Digital Image Processing

Lecture 4
Intensity Transformation

Rahul JAIN

Spring 2022

Outline

- Spatial filtering for Image Processing
  - Spatial filtering
  - Types of spatial filters
    - Smoothing
      - Average filters
      - Median filters
    - Sharpening
      - Laplacian operators
      - Sobel operators
      - Prewitt operators

Spatial filters for image processing

- Spatial filtering: Mask, kernel, template or window
- Frequency domain processing uses the term ‘filter’
- We can accept or reject signal components of certain frequencies by applying a filter (filtering)
  - A low-pass filter is a filter that passes low frequencies
  - A low-pass filter can be used to blue (smoothing) the image
  - A high-pass filter is a filter that passes high frequencies
  - The use of high-pass filter can help to sharpen the image

Spatial filter

- Spatial filter (kernel) applied to a pixel position (pixel and its neighborhood)

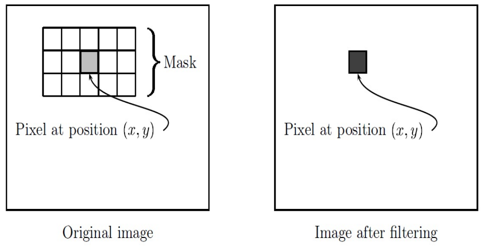


Figure 3.1: Using a spatial mask on an image

Image Source: An Introduction to Digital Image Processing with Matlab (Alasdair McAndrew)

  - A predefined operation is applied to image pixels within a neighborhood

- If the image processing procedure (operation) is linear or nonlinear, the filter is called a linear/nonlinear spatial filter

Linear equation : $y = mx + c$

Nonlinear equation : $ax^2 + by^2 = c$

Linear spatial filtering

- Linear spatial filtering of an image f with a filter w can be given as:

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

where (x,y) pixel position, $(-a,a)$ and $(-b,b)$ are neighborhood positions

$w(s,t)$ is a spatial filter

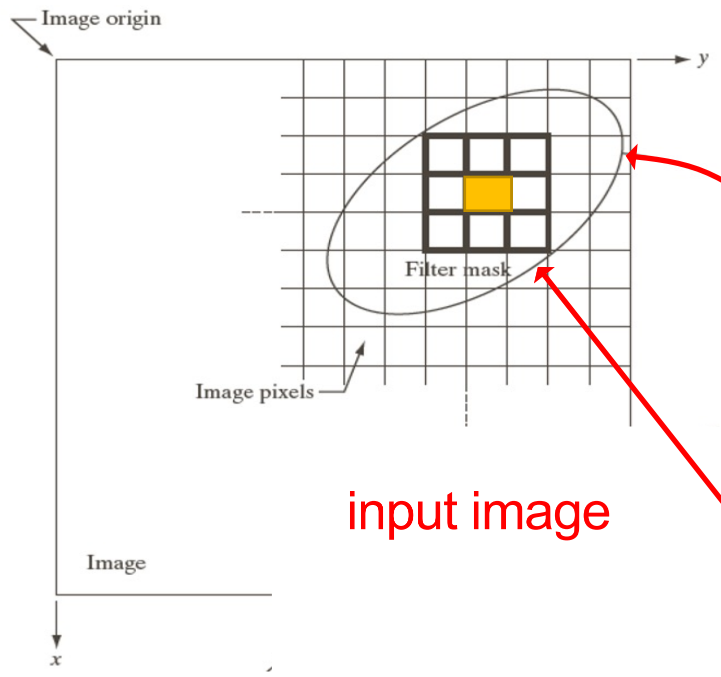
$f(x,y)$ is pixel value at (x,y) coordinate (that is the center of the neighborhood)

$g(x,y)$ is new pixel value (result of the filtering operation)

- We can obtain a filtered image by applying a filter to each pixel position of an image in a sliding window manner

Linear spatial filtering

- Example of linear spatial filtering using a filter of size 3×3 :



input image

3×3 filter w

$w(-1,-1)$	$w(-1,0)$	$w(-1,1)$
$w(0,-1)$	$w(0,0)$	$w(0,1)$
$w(1,-1)$	$w(1,0)$	$w(1,1)$

3×3 neighborhood

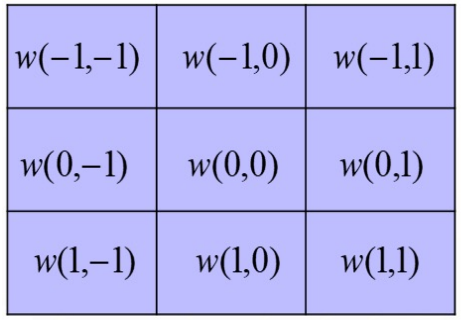
$f(x-1,y-1)$	$f(x-1,y)$	$f(x-1,y+1)$
$f(x,y-1)$	$f(x,y)$	$f(x,y+1)$
$f(x+1,y-1)$	$f(x+1,y)$	$f(x+1,y+1)$

A filter is implemented by multiplying all elements in the mask by corresponding elements in the neighborhood, and summing up all these individual products

Linear spatial filtering

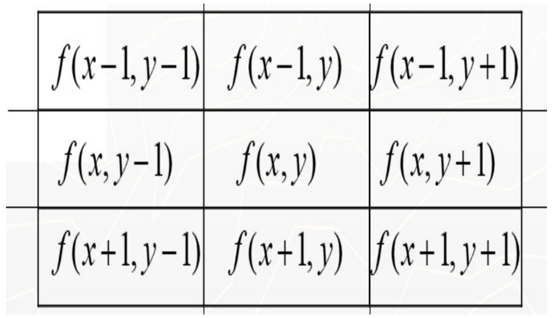
- The operation can be expressed as:

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$



$w(-1,-1)$	$w(-1,0)$	$w(-1,1)$
$w(0,-1)$	$w(0,0)$	$w(0,1)$
$w(1,-1)$	$w(1,0)$	$w(1,1)$

$\times$



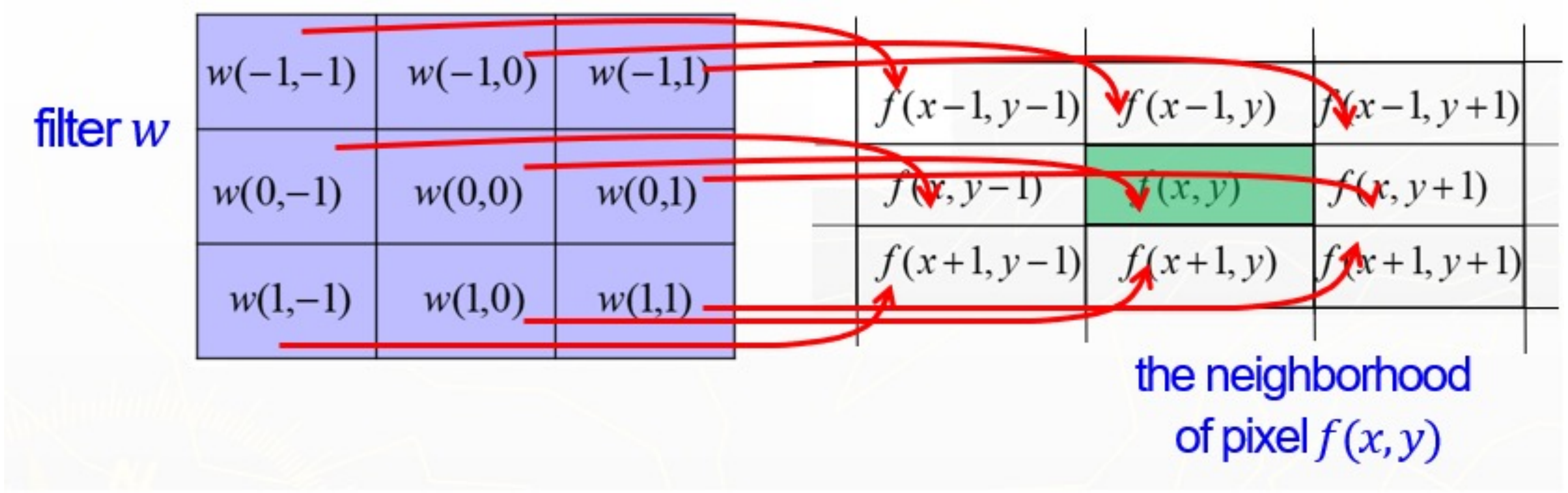
$f(x-1,y-1)$	$f(x-1,y)$	$f(x-1,y+1)$
$f(x,y-1)$	$f(x,y)$	$f(x,y+1)$
$f(x+1,y-1)$	$f(x+1,y)$	$f(x+1,y+1)$

$$g(x,y) = w(-1,-1) f(x-1, y-1) + w(-1,0) f(x-1, y) + w(-1,1) f(x-1, y+1) + w(0,-1)(x, y-1) \ + w(0,0)(x,y) + w(0,1)(x, y+1) + w(1,-1)(x+1, y-1) + w(1,0)(x+1, y) + w(1,1)(x+1, y+1)$$

The sum of the products of the filter w and the neighborhood enclosed by w is the output $g(x,y)$ at any pixel position

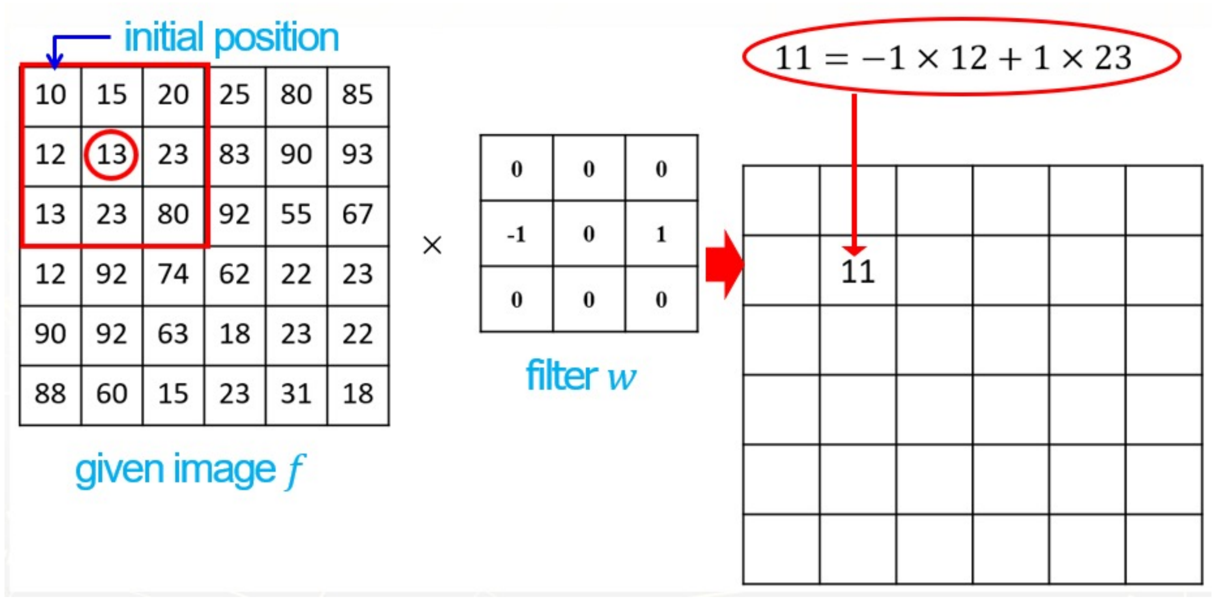
Linear spatial filtering

The sum of the products of the filter w and the neighborhood enclosed by w is the output $g(x, y)$ at any pixel position



Linear spatial filtering

- The linear spatial filtering technique is demonstrated in the following example:
  - Filtering must begin at the first pixel (initial) position
  - At this pixel position, the linear spatial filtering operation calculates a new pixel value



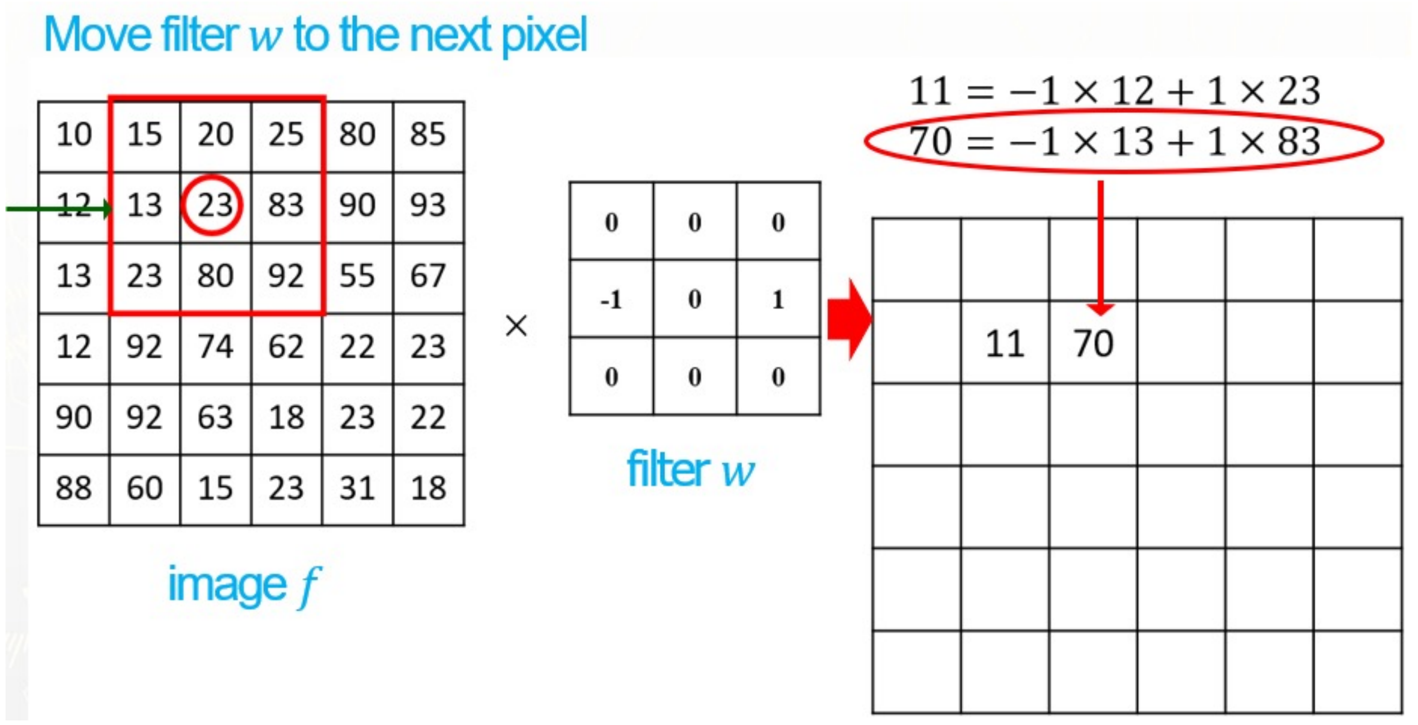
10	15	20	25	80	85
12	13	23	83	90	93
13	23	80	92	55	67
12	92	74	62	22	23
90	92	63	18	23	22
88	60	15	23	31	18

0	0	0
-1	0	1
0	0	0

$$11 = -1 \times 12 + 1 \times 23$$

Linear spatial filtering

- Slide the filter w to the next pixel position
- Perform the filtering operation in the same manner to obtain a new pixel value for that pixel position



Move filter w to the next pixel
$11 = -1 \times 12 + 1 \times 23$
$70 = -1 \times 13 + 1 \times 83$
image f
filter w
$\times$

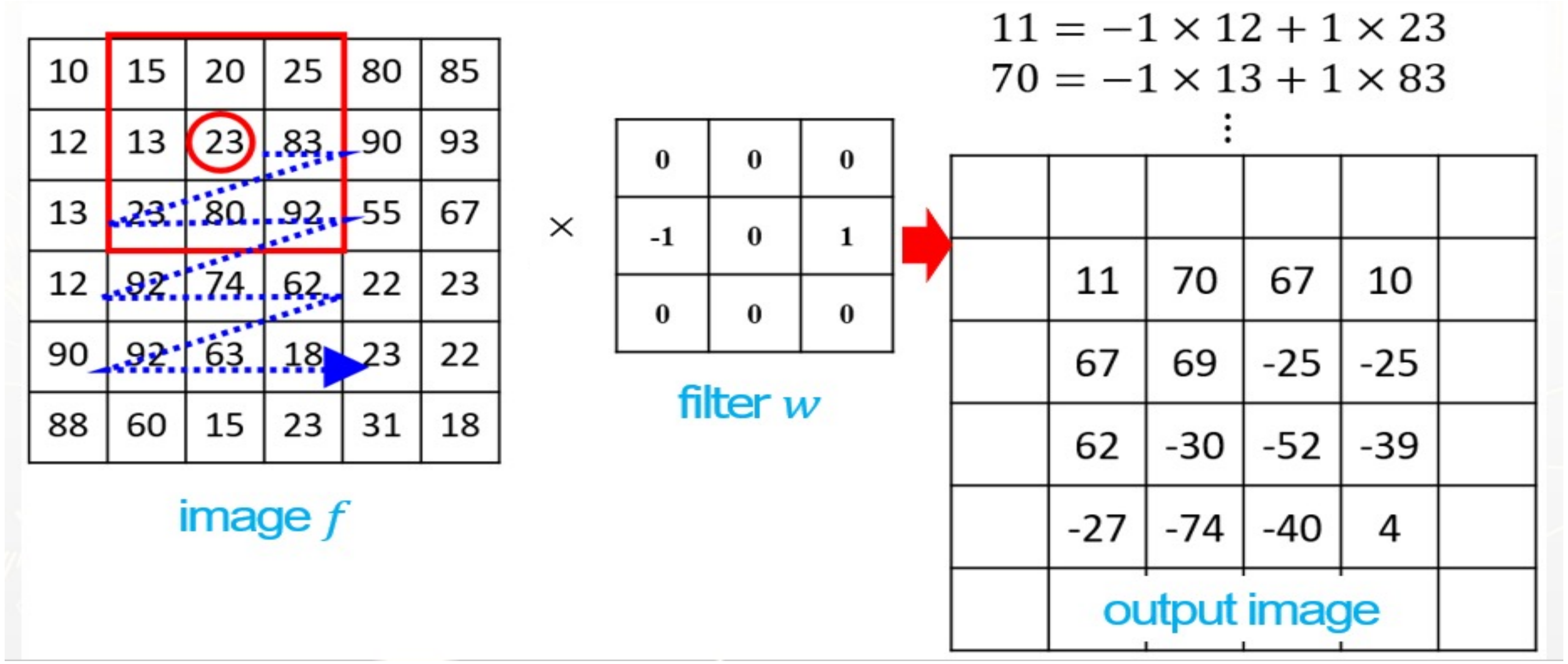
10	15	20	25	80	85
12	13	23	83	90	93
13	23	80	92	55	67
12	92	74	62	22	23
90	92	63	18	23	22
88	60	15	23	31	18

0	0	0
-1	0	1
0	0	0

	11	70			

Linear spatial filtering

- After then, the filtering procedure is repeated for all x and y values, ensuring that the filter w is facing every pixel in the image f
- Obtaining the final image



10	15	20	25	80	85
12	13	23	83	90	93
13	23	80	92	55	67
12	92	74	62	22	23
90	92	63	18	23	22
88	60	15	23	31	18

image f

$\times$

0	0	0
-1	0	1
0	0	0

filter w

$$11 = -1 \times 12 + 1 \times 23$$
$$70 = -1 \times 13 + 1 \times 83$$
$$\vdots$$

	11	70	67	10	
	67	69	-25	-25	
	62	-30	-52	-39	
	-27	-74	-40	4	

output image

Linear spatial filtering

The sum of the products of the filter w and the neighborhood enclosed by w is the output $g(x, y)$ at any pixel position

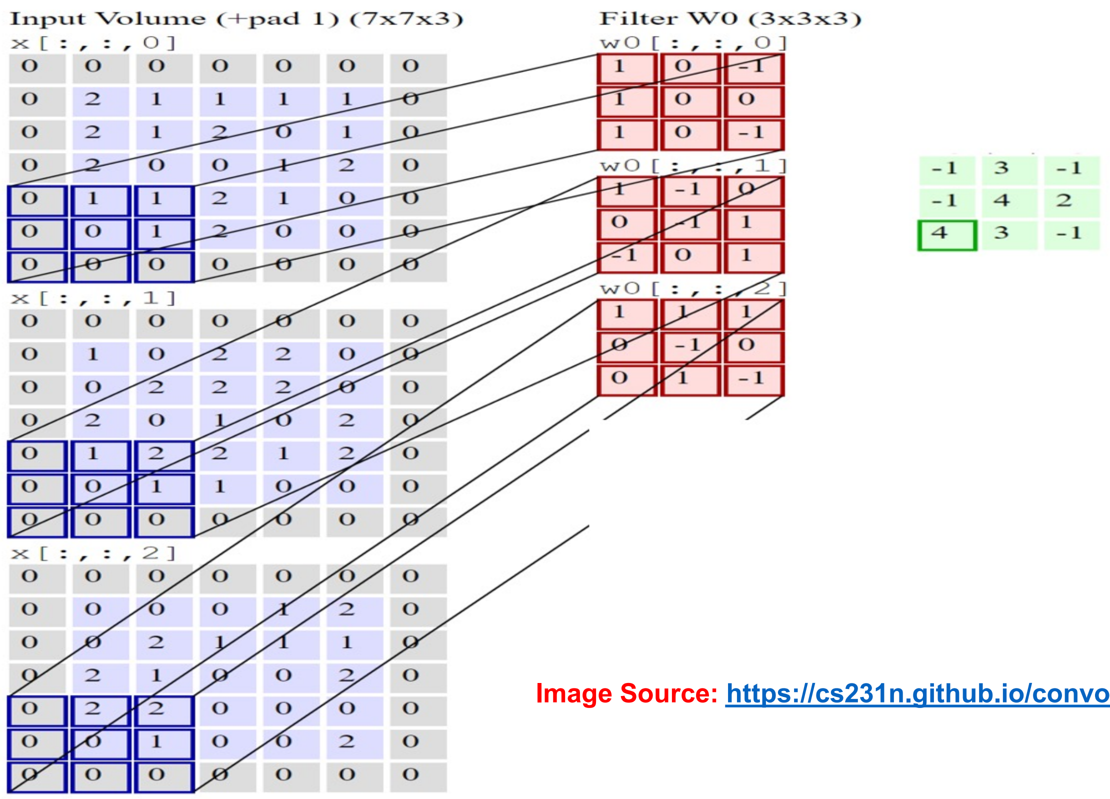


Image Source: https://cs231n.github.io/convolutional-networks/

Spatial Correlation and Convolution

- There are two concepts: Correlation and Convolution
- **Correlation:**
  - Value of an output pixel is computed as a weighted sum of neighboring pixels
  - Process of sliding a filter mask (correlation kernel) over an image and computing the sum of individual products at each pixel location

Correlation operation on an image f using a filter w can be shown as:

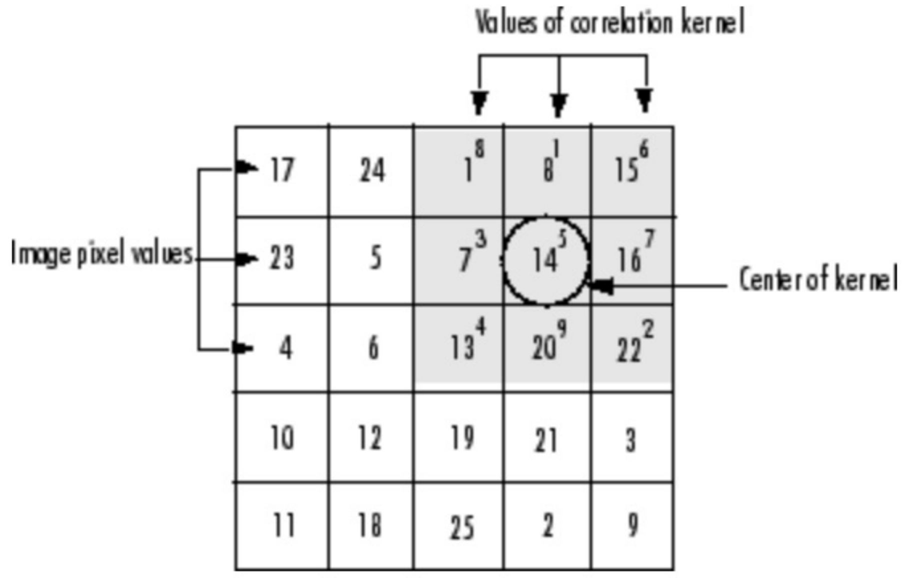
$$g(x, y)\ f(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) f(x + s, y + t)$$

Spatial Correlation and Convolution

- Correlation
  - It is can be referred as measurement of the similarity between two signals/sequences

$$1\cdot 8+8\cdot 1+15\cdot 6+7\cdot 3+14\cdot 5+16\cdot 7+13\cdot 4+20\cdot 9+22\cdot 2=585$$

Computing the (2,4) Output of Correlation



For example, suppose the image is

```
A = [17  24   1   8  15
     23   5   7  14  16
      4   6  13  20  22
     10  12  19  21   3
     11  18  25   2   9]
```

Image

the correlation kernel is

```
h = [8   1   6
     3   5   7
     4   9   2]
```

Kernel

Image Source: https://au.mathworks.com/help/images/what-is-image-filtering-in-the-spatial-domain.html#f16-21080

Spatial Correlation and Convolution

- **Convolution:**
  - Convolution process rotates the correlation kernel (filter) 180 degrees about its center element to create a convolution kernel
  - Slide the center element of the convolution kernel so that it lies on top of the element of image f
  - Multiply each weight in the rotated convolution kernel by the pixel of image f underneath.
  - Sum the individual products

convolving an image f using a filter w :

$$g(x,y)f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f \overset{\text{flip } f}{\boxed{(x - s, y - t)}}$$

Spatial Correlation and Convolution

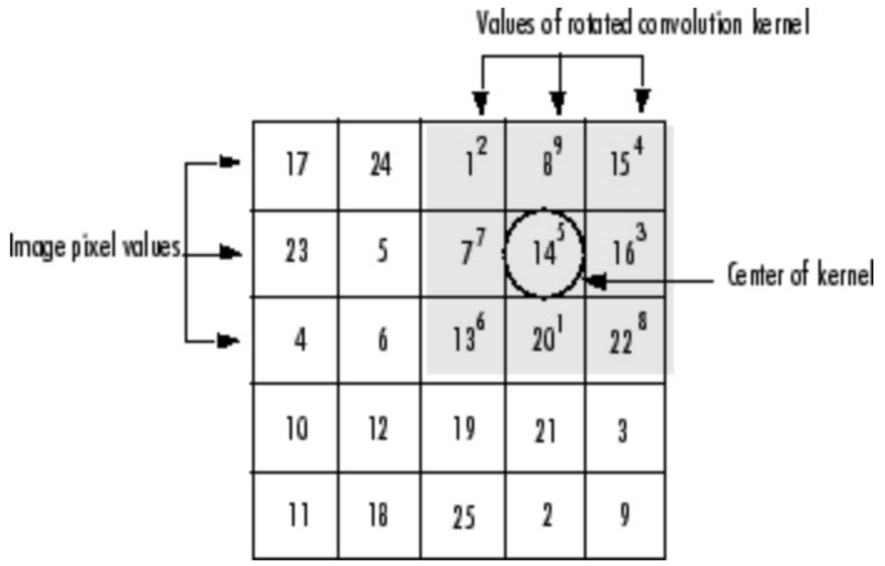
- Convolution is measurement of effect of one signal on the other signal

Hence the (2,4) output pixel is

$$1 \cdot 2 + 8 \cdot 9 + 15 \cdot 4 + 7 \cdot 7 + 14 \cdot 5 + 16 \cdot 3 + 13 \cdot 6 + 20 \cdot 1 + 22 \cdot 8 = 575$$

Shown in the following figure.

Computing the (2,4) Output of Convolution



For example, suppose the image is

```
A = [17  24   1   8  15
     23   5   7  14  16
      4   6  13  20  22
     10  12  19  21   3
     11  18  25   2   9]
```

Image

the correlation kernel is

```
h = [8   1   6
     3   5   7
     4   9   2]
```

Kernel

Image Source: https://au.mathworks.com/help/images/what-is-image-filtering-in-the-spatial-domain.html#f16-21080

Size of a filter mask

- For a filter of a size $m \times n$,

  $m = 2a + 1$, $n = 2b + 1$

  a and b are positive integers

- Normally odd numbers are used as filter size

  3×3 is commonly used filter size

$f(x, y)$

2	3	4	5	6
17	0	0	0	7
16	0	1	0	8
15	0	0	0	9
14	13	12	11	10

w

1	2	3
4	5	6
7	8	9

Filtering

- Edge of the image:

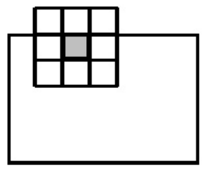
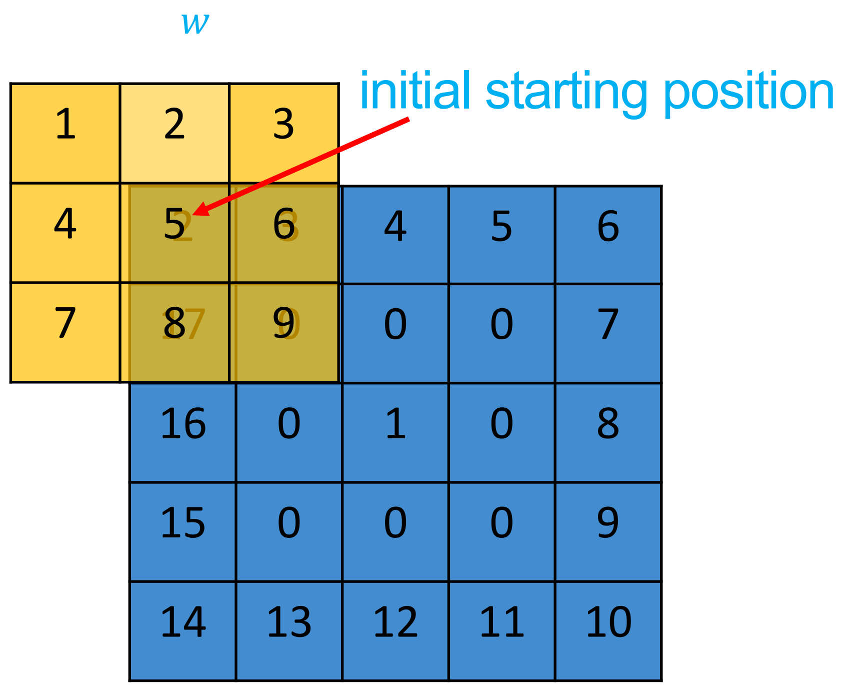

A mask at the edge of an image

Image *f*(x,y)

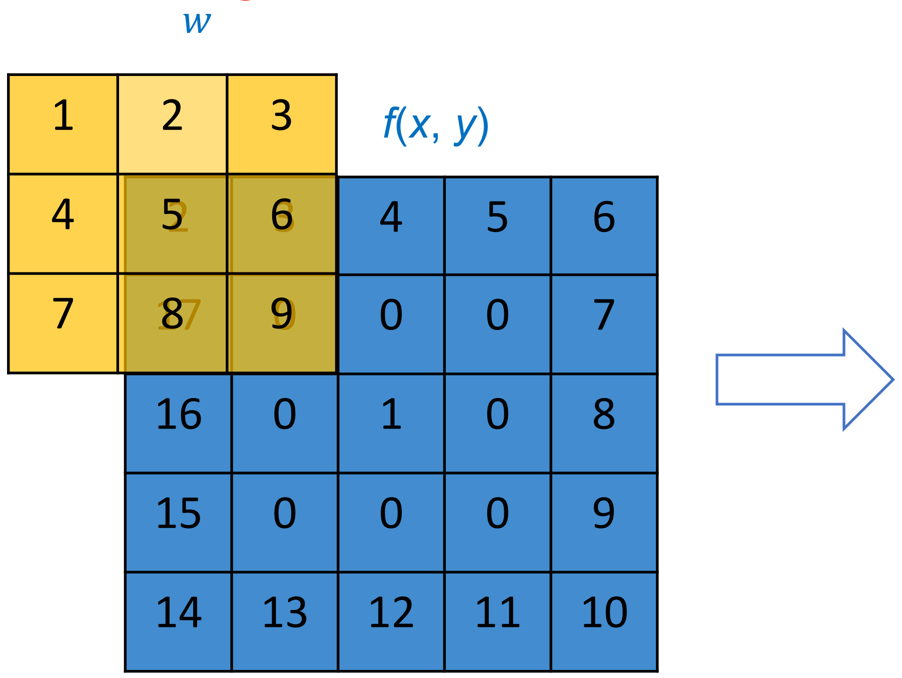
2	3	4	5	6
17	0	0	0	7
16	0	1	0	8
15	0	0	0	9
14	13	12	11	10



- There is an obviously problem in applying a filter – what happens at the edge of the image, where the mask kernel is partially falls outside the image

Padding

- Edge of the image:



padded f

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	2	3	4	5	6	0	0
0	0	17	0	0	0	7	0	0
0	0	16	0	0	0	8	0	0
0	0	15	0	0	0	9	0	0
0	0	14	13	12	11	10	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

As m=n=3, we can pad the image with 2 rows and 2 columns of 0s around f

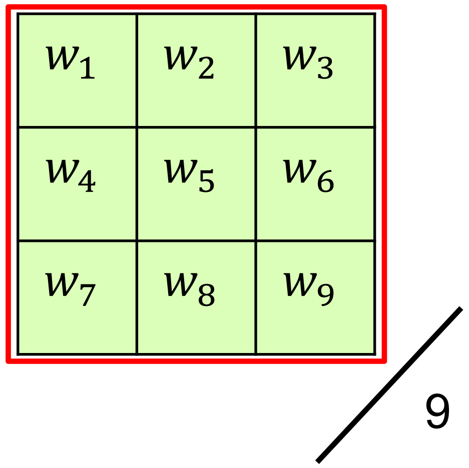
- We can assume that all necessary values are outside the image are zero.
- This gives us all values to work with and will return an output image of the same size as the original, but may have the effect of introducing unwanted artifacts (for example, edges) around the image

Spatial filter masks generation

- To generate an $m \times n$ linear spatial filter requires mn mask coefficients
- These coefficients values are selected based the operation of filter
  - Linear filtering is implemented using a sum of products
  - Suppose if we want to replace the pixels in an image by the average intensity of 3 × 3 neighborhood centered on (x, y) in the image is the sum of the nine intensity values in the 3 × 3 neighborhood centered on (x, y) divided by 9

$$\boldsymbol{R = w_1 z_1 + w_2 z_2 + w_3 z_3 + \cdots + w_9 z_9 = \frac{1}{9} \sum_{k=1}^{9} w_k z_k}$$

where z_k is pixel values in the neighborhood



Smoothing

- The output of a smoothing (linear spatial filter) is simply the average of the pixels contained in the neighborhood of the filter mask
- These filters are called averaging filters
- Smoothing filters are also referred as lowpass filter
- Smoothing filters are used for blurring and for noise reduction
  - Blurring is useful in preprocessing tasks such as removal of small details from an image
  - Bridging the gap between small gaps and lines

Averaging filters

- A 3×3 averaging filter mask is used to calculate the standard average of the pixels under the mask

$$\frac{1}{9} \times \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

$$R = \frac{1}{9} \sum_{k=1}^{9} w_k \, z_k$$

- In most cases, an equivalent filter with all 1s as coefficients is used, and the result is divided by total number of pixels
- It is computationally more efficient

Averaging filters

- This mask yields a weighted average
- Pixel values are multiplied by different coefficients
- In this example, the pixel at the center of mask is multiplied by a higher value than any other, thus giving more importance in the calculation of the average

$$\frac{1}{16} \times \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

$$R = \frac{1}{16} \sum_{k=1}^{9} w_k z_k$$

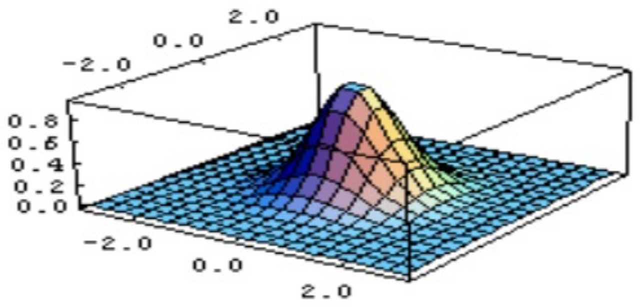
- This process also results in smooth image

Gaussian filter

- This filter is an approximation of a Gaussian function:

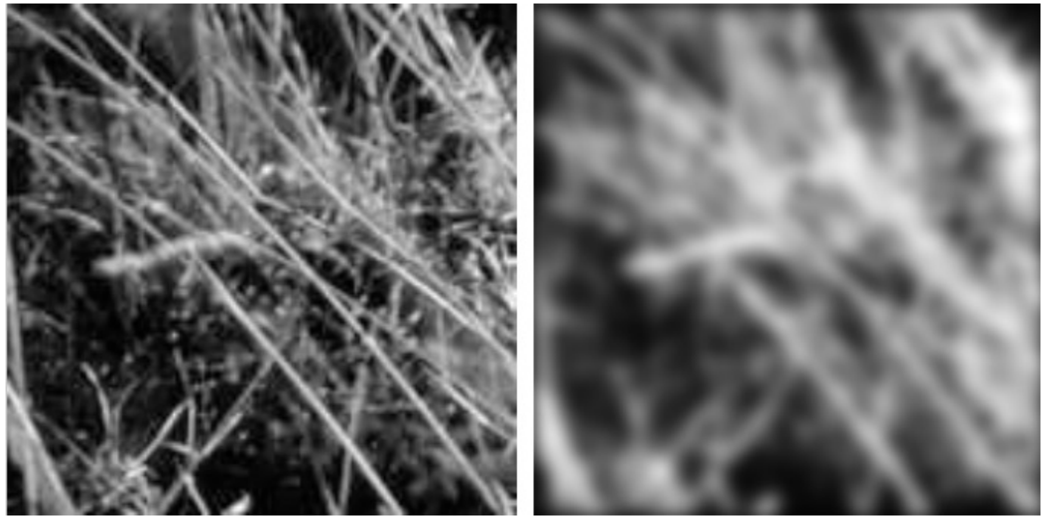
$$G_{2D}(x, y, \sigma) = \frac{1}{2\pi\sigma^2} exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

exp (exponential functions: e – 2.7182818…)
σ determines the width of the Gaussian kernel
x is the distance from the origin in the horizontal axis
y is the distance from the origin in the vertical axis



$\frac{1}{16} \times$

1	2	1
2	4	2
1	2	1

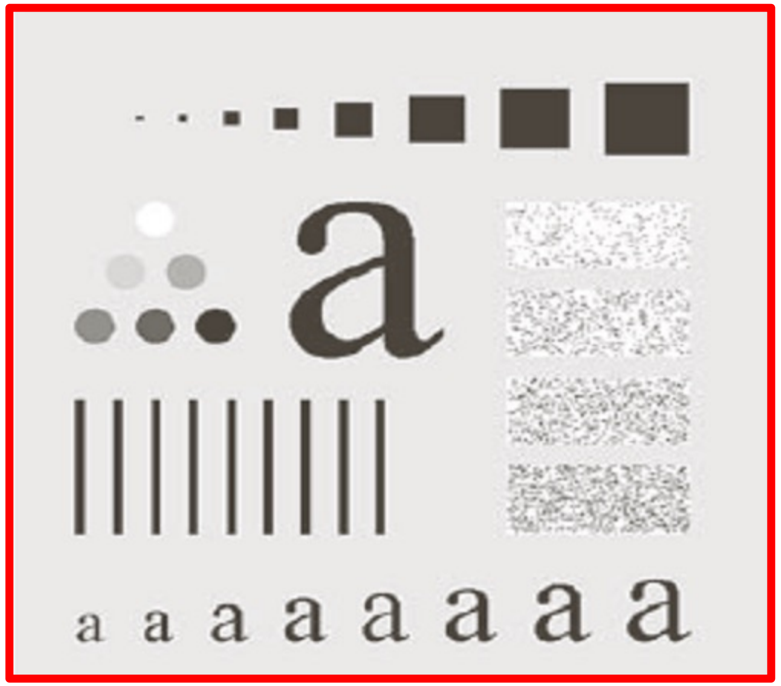


Smoothed image by a Gaussian filter

What happens if you increase $\boldsymbol{\sigma}$?

Effects of Averaging Filtering

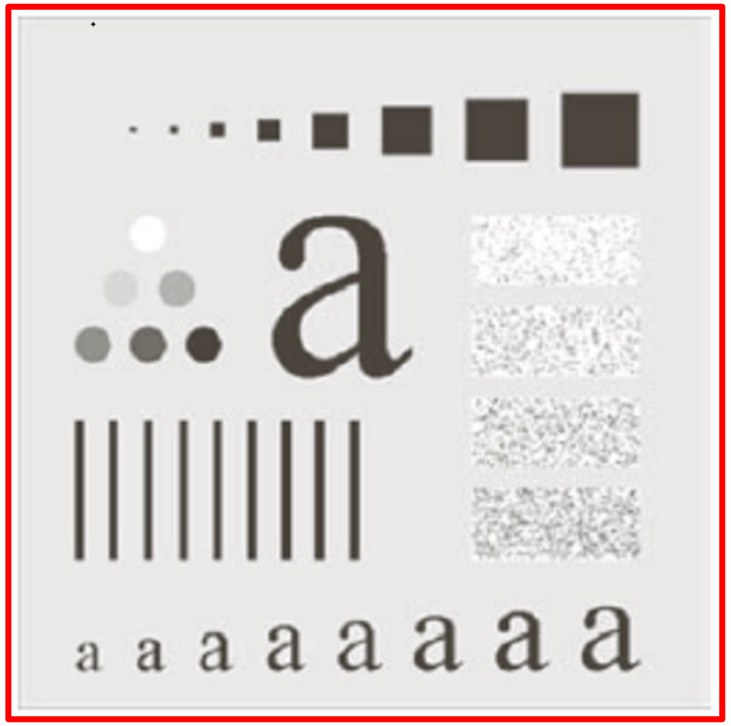
- The image of size 500×500 with some different sizes of
- Black squares
- Letters
- Fine grain noise
- Lines
- Circles



- Using an average filter with different filter sizes (m=3,5,9,15, and 35 pixels), we can Smooth this image.

Effects of Averaging Filtering

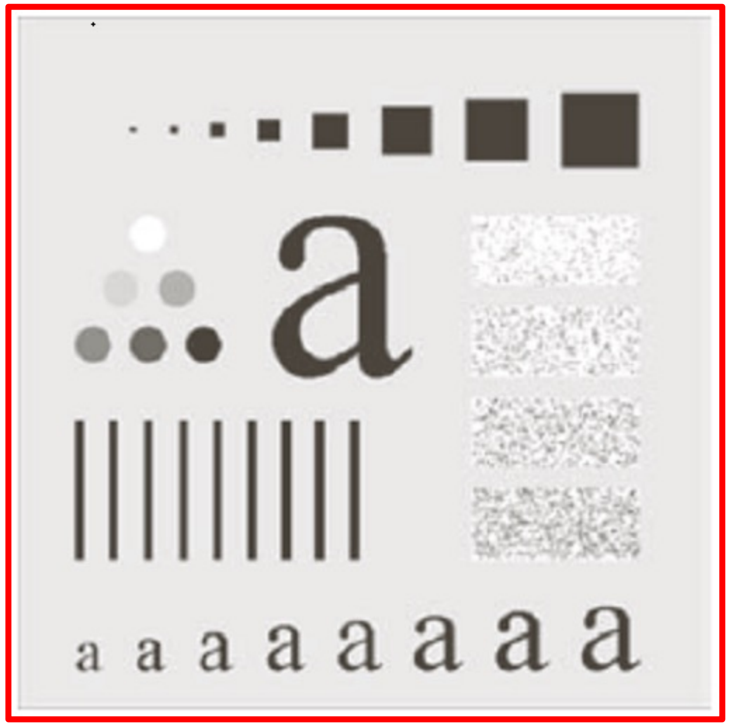
- A slight blurring throughout the whole image
- In the noise, there is some blurring



$m = 3$

Effects of Averaging Filtering

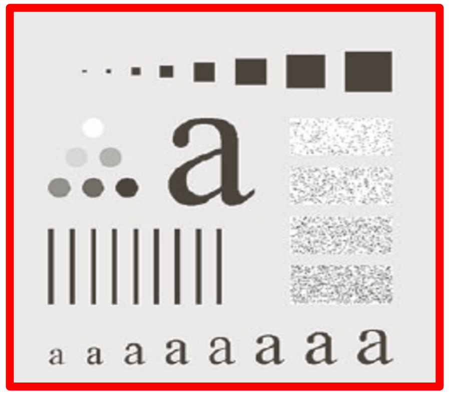
- Almost similar, with a minor increase in blurring

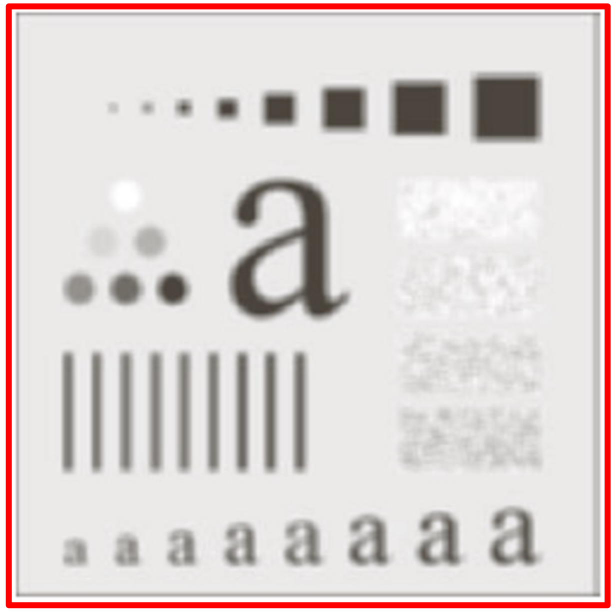


$m = 5$

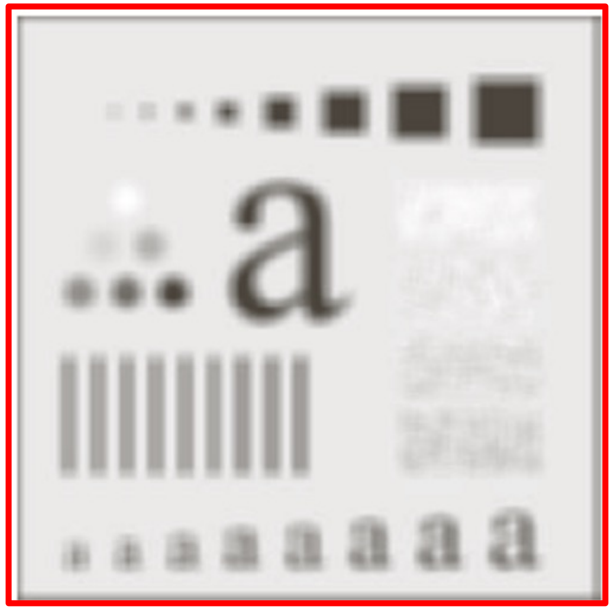
Effects of Averaging Filtering

- There is more blurring in the final images than in the earlier ones
- Smoothing can be used as a pre-processing step before object detection
- It allows blending of smaller elements into the background
- Enable large objects to become more detectable

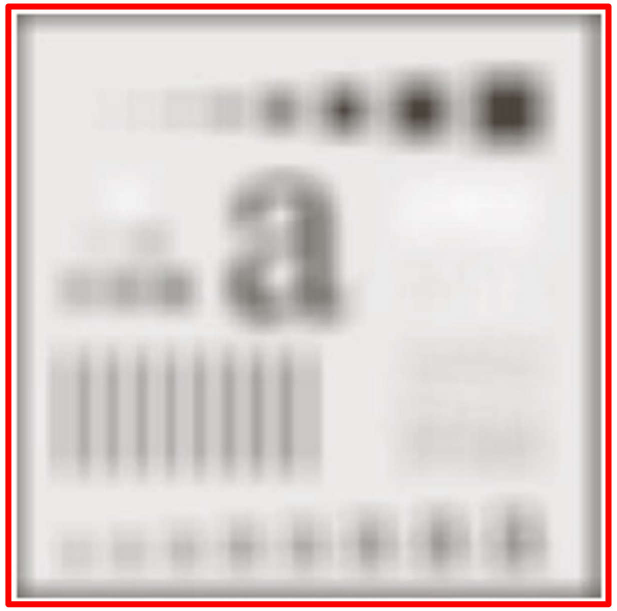




$m = 9$



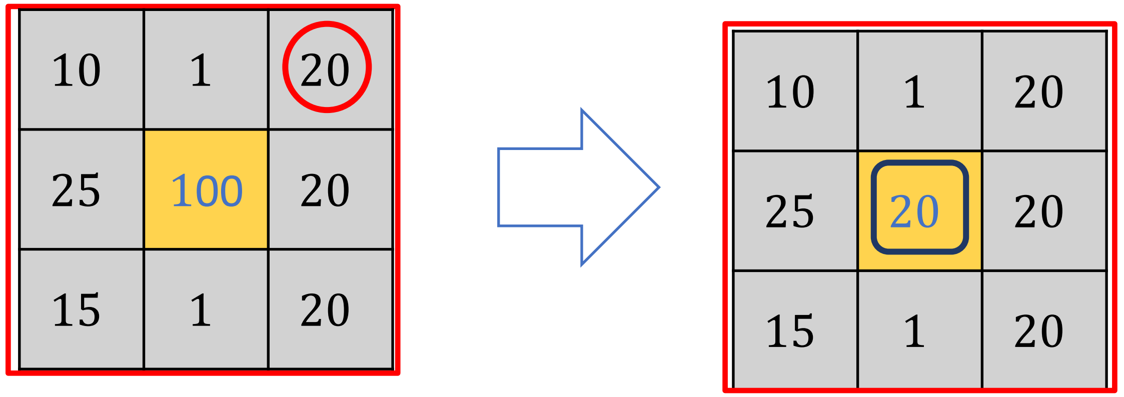
$m = 15$



$m = 35$

Median filter

- Median filter:



- Suppose at a pixel position (*x*, *y*), a 3×3 neighborhood has value

  (10, 1, 20, 25, 100, 20, 15, 1, 20)

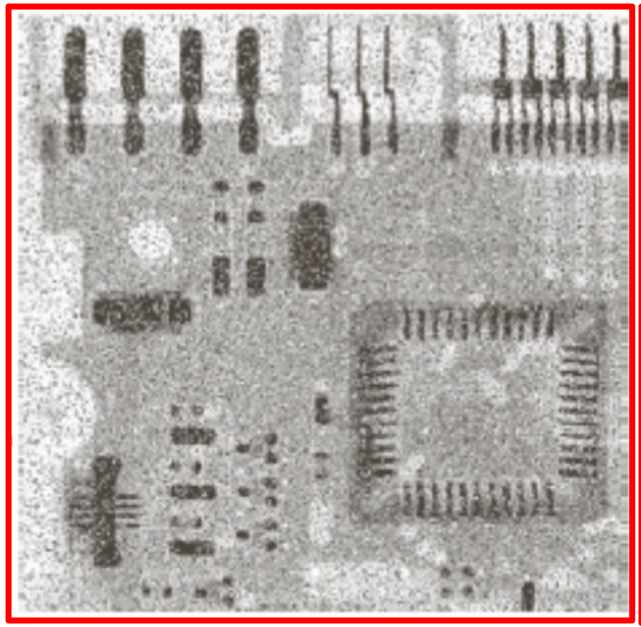
- We can sort these values in ascending order as:

  (1, 1, 10, 15, 20, 20, 20, 25, 100)

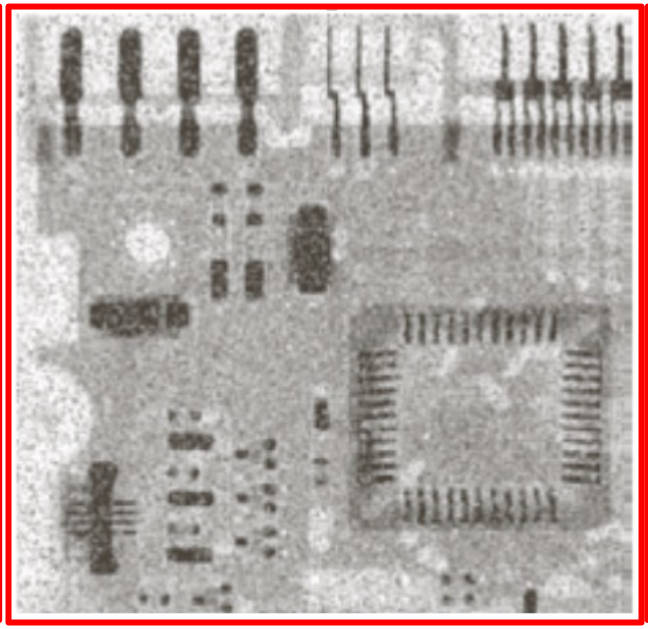
- Replace the value of the pixel $f(x,y)$ by 20

Noise reduction using median filter

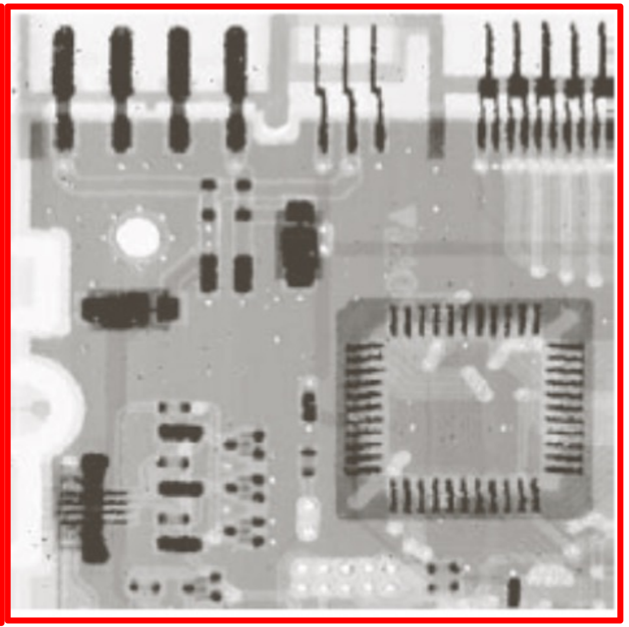
- This filter is particularly effective for removing salt-and-pepper noise, the noise appears as white and black dots



Original image with noise

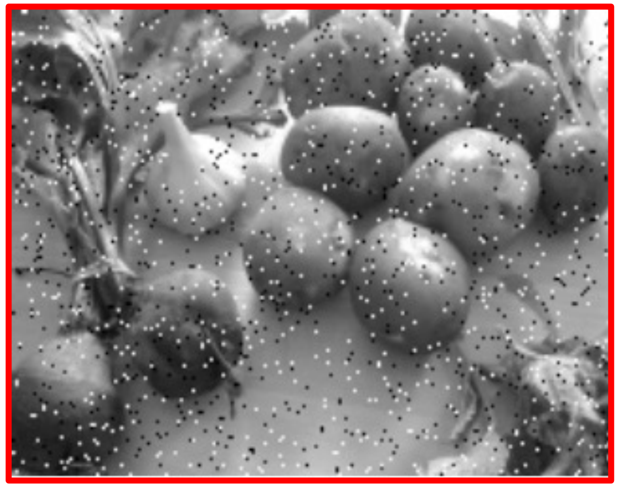


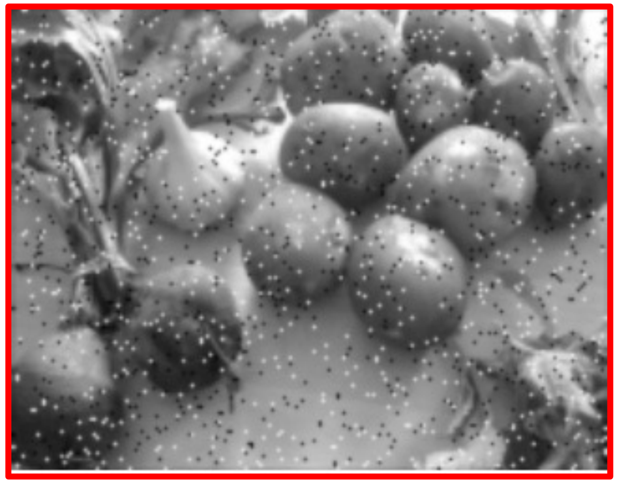
Resulting image with a 3×3 averaging mask

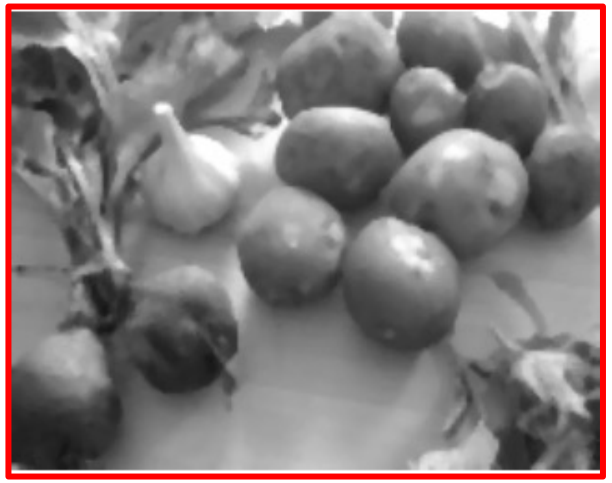


Resulting image with a 3×3 median mask

Noise reduction using median filter


Original image with noise


Resulting image with a 3×3 averaging mask


Resulting image with a 3×3 median mask

Sharpening spatial filters

- Sharpening is a technique for emphasizing transections in intensity by enhancing edges, other boundaries, and discontinuities
- Spatial differentiation is a method for sharpening images
- Sharpening filters are based on first- and second- order derivatives

Sharpening spatial filters

- The derivatives of a digital function f are defined in terms of differences

- The first-order derivative of a one-dimensional function $f(x)$:

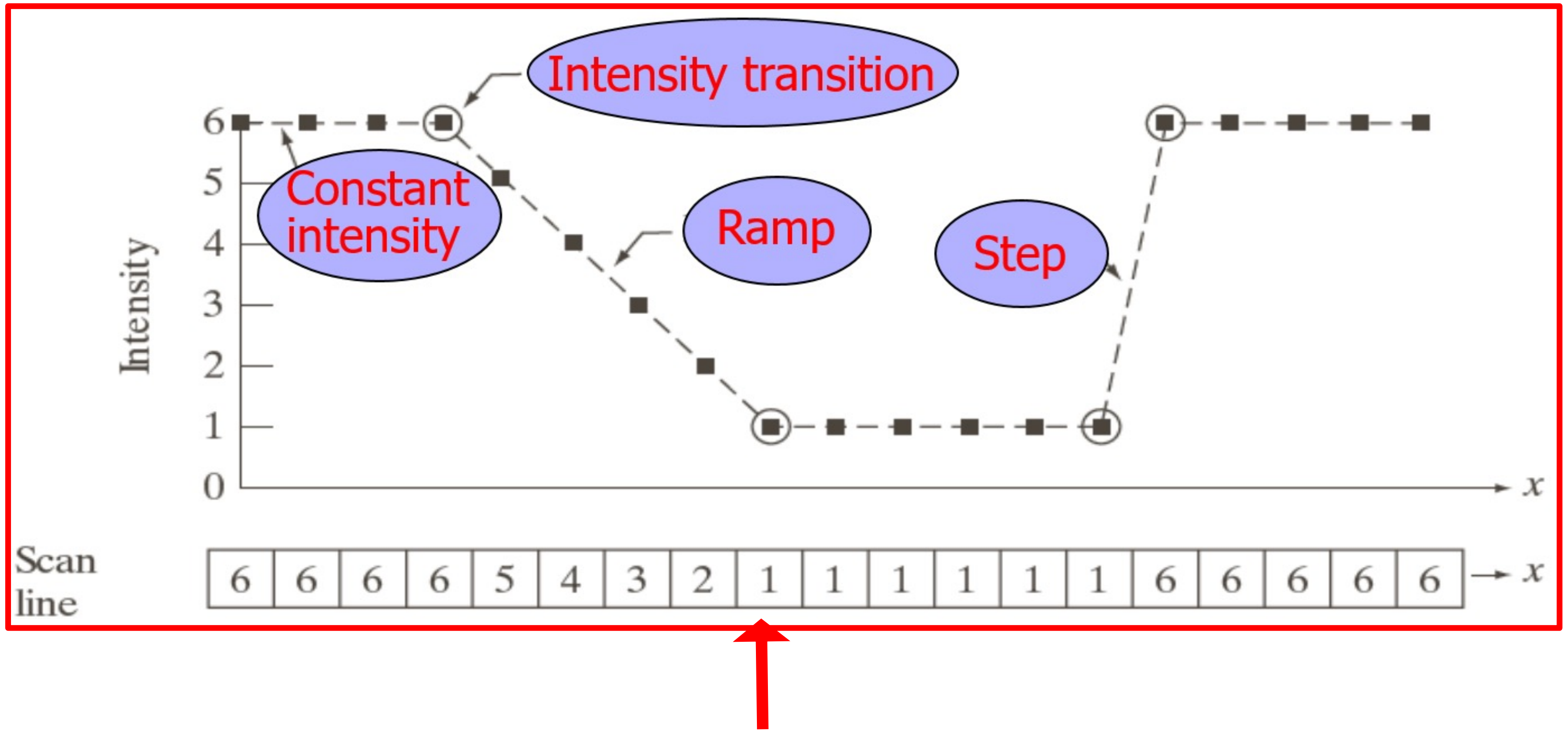
$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

- The second-order derivative of $f(x)$:

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

Sharpening spatial filters

- As an example, consider a scan line (horizontal intensity profile) of an image
- This scan line contains an intensity ramp, three sections of constant intensity, and an intensity step

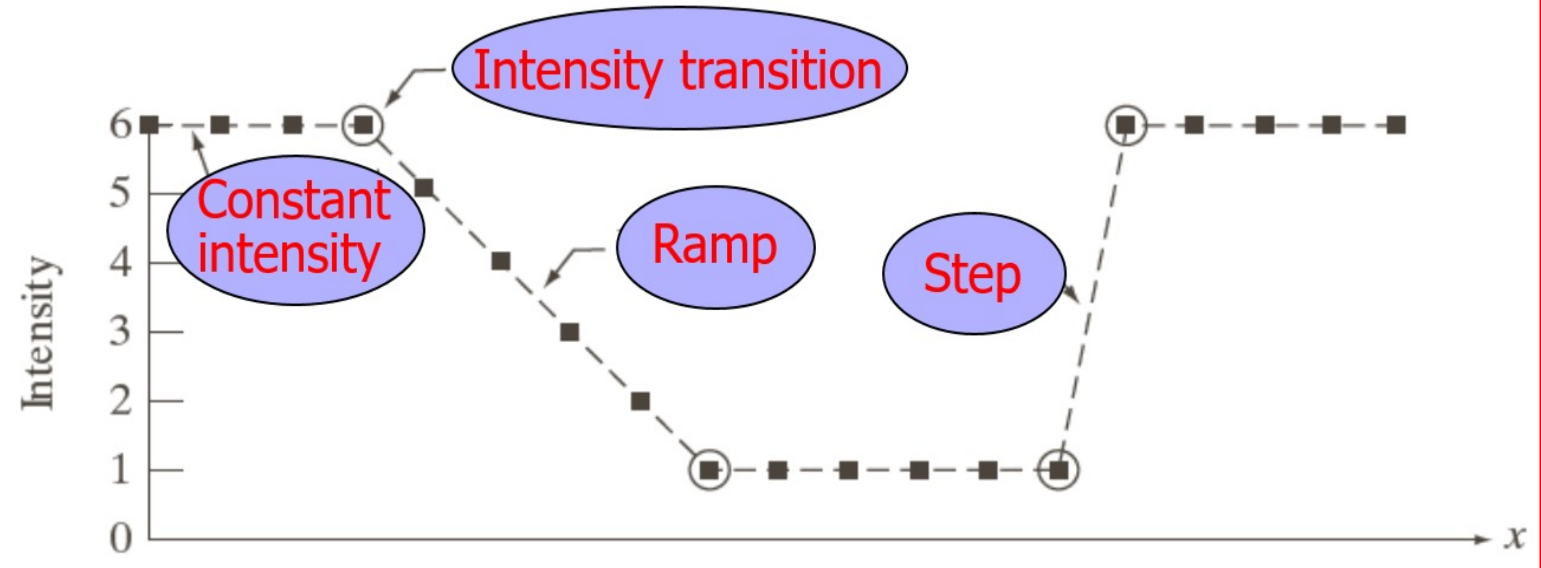


6	6	6	6	5	4	3	2	1	1	1	1	1	1	6	6	6	6	6

intensity values

Sharpening spatial filters

- First- and second- order derivatives can be calculated as:



Scan line																			
	6	6	6	6	5	4	3	2	1	1	1	1	1	1	6	6	6	6	6

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

1st derivative	0	0	−1	−1	−1	−1	−1	0	0	0	0	0	5	0	0	0	0
2nd derivative	0	0	−1	0	0	0	0	1	0	0	0	0	5	−5	0	0	0

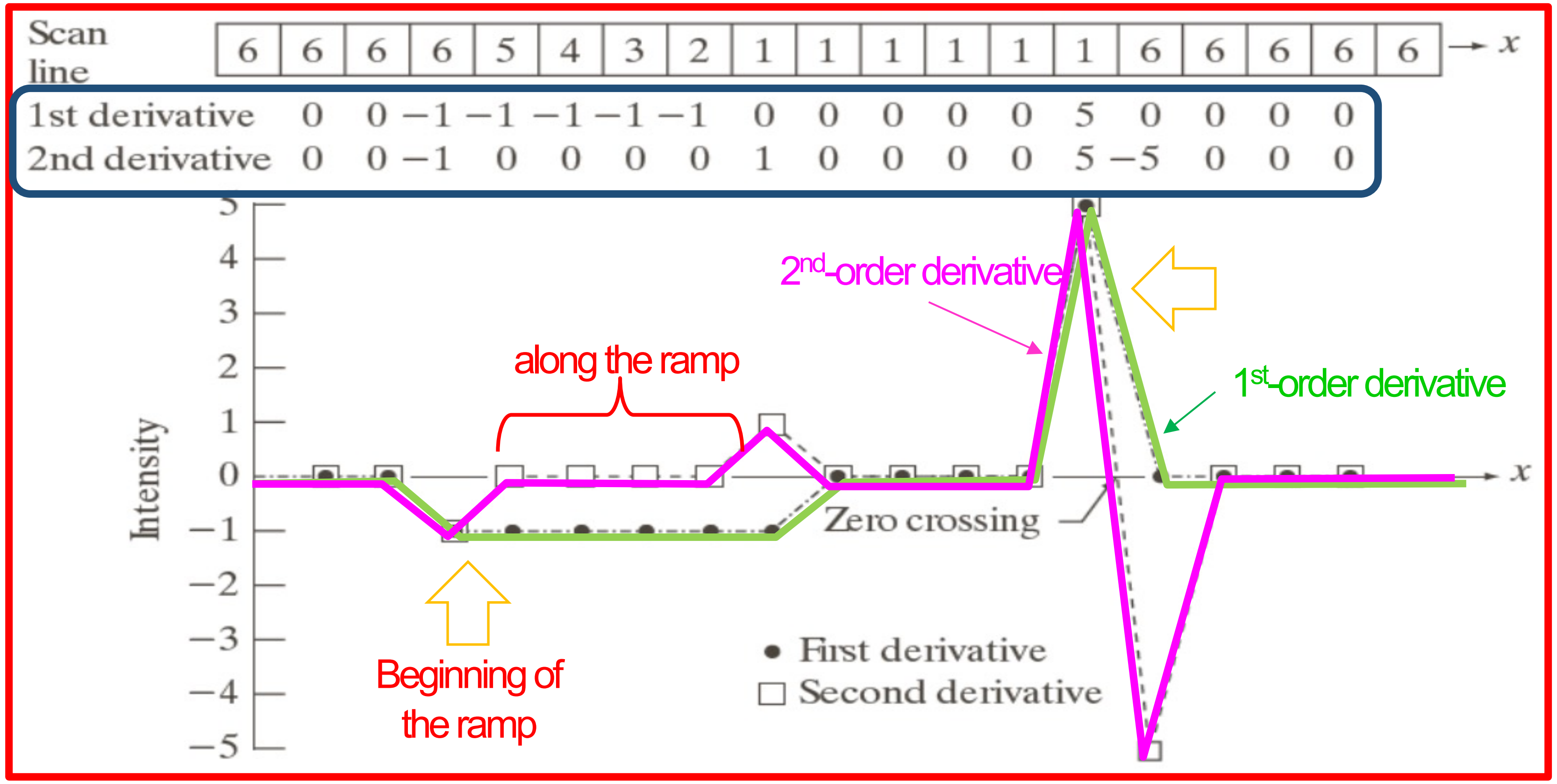
$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

First- and second- order derivatives

Sharpening spatial filters

- Properties of Derivatives:
  - For first-order derivatives:
    - Must be zero in areas of constant intensity
    - Must be nonzero at the onset of an intensity step or ramp
    - Must be nonzero along the ramp
  - For first-order derivatives:
    - Must be zero in constant areas
    - Must be nonzero at the onset and end of an intensity step or ramp
    - Must be zero along ramps of constant slope

Sharpening spatial filters



Scan line	6	6	6	6	5	4	3	2	1	1	1	1	1	1	6	6	6	6	6
1st derivative		0	0	−1	−1	−1	−1	−1	0	0	0	0	0	5	0	0	0	0	
2nd derivative		0	0	−1	0	0	0	0	1	0	0	0	0	5	−5	0	0	0	

- The properties of the derivatives:
  - Constant intensity areas: both derivatives are zero.
  - Beginning of ramp & step: both derivative are nonzero.
  - Along ramp: 1st derivative is nonzero, and 2nd derivative is zero

Sharpening spatial filters

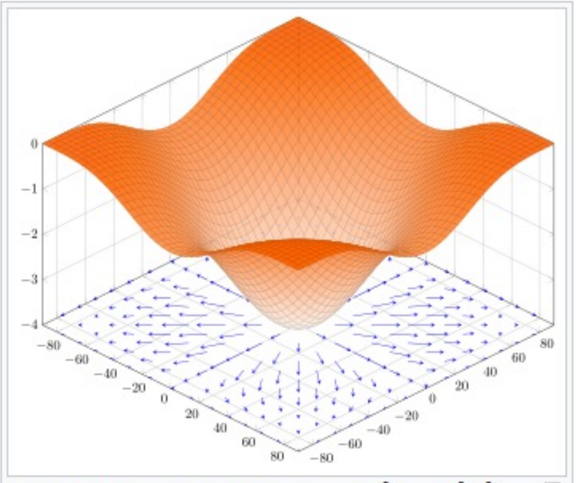
Laplacian operator:

- Laplacian operator is based on second-order derivative
- It is **used to find edges in an image**
- It emphasize the intensity discontinuities in an image
- It is an isotropic filter; which means it is rotation invariant
  - Its response is independent of the direction of the discontinuities i filter is applied

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

f is a twice-differentiable real-valued function

divergence $\nabla = \frac{d}{d_{x_1}}, \ldots, \frac{d}{d_{x_n}}$



The gradient of the function $f(x,y) = -(\cos^2 x + \cos^2 y)^2$ depicted as a projected vector field on the bottom plane.

Image Source: https://en.wikipedia.org/wiki/Gradient

Sharpening spatial filters

Laplacian operator:

For a function (image) $f(x, y)$ of two variables: $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

- In the x-direction, we have

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

- Similarly, in the y-direction we have

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

The discrete Laplacian of two variables is:

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

Sharpening spatial filters

Implementation of the Laplacian

- This equation can be implemented using the filter masks:

(a) $\begin{matrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{matrix}$

(b) $\begin{matrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{matrix}$

(a) Filter mask used to implement equation: $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

(b) Filter mask used to implement an extension of this equation

(c) $\begin{matrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{matrix}$

(d) $\begin{matrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{matrix}$

- The signs of coefficients are opposite in masks
- They result in negative images of the masks

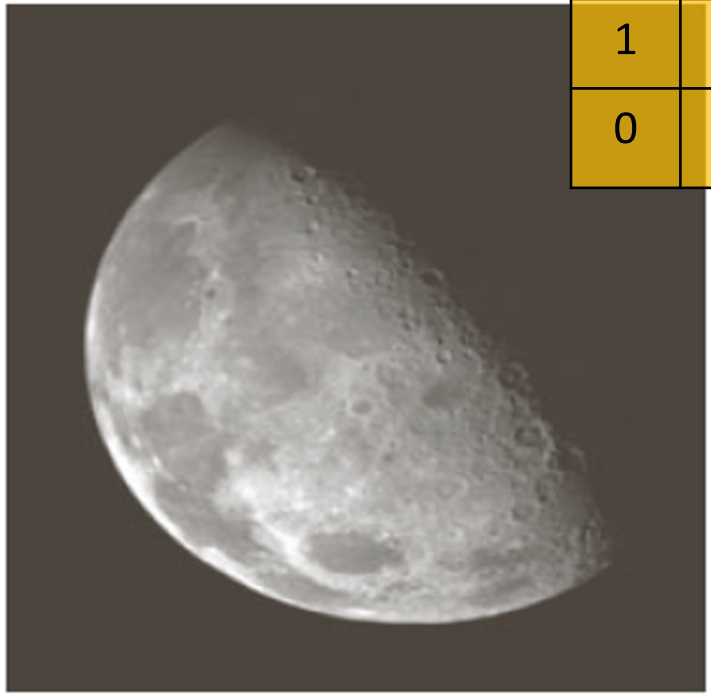
(c)(d) Two other implementations of the Laplacian found frequently in practice

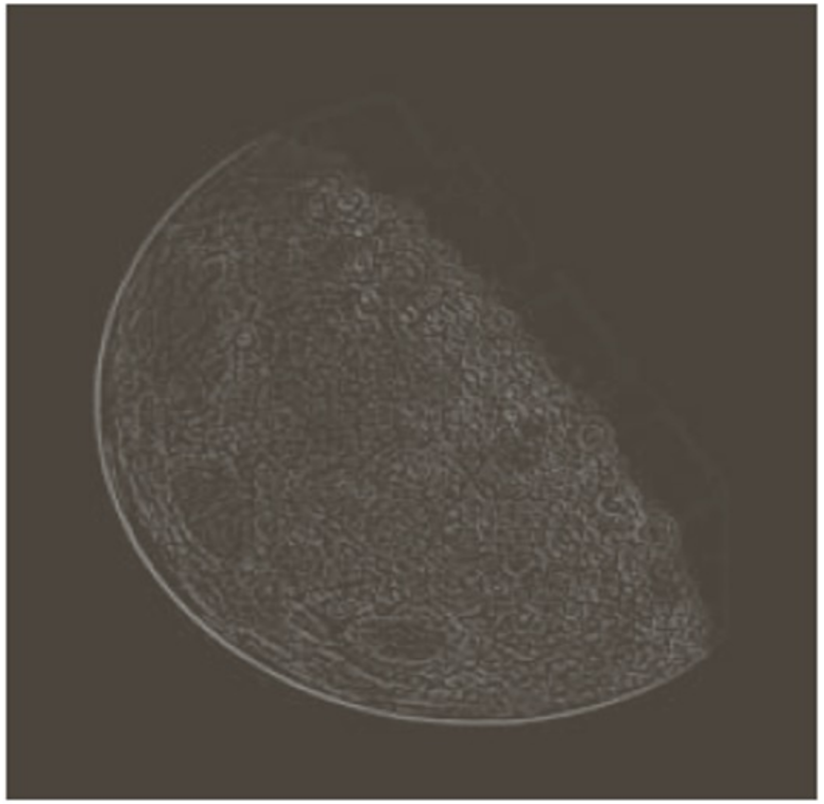
Notice that all the mask coefficients sum to zero.

Sharpening spatial filters

Laplacian filter

0	1	0
1	-4	0
0	1	0

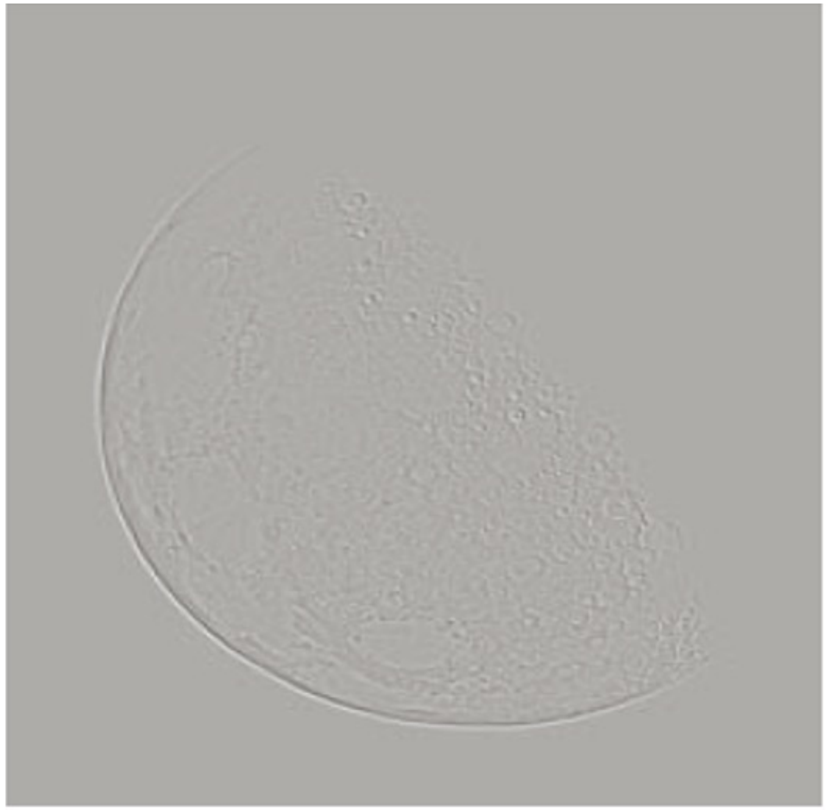

Blurred image of the North pole of the moon


Laplacian without scaling
(using the original intensity pixel values)

Large sections of the image are black because the Laplacian contains both positive and negative values and all negative values are clipped at 0 by the display

Sharpening spatial filters

Laplacian filter


Scaled Laplacian image

Scaling:

Image f:

- Create an image f_m with minimum value is 0:

$$f_m = f - \min(f)$$

- Now scale the pixel intensity values within the range of [0,*L*-1]:

$$f_s = (L-1)[f_m/\max(f_m)]$$

L=256 results in a scaled image with intensities ranging from 0 to 255

Sharpening spatial filters

- Add the Laplacian image to the original image:

$$g(x,y) = f(x,y) + c[\nabla^2 f(x,y)]$$

$g = (x,y)$ is sharpened image

- $c = -1$ if mask (a) or (b) is used
- $c = 1$ if mask (c) or (d) is used

(a) $\begin{array}{ccc} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{array}$

(b) $\begin{array}{ccc} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{array}$

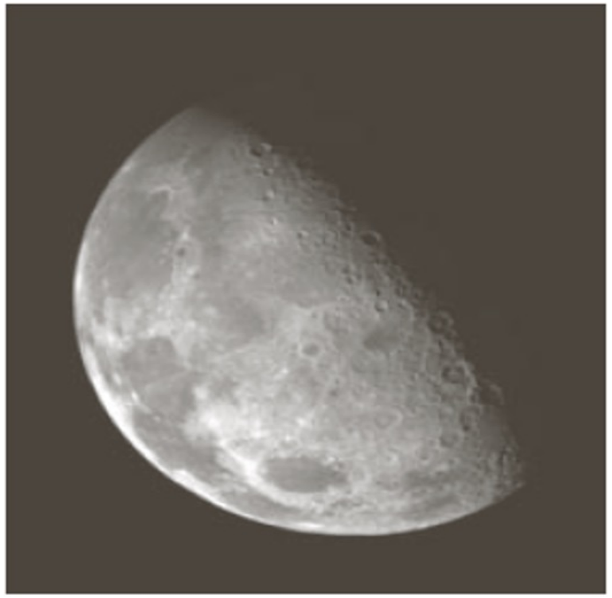
(a) Filter mask used to implement equation: $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

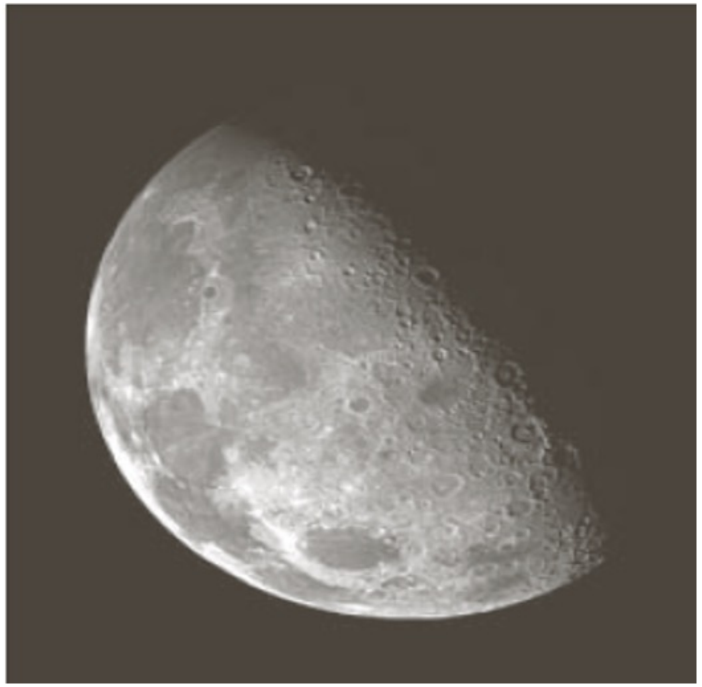
(b) Filter mask used to implement an extension of this equation

(c) $\begin{array}{ccc} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{array}$

(d) $\begin{array}{ccc} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{array}$

(c)(d) Two other implementations of the Laplacian found frequently in practice

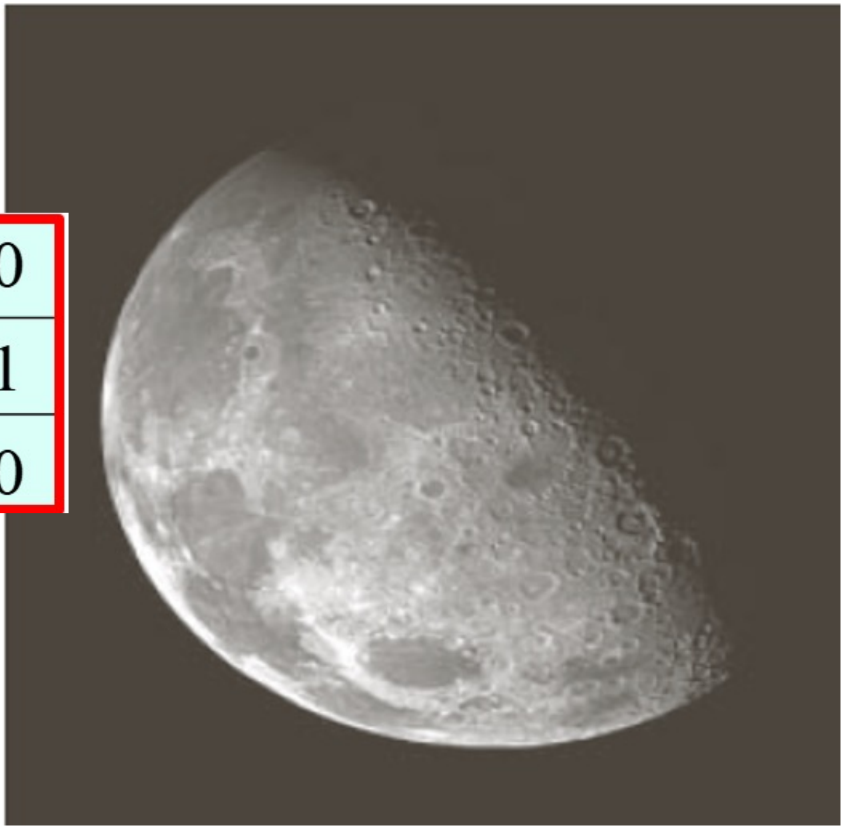

original image

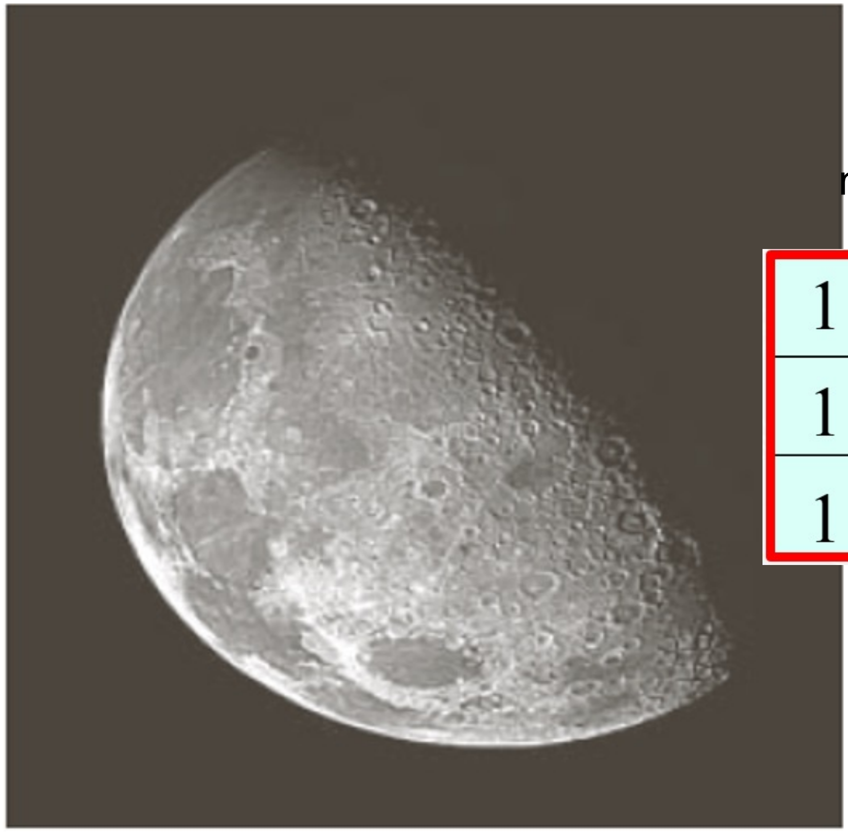

sharpened image

Sharpening spatial filters

mask (a)

0	1	0
1	–4	1
0	1	0


Sharpened image


Sharpened image

mask (b)

1	1	1
1	–8	1
1	1	1

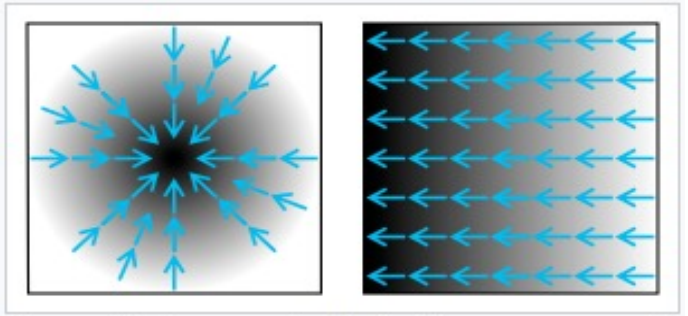
- left: details are clearer and sharper in comparison to the source image
- Right: In comparison to the left, the right image shows a significant improvement in sharpness

First-order derivative

- First derivatives in image processing are implemented using the magnitude of gradient
- The gradient of f (image) at coordinates (x,y) is defined as a 2-D vector

$$\nabla f \equiv \text{grad}(f) \equiv \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

This vector points in the direction of greatest rate of change of f at the coordinate (x, y)



The gradient, represented by the blue arrows, denotes the direction of greatest change of a scalar function. The values of the function are represented in greyscale and increase in value from white (low) to dark (high).

Image Source: https://en.wikipedia.org/wiki/Gradient

- The magnitude (length) of vector ∇f is denoted as:

$$M(x, y) = \text{mag}(\nabla f) = \sqrt{g_x^2 + g_y^2}$$

is the value at (x, y) of the rate of change in direction of the gradient vector

$M(x, y)$ is an image of the same size as the original, created when x and y are allowed to vary over all pixel locations in f

First-order derivative

- $M(x, y)$ is a nonlinear operator and is rotation invariant

$$M(x, y) = \text{mag}(\nabla f) = \sqrt{g_x^2 + g_y^2}$$

- In some applications, it is more suitable to approximate squares and square root operations by absolute values:

$$M(x, y) \approx |g_x| + |g_y|$$

Sobel operator

- Definition in a discrete form:

$$M(x,y) \approx \underbrace{|(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)|}_{g_x} + \underbrace{|(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)|}_{g_y}$$

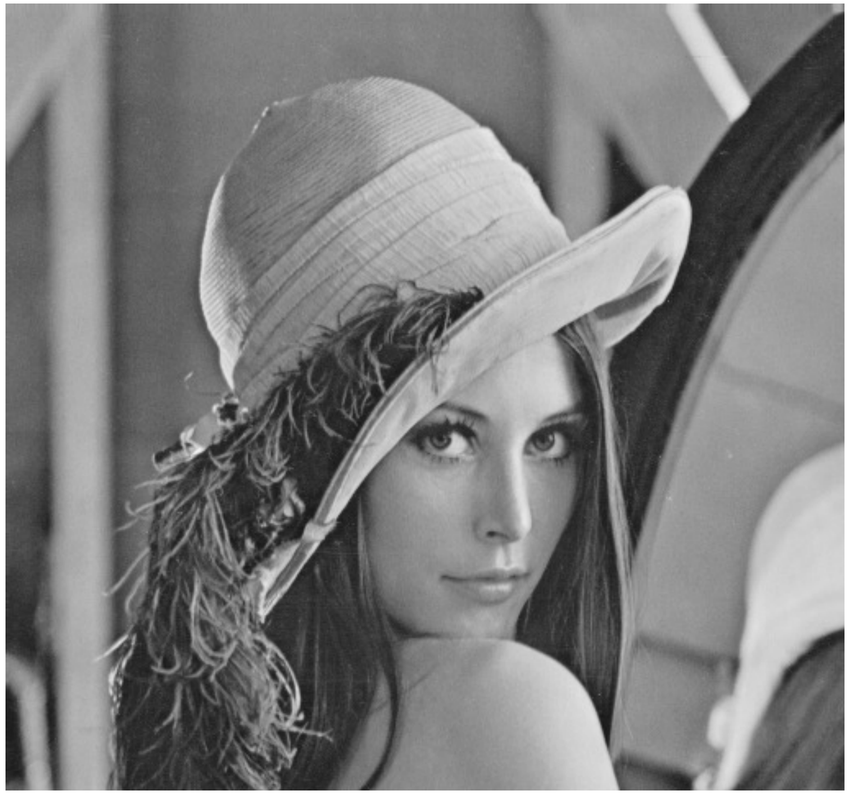
$$\begin{bmatrix} z_1 & z_2 & z_3 \\ z_4 & z_5 & z_6 \\ z_7 & z_8 & z_9 \end{bmatrix}$$

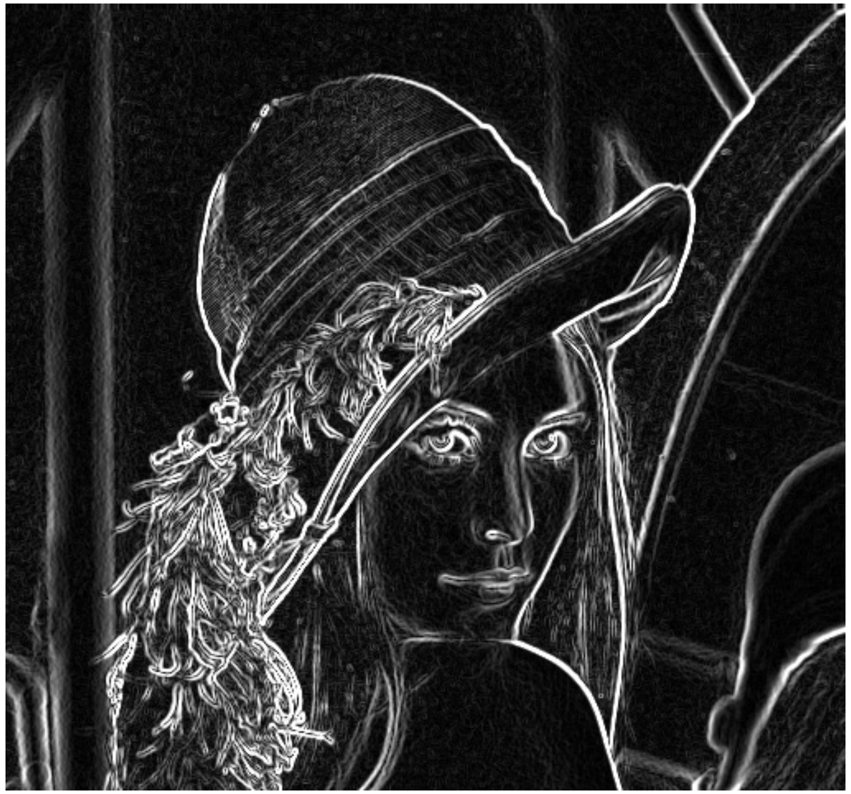
Filter masks:

$$\underbrace{\underset{g_x}{\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}} \quad \underset{g_y}{\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}}}_{\text{Sobel operators}}$$

Image Sharpening by first-order Derivatives

- The result of using Sobel filter


original image


gradient image

Prewitt operator

- Definition in a discrete form:

$$M(x,y) \approx \underbrace{|(z_7 + 2z_8 + z_9) - (z_1 + z_2 + z_3)|}_{g_x} + \underbrace{|(z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)|}_{g_y}$$

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

Filter masks:

-1	-1	-1
0	0	0
1	1	1

g_x

-1	0	1
-1	0	1
-1	0	1

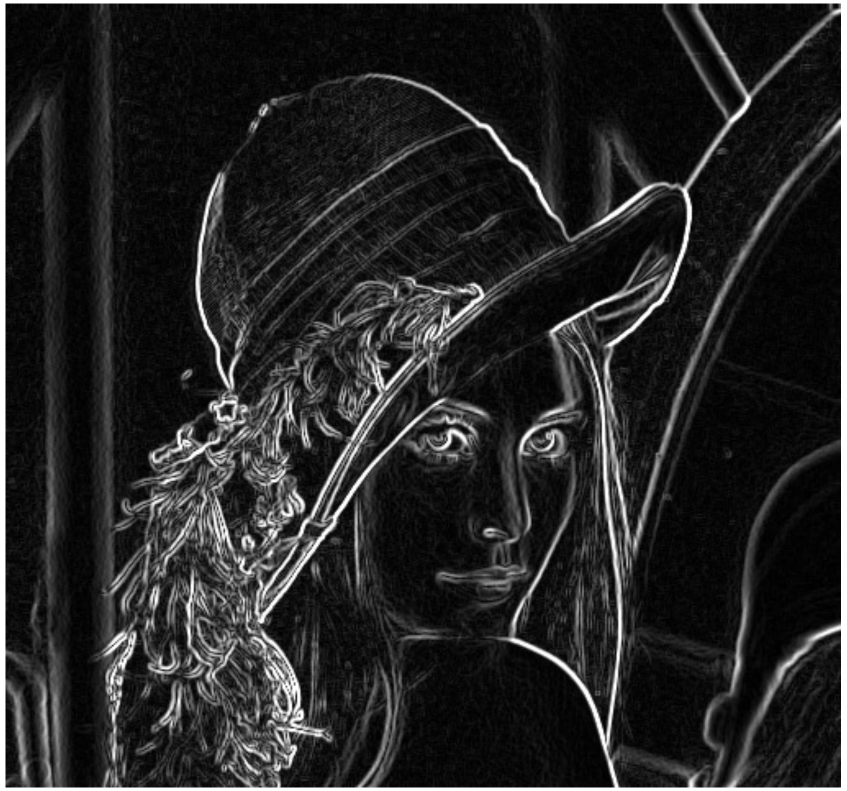
g_y

Sobel operators

Image Sharpening by first-order Derivatives

- The result of using Prewitt filter

original image


gradient image

Thank you for your attention