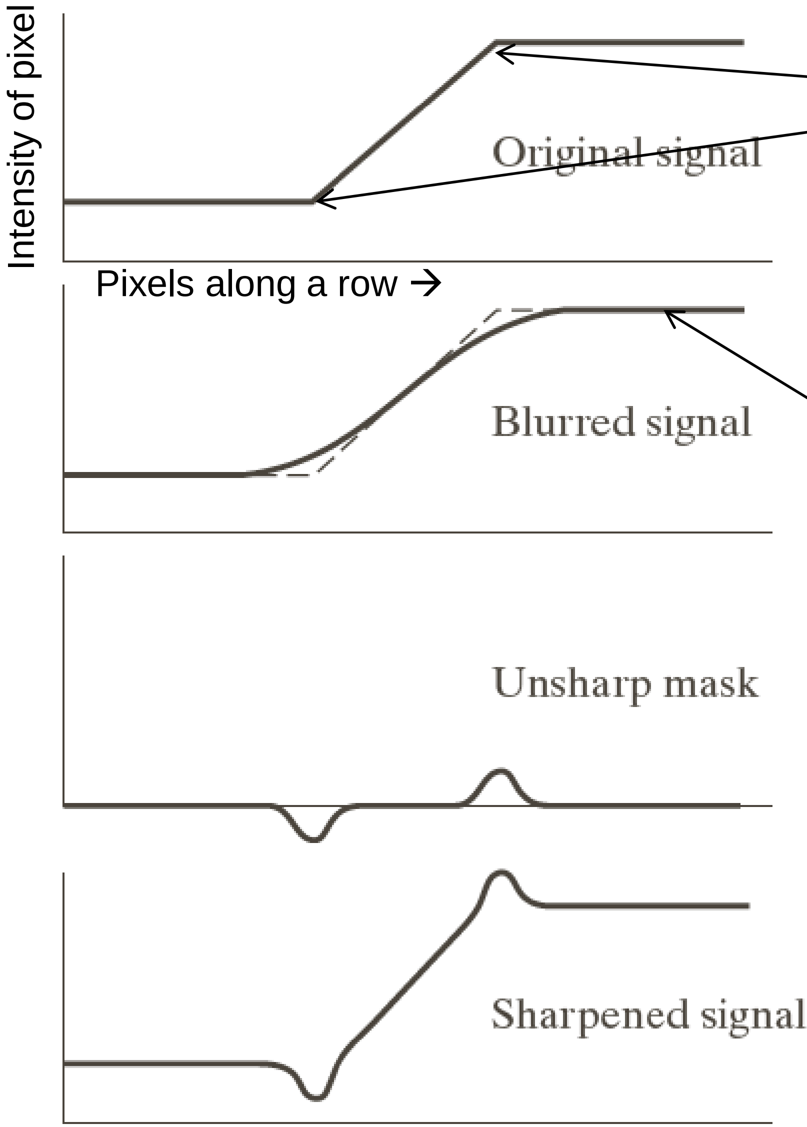
# Highboost Filtering




$f_b(x,y) = f(x,y)$ * averaging filter

$g_{mask}(x,y) = f(x,y) - f_b(x,y)$

# Highboost Filtering

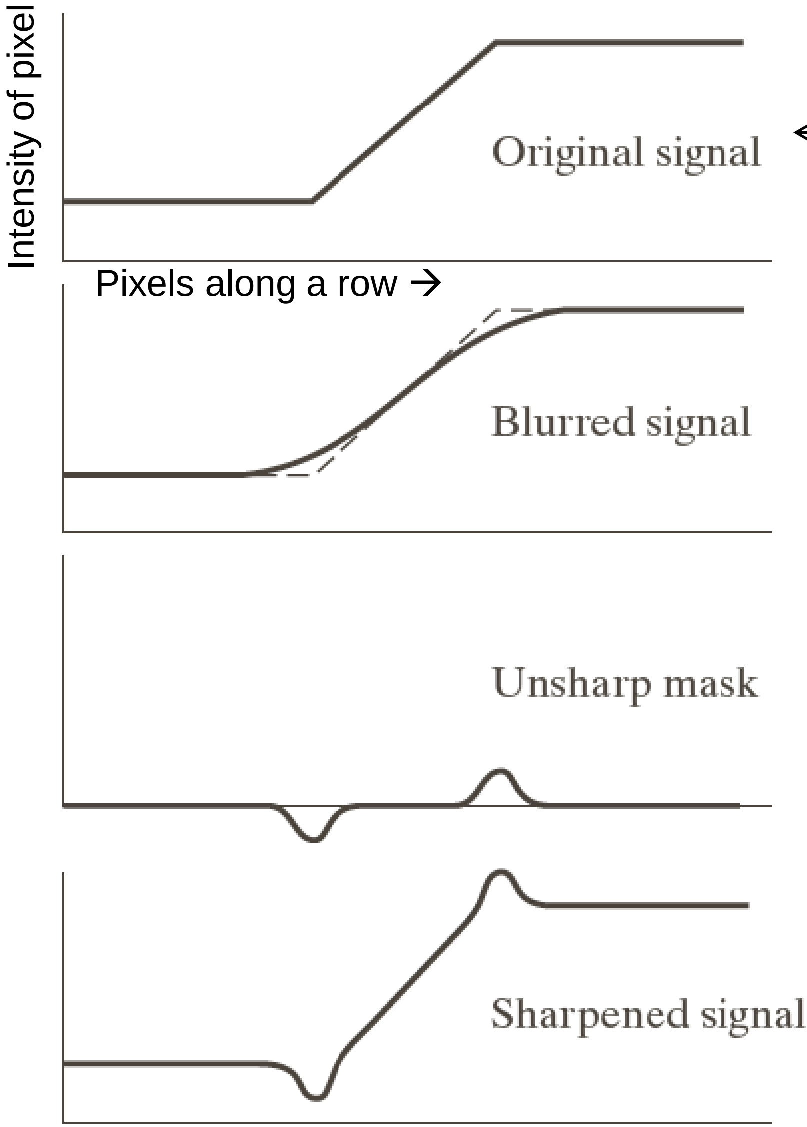



$f_b(x,y) = f(x,y) * \text{averaging filter}$

$g_{mask}(x,y) = f(x,y) - f_b(x,y)$

$g(x,y) = f(x,y) + k*g_{mask}(x,y)$, where, $k \geq 0$

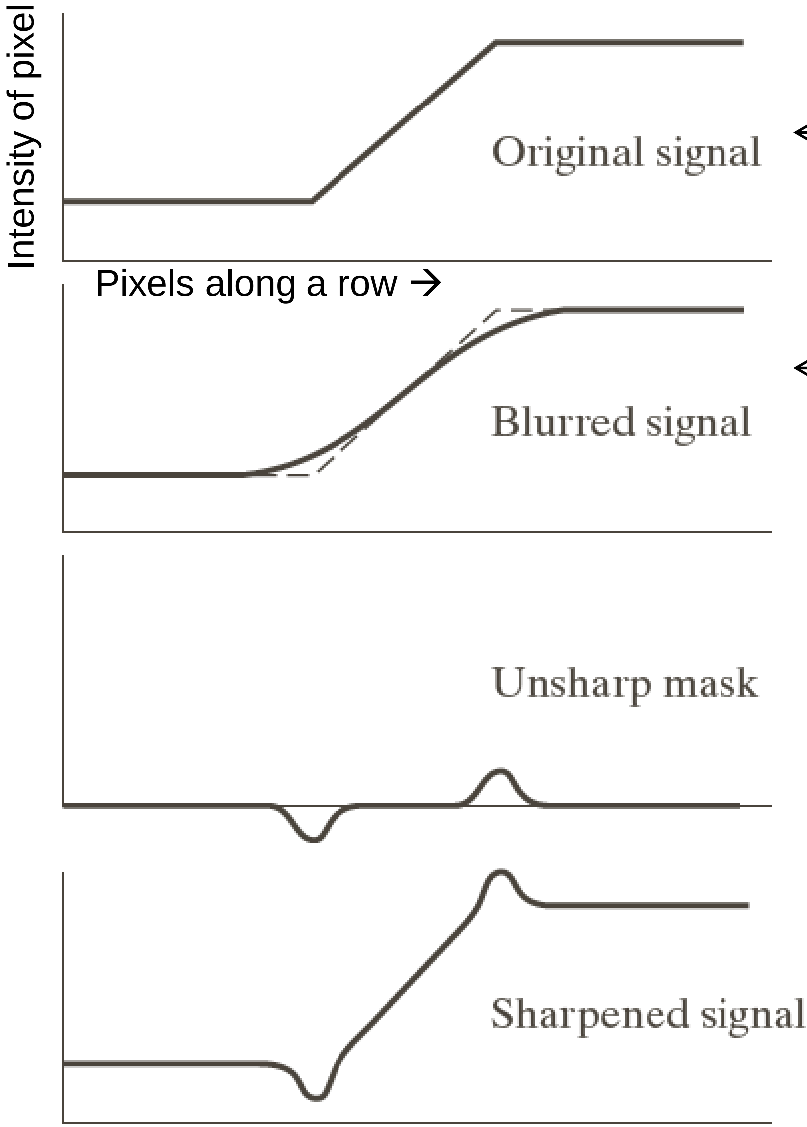
# Highboost Filtering





f(x,y)

# Highboost Filtering

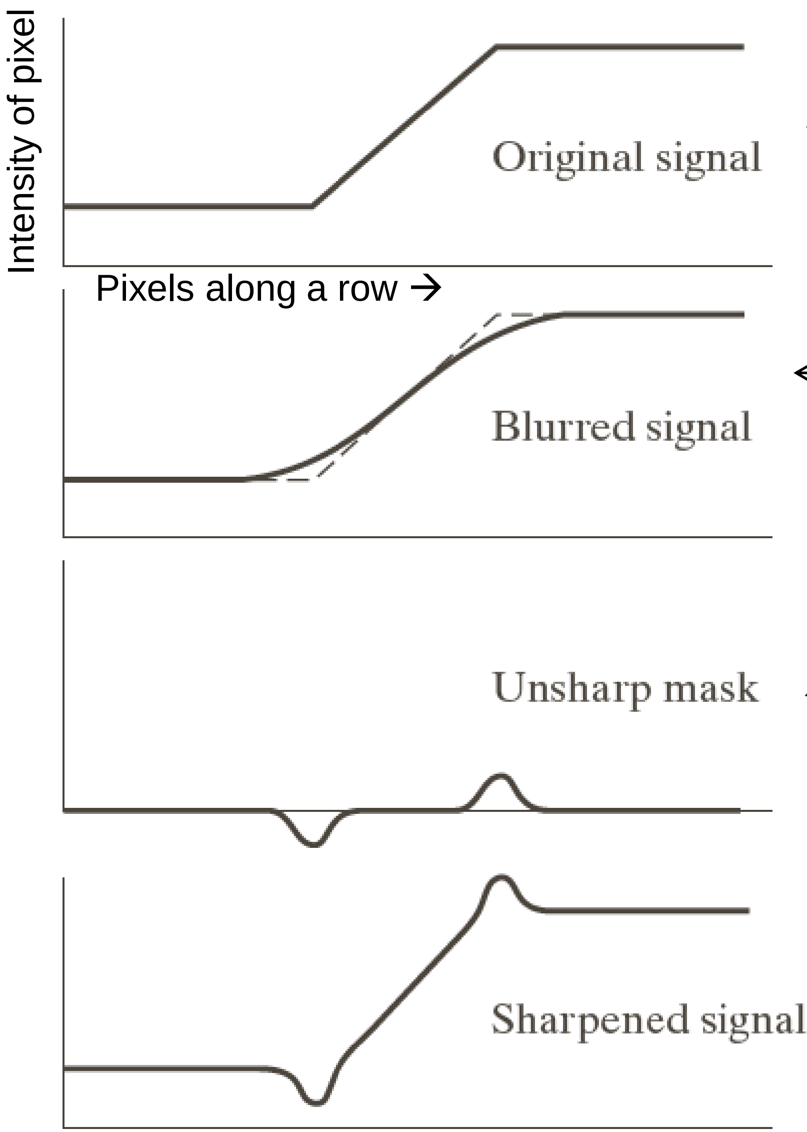




$f(x,y)$

$f_b(x,y) = f(x,y) * h(x,y)$

# Highboost Filtering



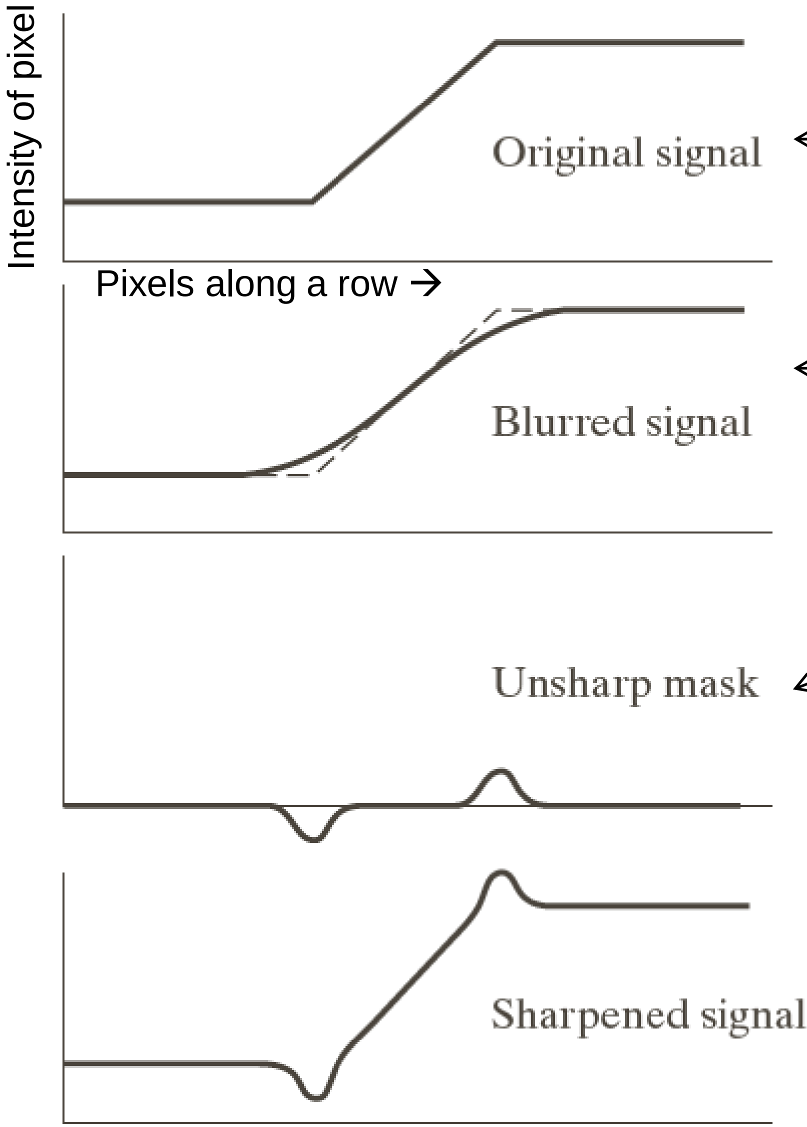


$f(x,y)$

$f_b(x,y) = f(x,y) * h(x,y)$

$g_{mask}(x,y) = f(x,y) - f_b(x,y)$

# Highboost Filtering





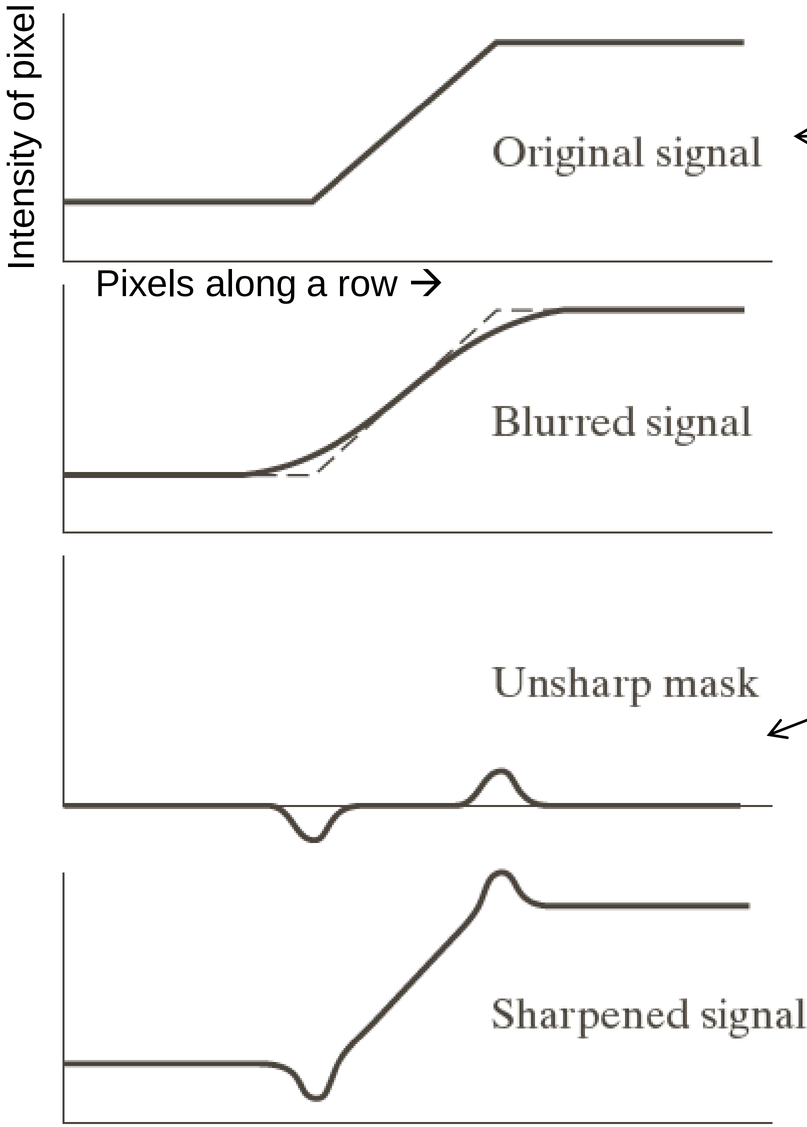
$f(x,y)$

$f_b(x,y) = f(x,y) * h(x,y)$

$g_{mask}(x,y) = f(x,y) - f_b(x,y)$

$f(x,y) + k * g_{mask}(x,y)$, $k=1$

# Highboost Filtering





$f(x,y)$

$f_b(x,y) = f(x,y) * h(x,y)$

$f(x,y) - f_b(x,y)$

$f(x,y) + k * g_{mask}(x,y)$, $k=1$

$f(x,y) + k * g_{mask}(x,y)$, $k=4.5$