Fundamentals of Spatial Filtering
and
Smoothing Filters

Recap

- Intensity transformation
  - Basic intensity transformation functions
  - Piecewise intensity transformation functions
  - Histogram processing
    - Histogram stretching
    - Histogram equalization
    - Histogram specification
    - Local histogram processing

Lecture Objectives

- Fundamentals of Spatial Filtering

- Correlation and Convolution

- How to construct Spatial filter masks?

- Smoothing (Lowpass) spatial filters
  - Box filter kernels
  - Gaussian filter kernels
  - Smoothing Non-linear Filters

Key Stages in DIP

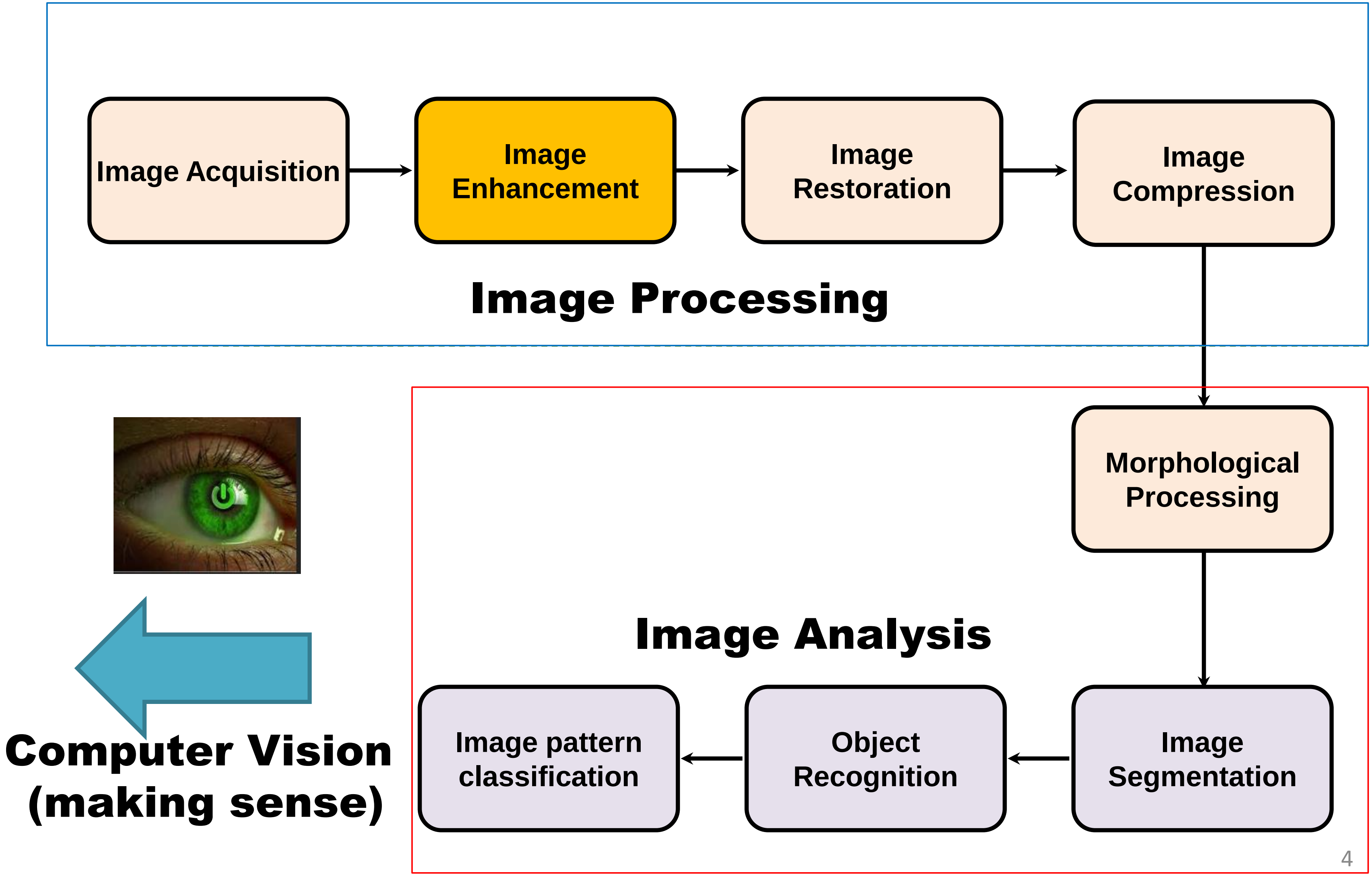


Image Acquisition

Image Enhancement

Image Restoration

Image Compression

Image Processing

Morphological Processing

Image Analysis

Image pattern classification

Object Recognition

Image Segmentation

Computer Vision
(making sense)

Fundamentals of Spatial Filtering

- **Filtering** mostly focus on image enhancement techniques.
- **What is a spatial filter?**
  - Borrowed from frequency domain processing (next topic)
  - **“Filtering”** refers to *passing*, *modifying*, or *rejecting* *specified frequency components of an image*.
    - ❑ For example - filters that pass *low-frequencies*/*high-frequencies* are called a ***lowpass*/*highpass filters***
  - Filter is also called “mask”, “kernel”, “template” or “window”

Fundamentals of Spatial Filtering

- Spatial filters are ***more versatile*** than their frequency domain counterparts.
  - Spatial domain permits non-linear filtering.

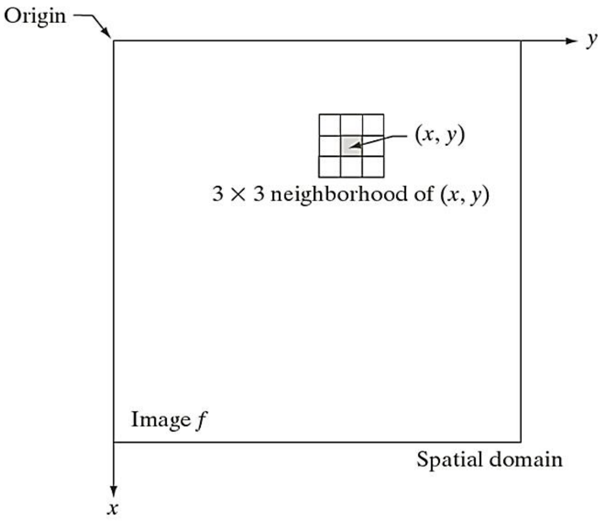
- Spatial filtering modifies an image by **replacing the value of each pixel by a function of the values of the pixel and its neighbors**.

- If the operation performed on the image pixels is linear, then the filter is called a *linear spatial filter*. Otherwise, the filter is a *nonlinear spatial filter*.

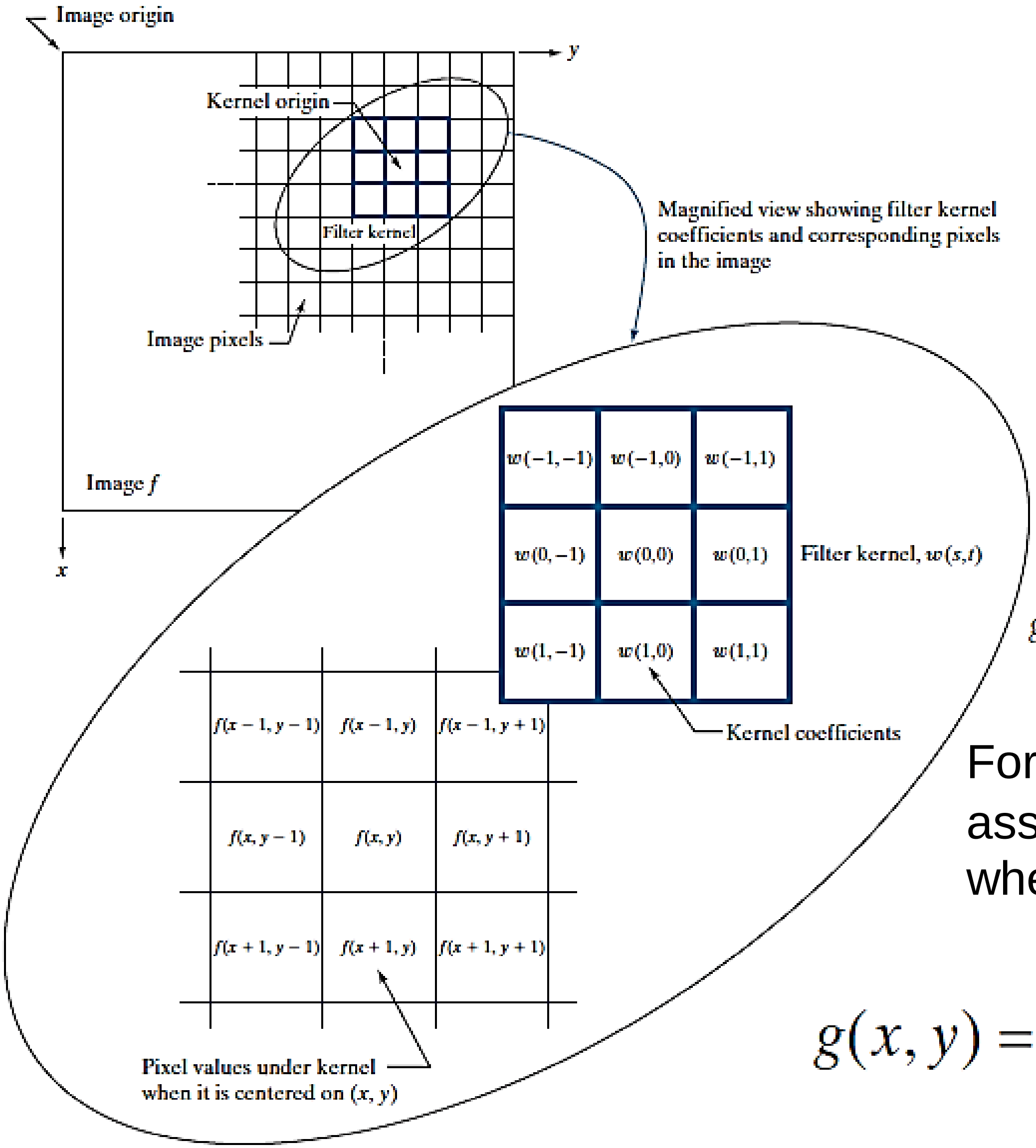
Fundamentals of Spatial Filtering

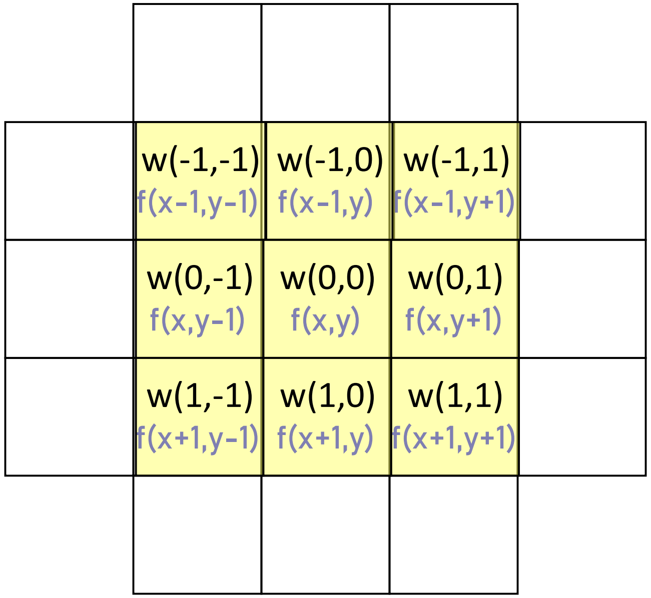
- A linear spatial filter performs a **sum-of-products operation** between an *image* ***f*** and a *filter kernel* ***w***.

- The kernel is an array whose **size** defines the *neighborhood* of operation, and whose **coefficients** determine the *nature of the filter*.



Fundamentals of Spatial Filtering





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)
f(x,y-1)
	w(0,0)
	f(x,y)
		w(0,1)
		f(x,y+1)

w(1,-1)
f(x+1,y-1)
	w(1,0)
	f(x+1,y)
		w(1,1)
		f(x+1,y+1)

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

For a kernel of **odd size** ***m*** × ***n***, we assume that ***m*** **= 2*a* + 1** and ***n*** **= 2*b* + 1**, where ***a*** and ***b*** are nonnegative integers

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

As coordinates ***x*** and ***y*** are **varied**, the center of the kernel **moves** from pixel to pixel, generating the filtered image, ***g***, in the process.

Fundamentals of Spatial Filtering

https://ezyang.github.io/convolution-visualizer/

Correlation and Convolution (important)

Correlation Vs. Convolution

- ***Spatial correlation*** *is the* process of moving a filter mask over the image and computing the sum of products at each location.

- ***Spatial convolution*** are the same as *Special correlation*, except that the correlation kernel is rotated by 180°.

- Thus, when the values of a kernel are **symmetric about its center**, correlation and convolution *yield the same result*.

- We begin by understanding the Spatial correlation operation with a **1-D illustration**.

Understanding Correlation in 1D

- Let **f(x)** be the **1D signal** of length L, and **w(x)** be the **1D kernel** of window size: 2a+1, where **a≥1**.

- The general linear spatial filtering expression given by:

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

becomes the following expression for the 1D correlation:

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

Correlation in 1D - example

- Let **f(x)=**

Value	5	4	2	3	7	4	6	5
index	0	1	2	3	4	5	6	7

- **Kernel:** Averaging with two neighbors

Let **w(i)=**

Value	1	1	1
index	-1	0	1

$$g(x) = \frac{1}{3}\sum_{i=-1}^{1} w(i) f(x+i)$$

Correlation in 1D - example

Kernel: Averaging with two neighbors

$$g(x) = \frac{1}{3}\sum_{i=-1}^{1} w(i) f(x+i)$$

f(x)=

Value	5	4	2	3	7	4	6	5
index	0	1	2	3	4	5	6	7

w(i)=

1	1	1
-1	0	1

$$\begin{aligned} g(1) &= \frac{1}{3}\,[\mathrm{w}(\text{-}1)\times \mathrm{f}(0) + \mathrm{w}(0)\times \mathrm{f}(1) + \mathrm{w}(1)\times \mathrm{f}(2)] \\ &= \frac{1}{3}\,[\,1\times 5 + 1\times 4 + 1\times 2] \\ &= \mathrm{ceil}\left(\frac{11}{3}\right) \\ &= \mathbf{4} \end{aligned}$$

Correlation in 1D - example

Kernel: Averaging with two neighbors

$$g(x) = \frac{1}{3}\sum_{i=-1}^{1} w(i) f(x+i)$$

f(x)=

Value	5	4	2	3	7	4	6	5
index	0	1	2	3	4	5	6	7

w(i)=

$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
-1	0	1

$$\mathbf{g(1)} = [w(-1)\times f(0) + w(0)\times f(1) + w(1)\times f(2)]$$

$$= [\tfrac{1}{3}\times 5 + \tfrac{1}{3}\times 4 + \tfrac{1}{3}\times 2]$$

$$= \text{ceil}\left(\frac{11}{3}\right)$$

$$= \mathbf{4}$$

Boundary Conditions

- What about **g(0)** and **g(7)**? How to handle **undefined area** during convolution?

f(x)=

Value	5	4	2	3	7	4	6	5
index	0	1	2	3	4	5	6	7

w(i)=

1	1	1
-1	0	1

- Zero padding - the 1D signal is **padded with zeros** in both directions.

f(x)=

Value	0	5	4	2	3	7	4	6	5	0
index	-1	0	1	2	3	4	5	6	7	8

w(i)=

1	1	1
-1	0	1

Boundary Conditions

- Zero padding - the 1D signal is **padded with zeros** in both directions.

f(x)=

Value	**0**	**5**	**4**	**2**	**3**	**7**	**4**	**6**	**5**	**0**
index	-1	0	1	2	3	4	5	6	7	8

w(i)=

1	**1**	**1**
-1	0	1

Boundary Conditions

- Replicate padding - the boundary values are **repeated** as necessary to "complete" the signal.

f(x)=

Value	5	5	4	2	3	7	4	6	5	5
index	-1	0	1	2	3	4	5	6	7	8

w(i)=

1	1	1
-1	0	1

f(x)=

Value	5	5	4	2	3	7	4	6	5	5
index	-1	0	1	2	3	4	5	6	7	8

w(i)=

1	1	1
-1	0	1

Boundary Conditions

- Mirror padding - the values outside the boundary are obtained by **mirror-reflecting** the image across its border to “complete” the signal.

$f(x)=$

Value	4	5	4	2	3	7	4	6	5	6
index	-1	0	1	2	3	4	5	6	7	8

$w(i)=$

1	1	1
-1	0	1

$f(x)=$

Value	4	5	4	2	3	7	4	6	5	6
index	-1	0	1	2	3	4	5	6	7	8

$w(i)=$

1	1	1
-1	0	1

Boundary Conditions

- Ignore - The signal values at the boundary locations is **ignored** during correlation.
  - **Output has less values than the input signal !!!**

f(x)=

Value	5	4	2	3	7	4	6	5
index	0	1	2	3	4	5	6	7

w(i)=

1	1	1
-1	0	1

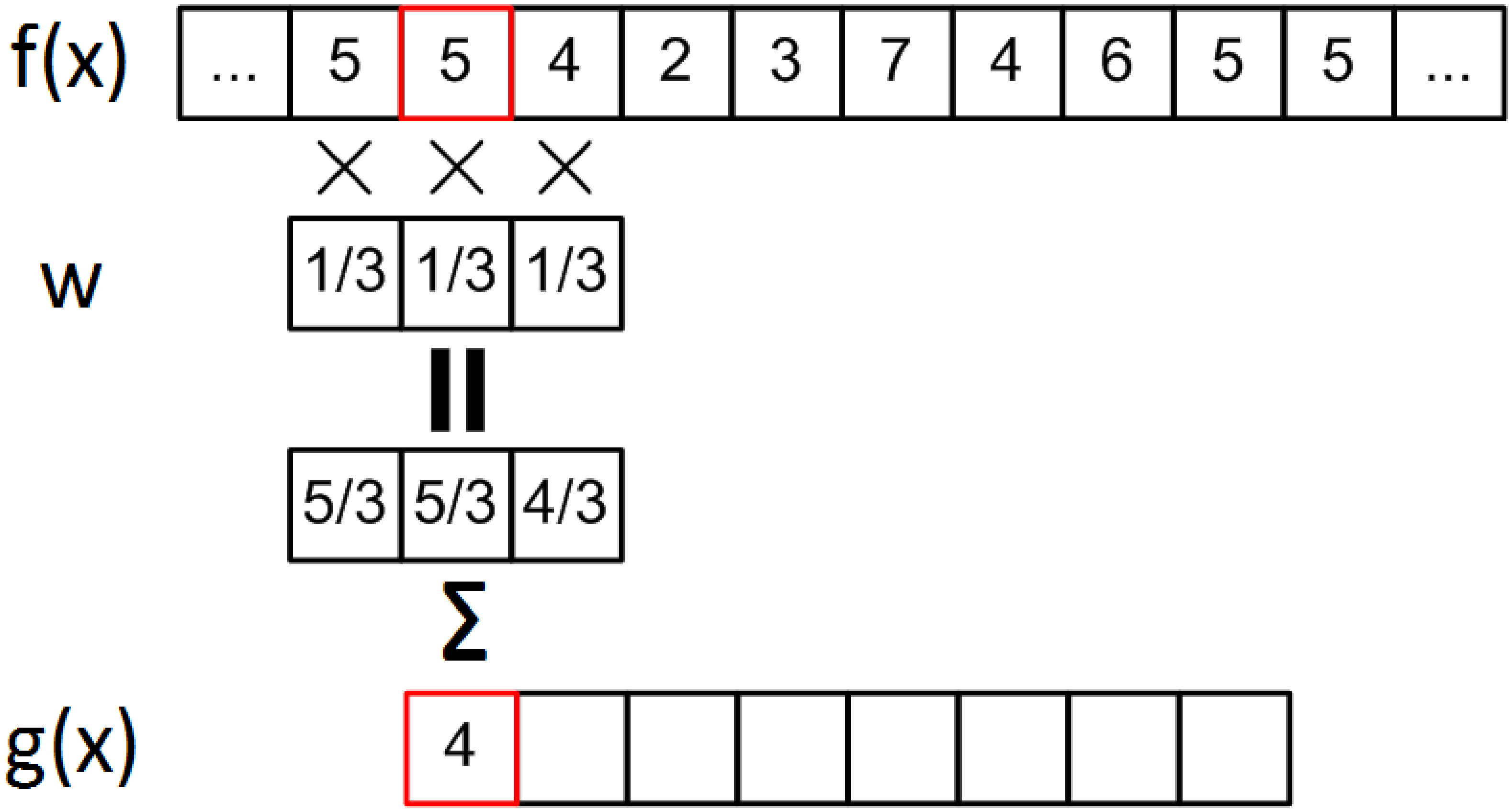
f(x)=

Value	5	4	2	3	7	4	6	5
index	0	1	2	3	4	5	6	7

w(i)=

1	1	1
-1	0	1

Correlation in 1D: Step-1



f(x) | ... | 5 | 5 | 4 | 2 | 3 | 7 | 4 | 6 | 5 | 5 | ...

× × ×

w | 1/3 | 1/3 | 1/3

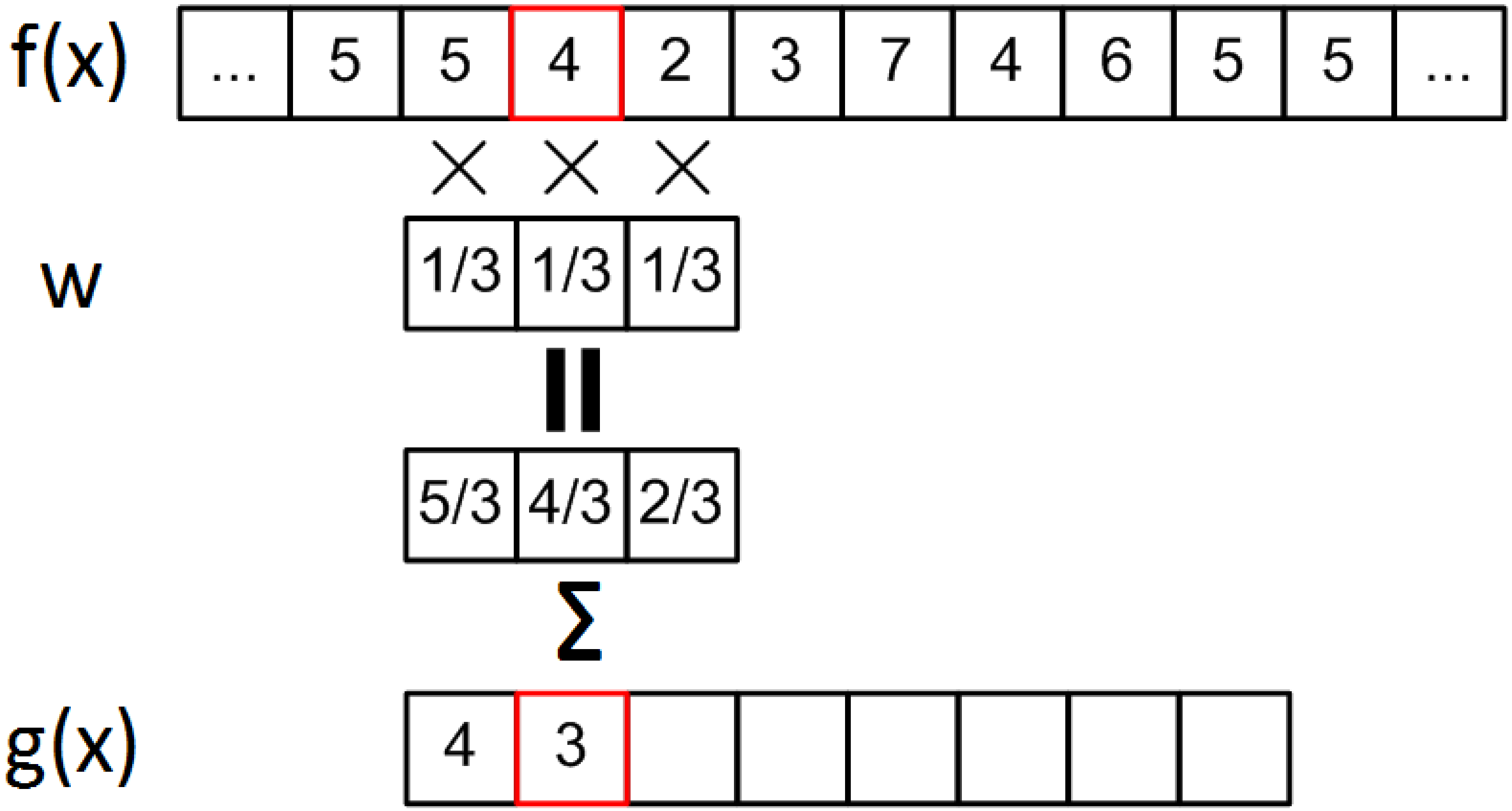
=

5/3 | 5/3 | 4/3

Σ

g(x) | 4 | | | | | | |

Correlation in 1D: Step-2



f(x) | ... | 5 | 5 | 4 | 2 | 3 | 7 | 4 | 6 | 5 | 5 | ...

× × ×

w | 1/3 | 1/3 | 1/3

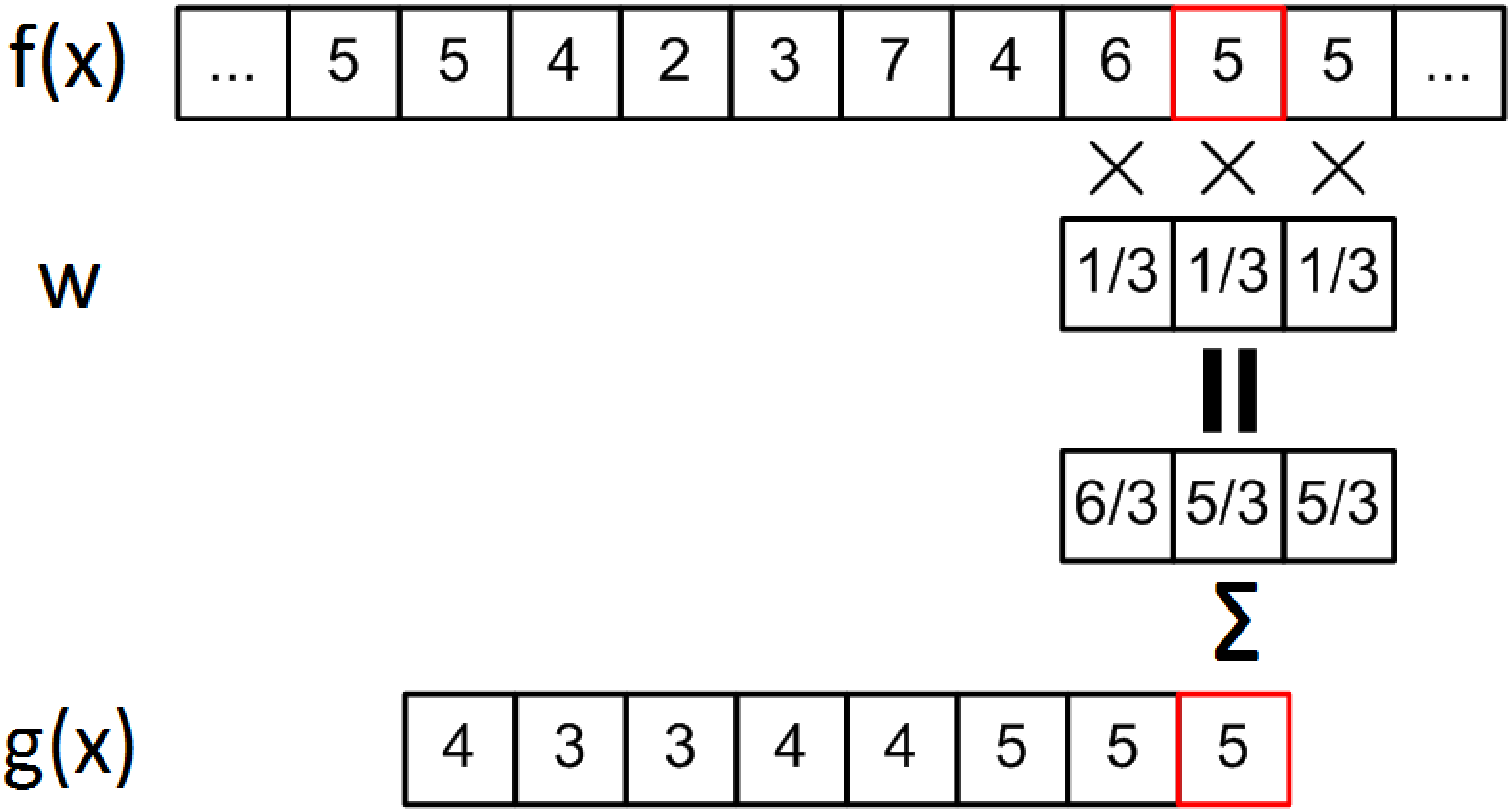
=

5/3 | 4/3 | 2/3

Σ

g(x) | 4 | 3 | | | | | |

Correlation in 1D: Step-8

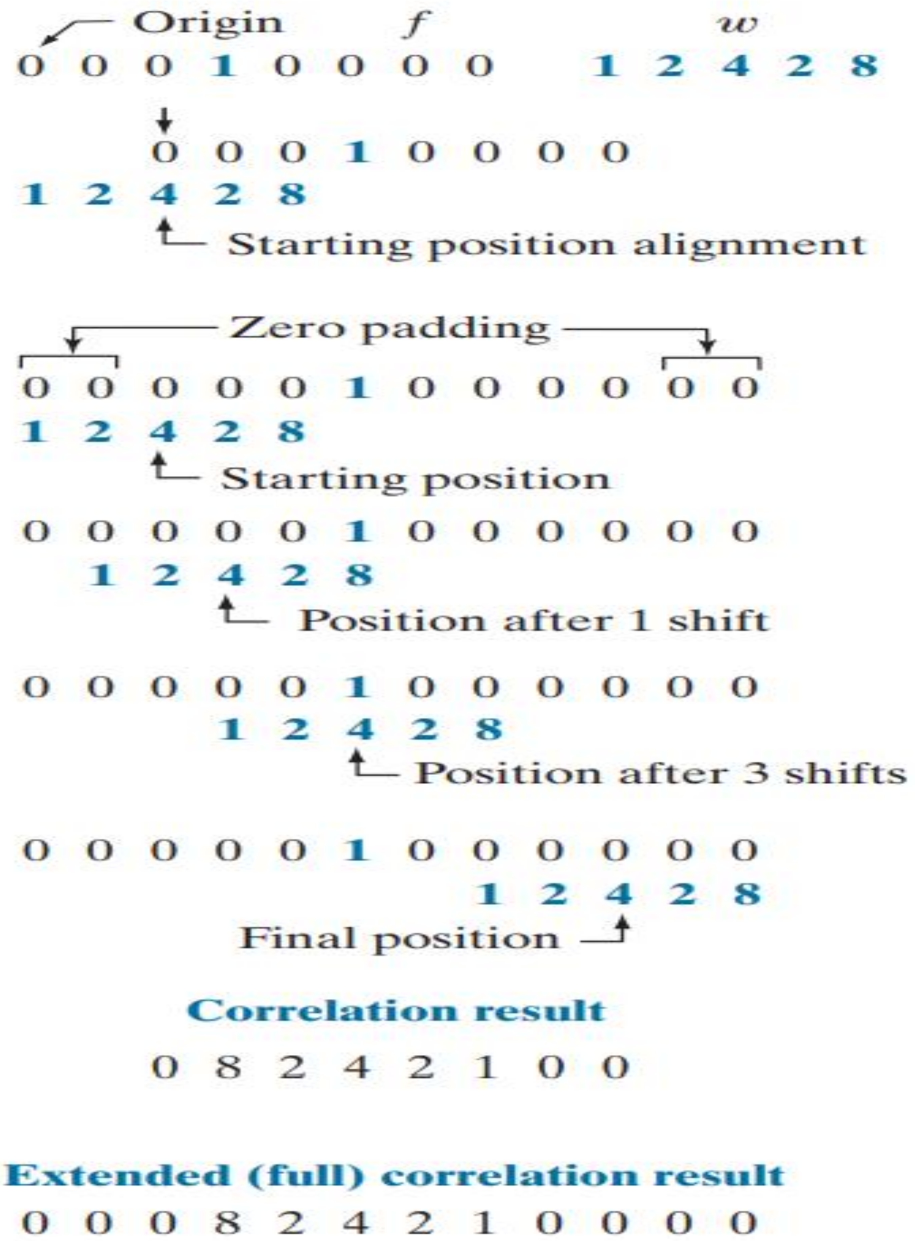


f(x)	...	5	5	4	2	3	7	4	6	5	5	...

w	1/3	1/3	1/3
	6/3	5/3	5/3

g(x)	4	3	3	4	4	5	5	5

Correlation in 1D - Example



Origin f w

0 0 0 1 0 0 0 0 1 2 4 2 8

0 0 0 1 0 0 0 0

1 2 4 2 8

Starting position alignment

Zero padding

0 0 0 0 0 1 0 0 0 0 0 0

1 2 4 2 8

Starting position

0 0 0 0 0 1 0 0 0 0 0 0

1 2 4 2 8

Position after 1 shift

0 0 0 0 0 1 0 0 0 0 0 0

1 2 4 2 8

Position after 3 shifts

0 0 0 0 0 1 0 0 0 0 0 0

1 2 4 2 8

Final position

Correlation result

0 8 2 4 2 1 0 0

Extended (full) correlation result

0 0 0 8 2 4 2 1 0 0 0 0

Correlation in 1D - facts

- Its basically a sliding window operation
  - The filter is placed on every input value in turn and the corresponding output value is computed.

- For convenience, assume odd length filter window.

- The center location on the filter window is its **origin** (location 0).

- Assume **2a+1** elements in a filter, the indices go from **[-a, a]** .

- The number of padding required on each side = **a** .

Correlation in 1D - Practice

- f(x) = [0, 0, 0, 1, 0, 0, 0]
- w(x) = [1, 2, 3, 4, 5]
- **g(x)** = ? **Use zero padding**

Correlation in 1D - Practice

- $f(x) = [0, 0, 0, 1, 0, 0, 0]$

- $w(x) = [1, 2, 3, 4, 5]$

- **g(x) = [0, 5, 4, 3, 2, 1, 0]**

Correlation in 1D - Practice

- $f(x) = [1, 2, 3, 4, 5, 6, 7]$

- $w(x) = [0, 0, 1, 0, 0]$

- $\mathbf{g(x)}$ = ? **Use zero padding**

Correlation in 1D - Practice

- f(x) = [1, 2, 3, 4, 5, 6, 7]
- w(x) = [0, 0, 1, 0, 0]
- **g(x) = [1, 2, 3, 4, 5, 6, 7]**

Correlation in 1D - facts

- f(x)= [0, 0, 0, 1, 0, 0, 0]
- w(x)=[1, 2, 3, 4, 5]
- g(x)= [0, 5, 4, 3, 2, 1, 0]

- f(x)= [1, 2, 3, 4, 5, 6, 7]
- w(x)=[0, 0, 1, 0, 0]
- g(x)= [1, 2, 3, 4, 5, 6, 7]

Discrete unit impulse

– A function that contains a ***single 1*** with the ***rest being 0's*** is called a **discrete unit impulse**.

– *Correlating a kernel with a discrete unit impulse* yields a rotated version of the kernel at the location of the impulse.

– *Correlating a function with a discrete unit impulse* yields the same function at the location of the impulse.

Correlation in 2D

- Let f(x,y) be a **2D signal/image** of size **L×L** and **w(x,y)** be a **2D kernel** of window size (2a+1,2b+1), where **a ≥1** and **b≥1**.

- The 2D correlation is represented by:

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

- For a kernel of size **m × n** , using zero padding, we pad the image with a minimum of **(m – 1)/2** rows of 0's at the **top** and **bottom** and **(n – 1)/2** columns of 0's on the **left** and **right**.

2D Correlation - Replicate padding example

f(x,y)

8	3	4	5
7	6	4	5
4	5	7	8
6	5	5	6

w(x,y)

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

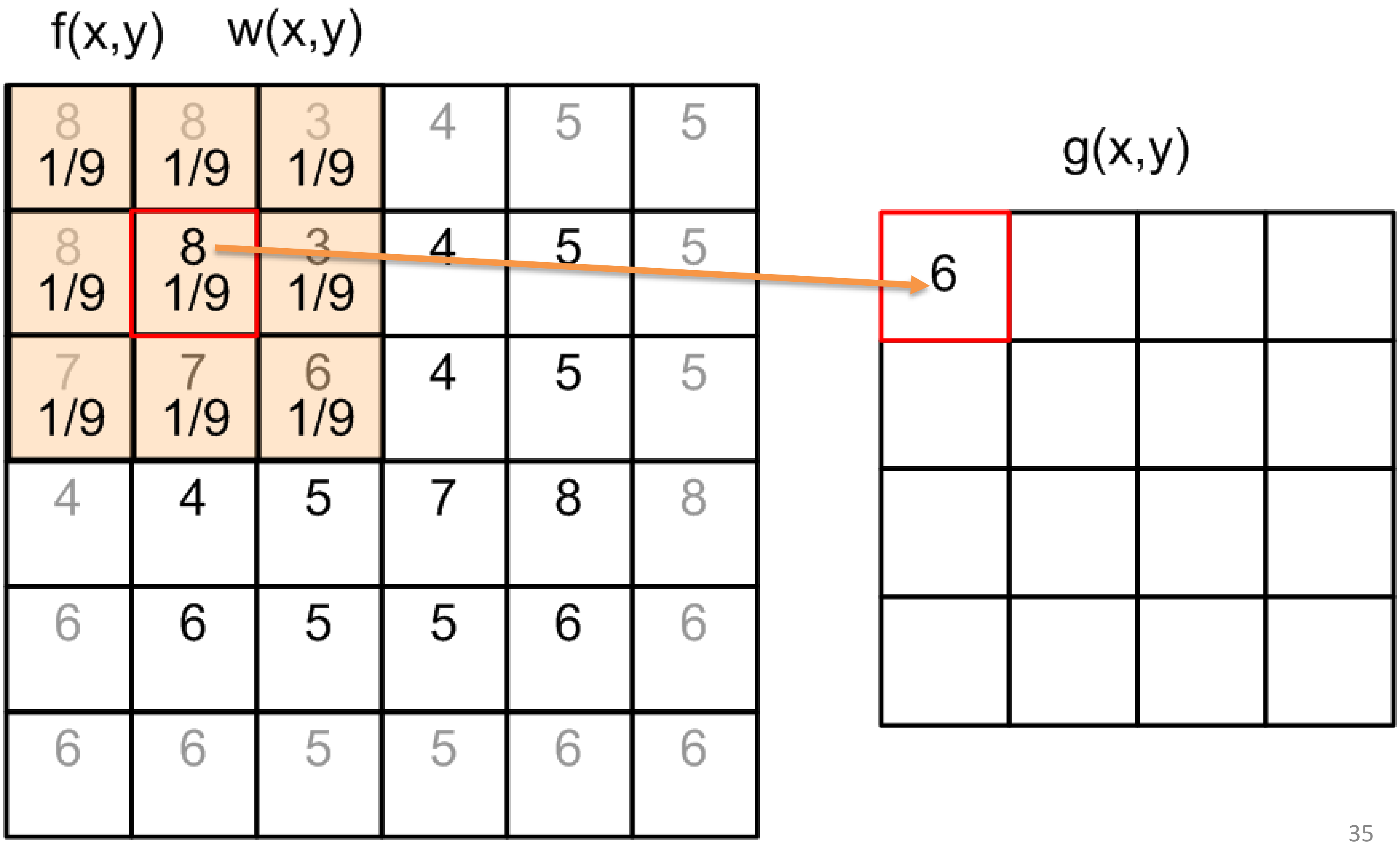
Step 0

f(x,y) with padded values

8	8	3	4	5	5
8	8	3	4	5	5
7	7	6	4	5	5
4	4	5	7	8	8
6	6	5	5	6	6
6	6	5	5	6	6

2D Correlation - Replicate padding example

step-1

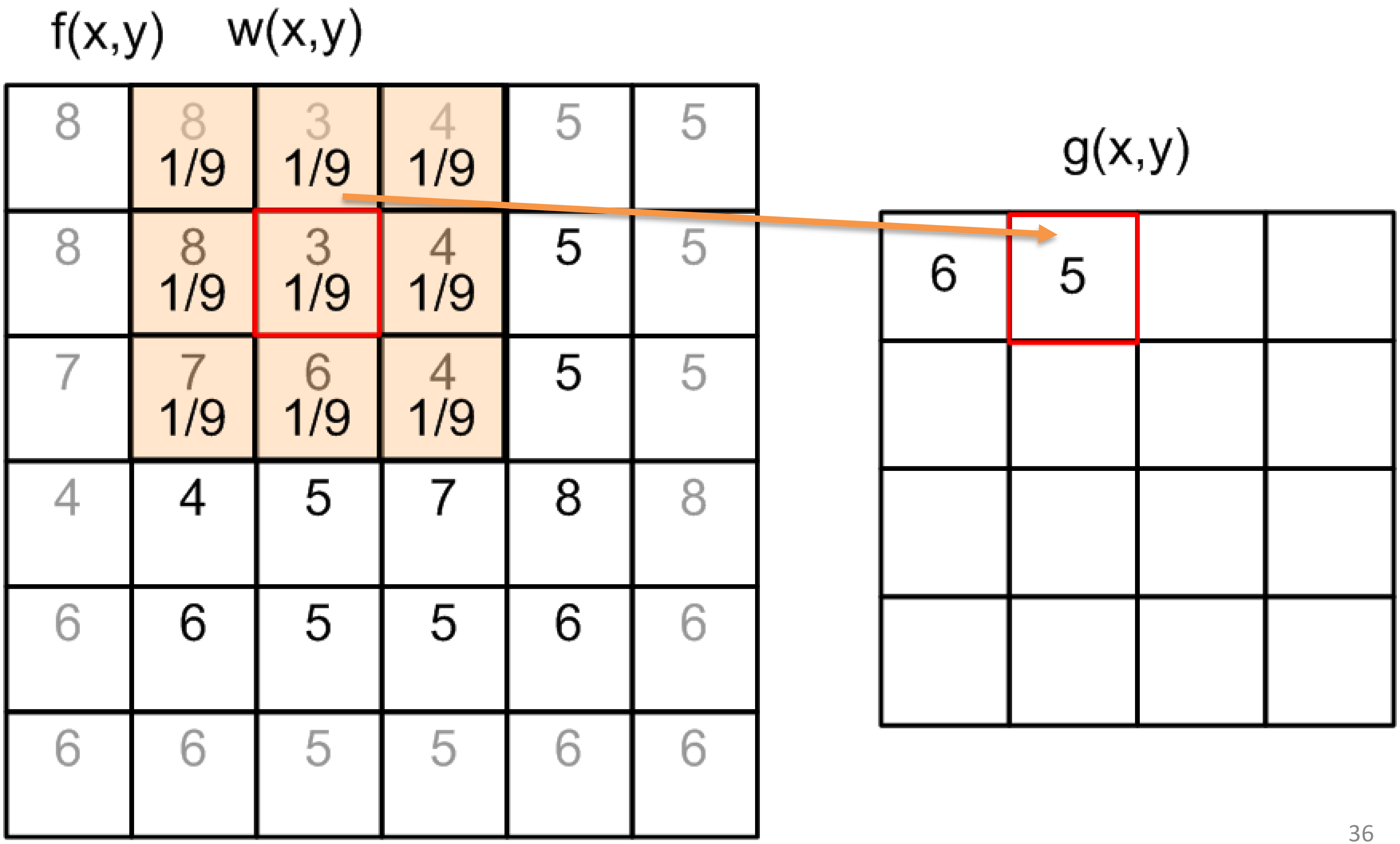


8 1/9	8 1/9	3 1/9	4	5	5
8 1/9	8 1/9	3 1/9	4	5	5
7 1/9	7 1/9	6 1/9	4	5	5
4	4	5	7	8	8
6	6	5	5	6	6
6	6	5	5	6	6

6			

2D Correlation - Replicate padding example

step-2



f(x,y) w(x,y)

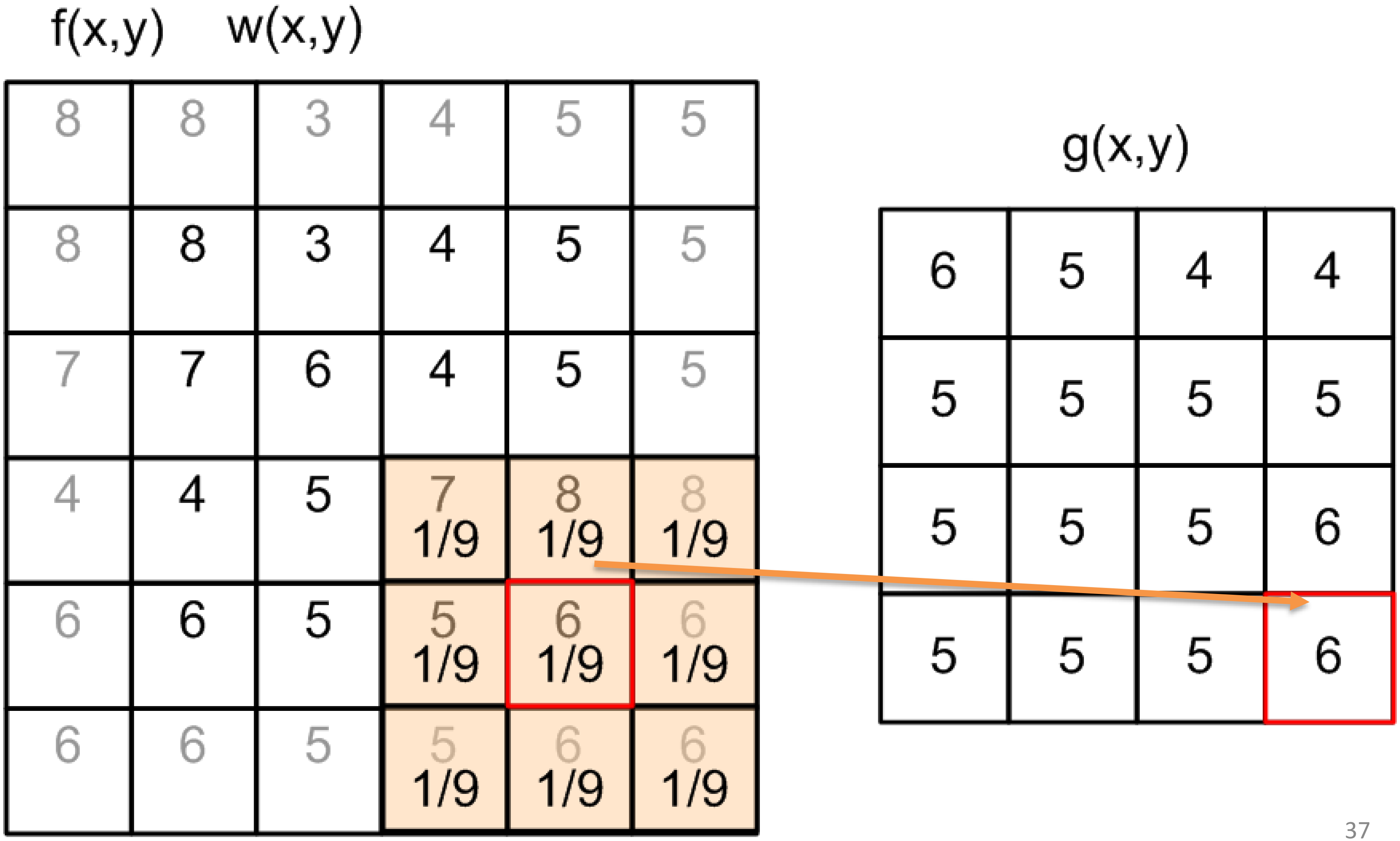
8	8 1/9	3 1/9	4 1/9	5	5
8	8 1/9	3 1/9	4 1/9	5	5
7	7 1/9	6 1/9	4 1/9	5	5
4	4	5	7	8	8
6	6	5	5	6	6
6	6	5	5	6	6

g(x,y)

6	5		

2D Correlation - Replicate padding example

step-16



f(x,y) w(x,y)

8	8	3	4	5	5
8	8	3	4	5	5
7	7	6	4	5	5
4	4	5	7 1/9	8 1/9	8 1/9
6	6	5	5 1/9	6 1/9	6 1/9
6	6	5	5 1/9	6 1/9	6 1/9

g(x,y)

6	5	4	4
5	5	5	5
5	5	5	6
5	5	5	6

2D Correlation - Replicate padding example practice

f(x,y)

0	0	0	0
0	1	1	0
0	1	1	0
0	0	0	0

w(x,y)

1	2	3
4	5	6
7	8	9

g(x,y)

2D Correlation - Replicate padding example practice

f(x,y)

0	0	0	0
0	1	1	0
0	1	1	0
0	0	0	0

w(x,y)

1	2	3
4	5	6
7	8	9

g(x,y)

9	17	15	7
15	28	24	11
9	16	12	5
3	5	3	1

Convolution

- Convolution is similar to correlation, except that the **filter is first flipped by 180°**
  - **1D** - reverse the values in the filter
    - $w_f[i] = w[N\text{-}i\text{-}1]$
    - Example of W=[1 2 3 4 5] becomes [5 4 3 2 1]
  - **2D** - flip the filter about the centre
    - $w_f[i][i] = w[N\text{-}i\text{-}1][N\text{-}i\text{-}1]$
    - Example of W= $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ becomes $\begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}$

Convolution in 1D

- Let **f(x)** be the **1D** signal of length **L**, and **w(x)** be the **1D** kernel of window size: **2a+1**, where **a≥1** .

- The 1D convolution is represented by :

$$g(x) = \sum_{s=-a}^{a} w(s) f(x-s)$$

where the **minus signs** *align the coordinates* of ***f*** and **w** when one of the functions is *rotated* by **180°**

Correlation and **Convolution** yield the same result if the kernel values are **symmetric** about the center.

Convolution in 1D - example

Kernel: Averaging with two neighbors

$$g(x) = \frac{1}{3}\sum_{i=-1}^{1} w(i)\, f(x-i)$$

f(x)=

Value	5	4	2	3	7	4	6	5
index	0	1	2	3	4	5	6	7

w(i)=

$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
-1	0	1

$$g(1) = [w(-1)\times f(2) + w(0)\times f(1) + w(1)\times f(0)]$$

$$= [\tfrac{1}{3}\times 2 + \tfrac{1}{3}\times 4 + \tfrac{1}{3}\times 5]$$

$$= \mathrm{ceil}\left(\frac{11}{3}\right)$$

$$= \mathbf{4}$$

Convolution in 1D - Practice

- f(x) = [0, 0, 0, 1, 0, 0, 0]
- w(x) = [1, 2, 3, 4, 5]
- **g(x)** = ? **Use zero padding**

Convolution in 1D - Practice

- $f(x) = [0, 0, 0, 1, 0, 0, 0]$
- $w(x) = [1, 2, 3, 4, 5]$

flip the kernel by **180°**

$w(x)=[5, 4, 3, 2, 1]$

- **g(x) = [0, 1, 2, 3, 4, 5, 0]**

Vs. Correlation in 1D

- $f(x) = [0, 0, 0, 1, 0, 0, 0]$
- $w(x) = [1, 2, 3, 4, 5]$
- **g(x) = [0, 5, 4, 3, 2, 1, 0]**

Convolution in 1D - Practice

- f(x) = [1, 2, 3, 4, 5, 6, 7]
- w(x) = [0, 0, 1, 0, 0]
- **g(x)** = ? **Use zero padding**

Convolution in 1D - Practice

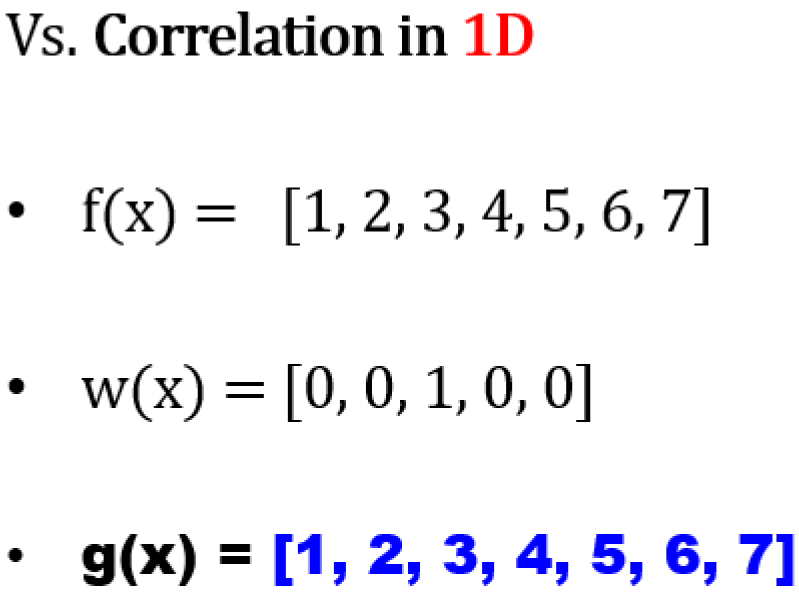
- f(x) = [1, 2, 3, 4, 5, 6, 7]

- w(x) = [0, 0, 1, 0, 0]

flipping the kernel by **180°** would result in the same kernel

w(x) = [0, 0, 1, 0, 0]

- **g(x) = [1, 2, 3, 4, 5, 6, 7]**



Vs. Correlation in 1D

- f(x) = [1, 2, 3, 4, 5, 6, 7]

- w(x) = [0, 0, 1, 0, 0]

- **g(x) = [1, 2, 3, 4, 5, 6, 7]**

Convolution in 1D - facts

- f(x)= [0, 0, 0, 1, 0, 0, 0]
- w(x)=[1, 2, 3, 4, 5] flipped = **[5, 4, 3, 2, 1]**
- g(x)= [0, 1, 2, 3, 4, 5, 0]

- f(x)= [1, 2, 3, 4, 5, 6, 7]
- w(x)=[0, 0, 1, 0, 0] flipped = **[0, 0, 1, 0, 0]**
- g(x)= [1, 2, 3, 4, 5, 6, 7]

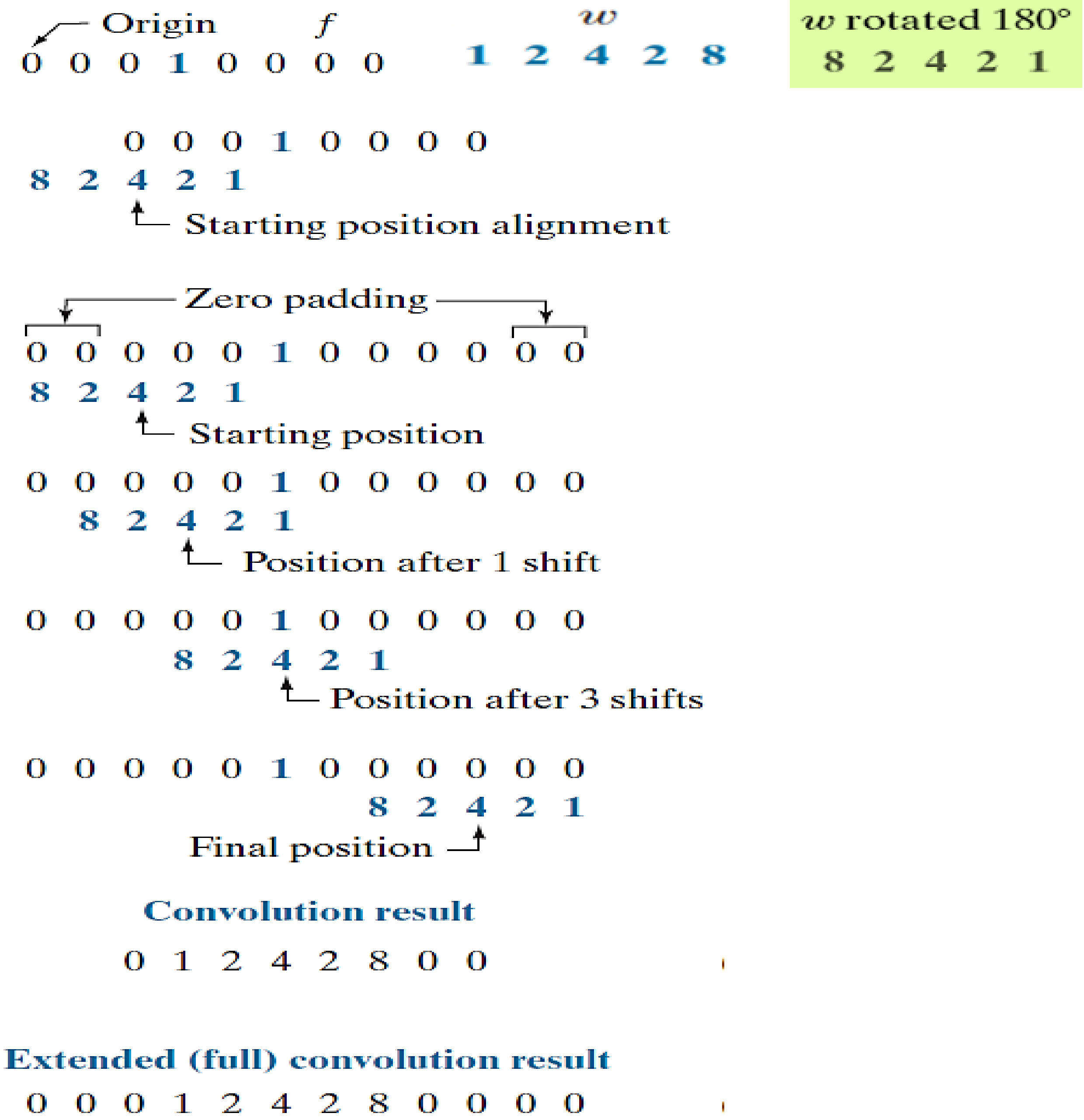
Discrete unit impulse

  - A function that contains a ***single 1*** with the ***rest being 0's*** is called a **discrete unit** impulse.

  - *Convolving a kernel with a discrete unit impulse* yields the same kernel at the location of the impulse.

  - *Convolving a function with a discrete unit impulse* yields the same function at the location of the impulse.

Convolution in 1D - Example



Origin f

0 0 0 1 0 0 0 0

w

1 2 4 2 8

w rotated 180°

8 2 4 2 1

0 0 0 1 0 0 0 0

8 2 4 2 1

Starting position alignment

Zero padding

0 0 0 0 0 1 0 0 0 0 0 0

8 2 4 2 1

Starting position

0 0 0 0 0 1 0 0 0 0 0 0

8 2 4 2 1

Position after 1 shift

0 0 0 0 0 1 0 0 0 0 0 0

8 2 4 2 1

Position after 3 shifts

0 0 0 0 0 1 0 0 0 0 0 0

8 2 4 2 1

Final position

Convolution result

0 1 2 4 2 8 0 0

Extended (full) convolution result

0 0 0 1 2 4 2 8 0 0 0 0

Convolution in 2D

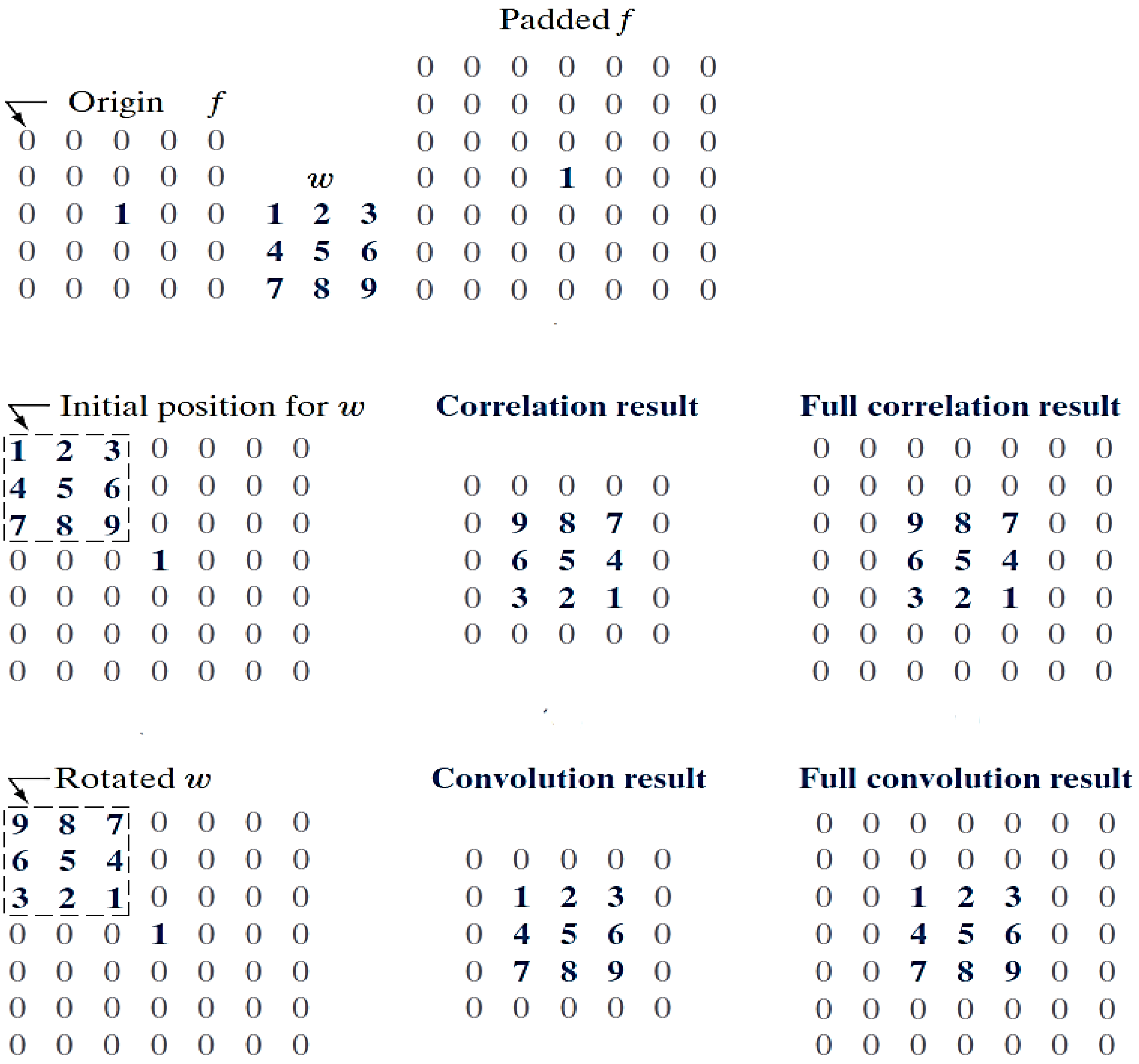
- Let f(x,y) be a **2D signal/image** of size L×L and w(x,y) be a **2D kernel** of window size (2a+1, 2b+1), where a≥1 and b≥1.

- The 2D convolution is represented by:

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

- For a kernel of size **m × n** , using zero padding, we pad the image with a minimum of **(m – 1)/2** rows of 0's at the **top** and **bottom** and **(n – 1)/2** columns of 0's on the **left** and **right**.

Convolution Vs. Correlation in 2D - Example



Origin f

$$\begin{matrix} 0&0&0&0&0\\ 0&0&0&0&0\\ 0&0&1&0&0\\ 0&0&0&0&0\\ 0&0&0&0&0 \end{matrix}$$

w

$$\begin{matrix} 1&2&3\\ 4&5&6\\ 7&8&9 \end{matrix}$$

Padded f

$$\begin{matrix} 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0 \end{matrix}$$

Initial position for w

$$\begin{matrix} 1&2&3&0&0&0&0\\ 4&5&6&0&0&0&0\\ 7&8&9&0&0&0&0\\ 0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0 \end{matrix}$$

Correlation result

$$\begin{matrix} 0&0&0&0&0\\ 0&9&8&7&0\\ 0&6&5&4&0\\ 0&3&2&1&0\\ 0&0&0&0&0 \end{matrix}$$

Full correlation result

$$\begin{matrix} 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&9&8&7&0&0\\ 0&0&6&5&4&0&0\\ 0&0&3&2&1&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0 \end{matrix}$$

Rotated w

$$\begin{matrix} 9&8&7&0&0&0&0\\ 6&5&4&0&0&0&0\\ 3&2&1&0&0&0&0\\ 0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0 \end{matrix}$$

Convolution result

$$\begin{matrix} 0&0&0&0&0\\ 0&1&2&3&0\\ 0&4&5&6&0\\ 0&7&8&9&0\\ 0&0&0&0&0 \end{matrix}$$

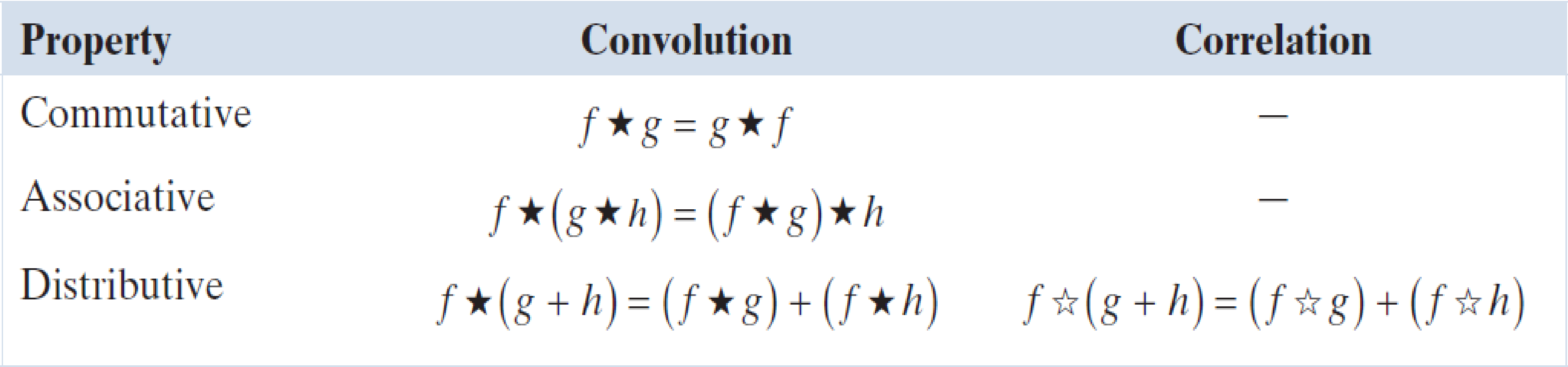
Full convolution result

$$\begin{matrix} 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&1&2&3&0&0\\ 0&0&4&5&6&0&0\\ 0&0&7&8&9&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0 \end{matrix}$$

Important Properties

- **Correlation** (☆) of a kernel with a discrete unit impulse gives a rotated version of the kernel centered at the impulse location.

- **Convolution** (★) of a kernel with a discrete unit impulse gives an exact copy of the kernel centered at the impulse location.



Property	Convolution	Correlation
Commutative	$f \bigstar g = g \bigstar f$	—
Associative	$f \bigstar (g \bigstar h) = (f \bigstar g) \bigstar h$	—
Distributive	$f \bigstar (g + h) = (f \bigstar g) + (f \bigstar h)$	$f \,☆\, (g + h) = (f \,☆\, g) + (f \,☆\, h)$

Multi-stage filtering with Convolution

- Sometimes an image is filtered (i.e., convolved) *sequentially*, in stages, using a different kernel in each stage.

- For example, suppose than an image ***f*** is first filtered with a kernel ***w1***, then the result is filtered with kernel ***w2*** , that result is filtered with a kernel ***w3*** , and so on, for ***Q*** stages.
  - Because of the **commutative property** of convolution, this multistage filtering can be done in a *single filtering operation* **w** ★ ***f*** where,

$$w = w_1 \star w_2 \star w_3 \star \cdots \star w_Q$$

Separable filter kernels

- A function ***G(x,y)*** is said to be *separable* if it can be written as the *product of two 1-D functions*, ***G_1(x)*** and ***G_2(x)****;* that is,

$$\mathbf{G(x, y) = G_1(x)\ G_2(y)}$$

- A spatial filter kernel is a *matrix*, and a separable kernel is a matrix that can be expressed as the outer product of two vectors:

$$w = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\mathbf{c} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \text{and} \quad \mathbf{r} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

That is,

$$\mathbf{c}\,\mathbf{r}^T = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = w$$

- Separable kernels have computational advantage (and thus execution-time advantage).

Separable filter kernels

- For an ***image*** of size ***M* × *N*** and a ***kernel*** of size *m* × *n*, the convolution operation requires the order of **MNmn** multiplications and additions.

- If the kernel is *separable,* then the convolution operation can be split as:

$$w \star f = (w_1 \star w_2) \star f = (w_2 \star w_1) \star f = w_2 \star (w_1 \star f) = (w_1 \star f) \star w_2$$

Step-1: The *first convolution* $(w_1 \star f)$ requires the order of **MNm** multiplications and additions because ***w1*** is of size **m × 1**. This result is of size ***M* × *N***.

Step-2: The convolution of ***w2*** with the result of *step-1* requires another **MNn** operations because ***w2*** is of size **1 × n** . The total multiplication and addition operations in *step-1* and *step-2* are just **MN(m+n)**.

Separable filter kernels

- The *computational advantage* of performing convolution with a *separable*, as opposed to a *non-separable*, kernel is defined as:

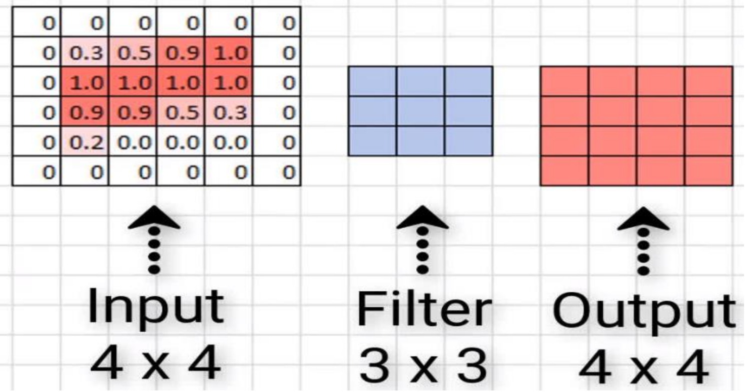
$$C = \frac{MNmn}{MN\left(m+n\right)} = \frac{mn}{m+n}$$

Example: For a kernel of size *11* × *11*, the computational advantage is **5.2**.

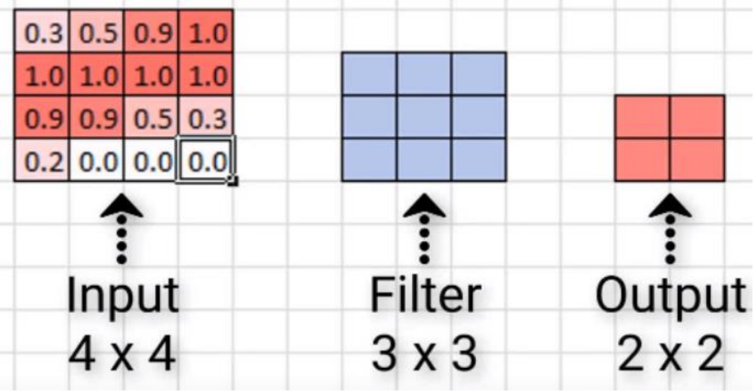
Strides, Padding, Kernel-size, and Convolution

- Stride: denotes how many steps we are moving in each steps during convolution.

- Padding: maintains the dimension of output as in input.



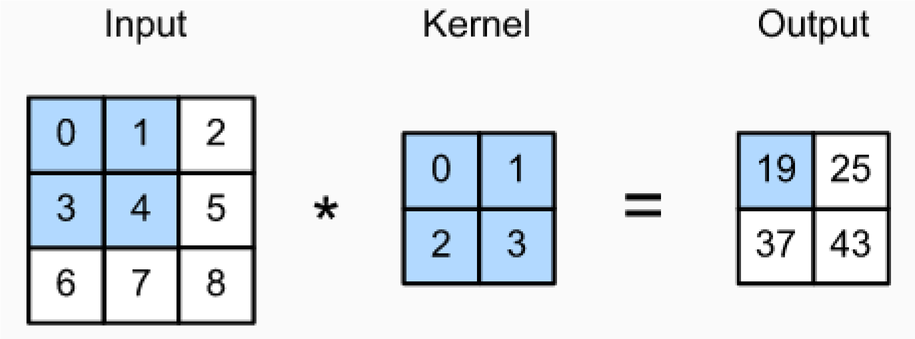
Padding with stride=1



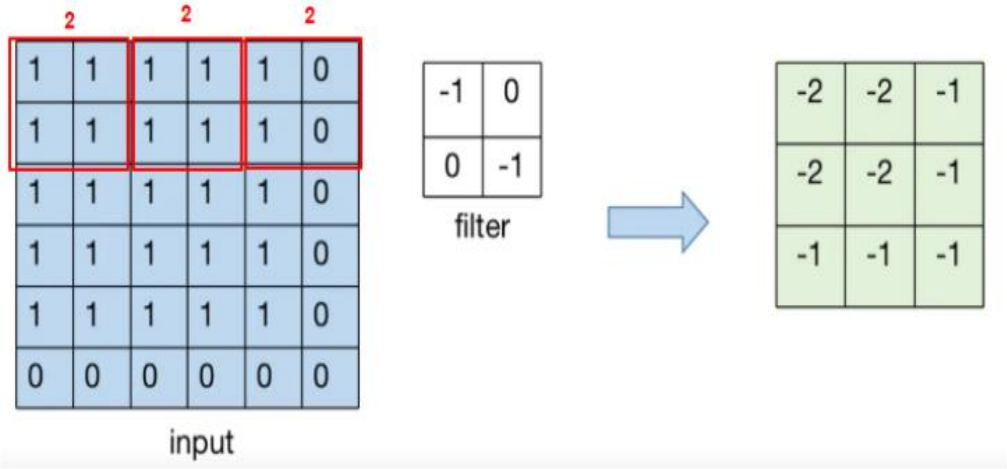
No-padding with stride=1

Strides, Padding, Kernel-size, and Convolution

- Even-sized kernel: **usually not used** for convolution



No-padding with stride=1



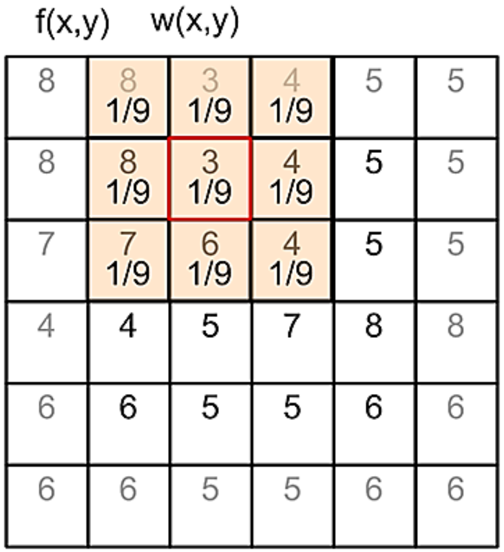
No-padding with stride=2

Vector Representation of Linear Filtering

- **Sum of product**

$$
\begin{aligned}
R &= w_1 z_1 + w_2 z_2 + \cdots + w_{mn} z_{mn} \\
&= \sum_{k=1}^{mn} w_k z_k \\
&= \mathbf{w}^T \mathbf{z},
\end{aligned}
$$

where ws are the coefficients of an $m \times n$ filter and zs are the corresponding image intensities encompassed by the filter



f(x,y) w(x,y)

8	8 1/9	3 1/9	4 1/9	5	5
8	8 1/9	3 1/9	4 1/9	5	5
7	7 1/9	6 1/9	4 1/9	5	5
4	4	5	7	8	8
6	6	5	5	6	6
6	6	5	5	6	6

$$\mathbf{w} = \left[\frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}\right]$$

$$\mathbf{z} = [8, 3, 4, 8, 3, 4, 7, 6, 4]$$

Shift Invariant Operations

- Let **H** be a general operator that produces **g(x,y)** for a given image **f(x,y)**

$$f(x,y):\ H[f(x,y)]=g(x,y)$$

- H is shift invariant if:

$$H[f(x+x_0,y+y_0)]=g(x+x_0,y+y_0)$$

Shift Invariant Operations - Examples

- $H[f(x,y)]=f(x-a,y-b)$

- $H[f(x,y)]=[f(x,y)]^2$

- $H[f(x,y)]=f(M\times x,N\times y)$, where $M,N \in Z+$

- $H[f(x,y)]=af(x,y)+b$, where a,b are arbitrary scalars

Summary

- Both Correlation & Convolution are the simplest image operations that can be performed in the spatial domain.

- Linear - Output pixel value is a linear combination of input pixel values.

- Shift invariant - Have same behavior at any location on the image plane

How to construct Spatial filter masks?

Constructing Spatial Filter Masks

Mainly there are **three** basic approaches for constructing spatial filters:

1. Formulating filters based on *mathematical properties*.
   - Averaging filter

2. Based on *sampling* a 2-D spatial function whose *shape has a desired property*.
   - Gaussian filter

3. Based on a specified *frequency response*.
   - max/median/min filters

Smoothing (Lowpass) Linear Spatial Filters

Smoothing spatial filters

- **Smoothing** (also called ***averaging***) spatial filters are used to *reduce sharp transitions in intensity*.

- **Applications:**
  - Noise reduction in an image.
  - Reduce irrelevant detail in an image where "irrelevant" refers to pixel regions that are small with respect to the size of the filter kernel.
  - Smoothing the false contours that result from using an insufficient number of intensity levels in an image
  - Used in combination with other techniques for image enhancement

Box filter (Arithmetic Mean Filter) kernels

- These are the simplest, separable, lowpass **averaging** filter kernels whose coefficients have the **same value** (typically **1**).

- An ***m* × *n*** box filter is an ***m* × *n* array of 1's**, with a *normalizing constant in front*, whose value is *1 divided by the sum of the values of the coefficients*.

$$\frac{1}{9} \times \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \qquad \frac{1}{4.8976} \times \begin{array}{|c|c|c|}\hline 0.3679 & 0.6065 & 0.3679 \\ \hline 0.6065 & 1.0000 & 0.6065 \\ \hline 0.3679 & 0.6065 & 0.3679 \\ \hline \end{array}$$

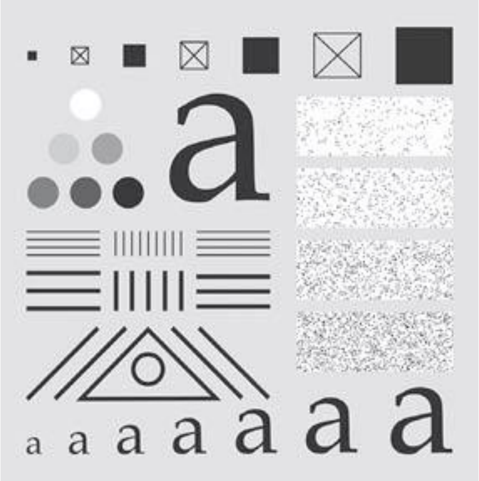
Box filter kernels

- The *normalizing constant* has two purposes:

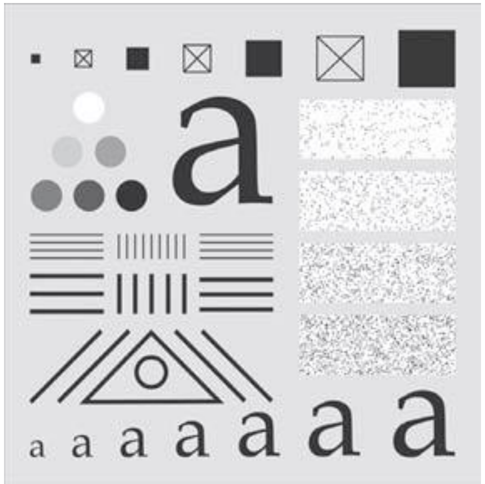
1. *The average value of an area of constant intensity* will **remain the same in the filtered image**, as it should be.

2. It *prevents introducing a bias* during filtering; that is, **the sum of the pixels in the original and filtered images will be the same**.

Box filter kernels

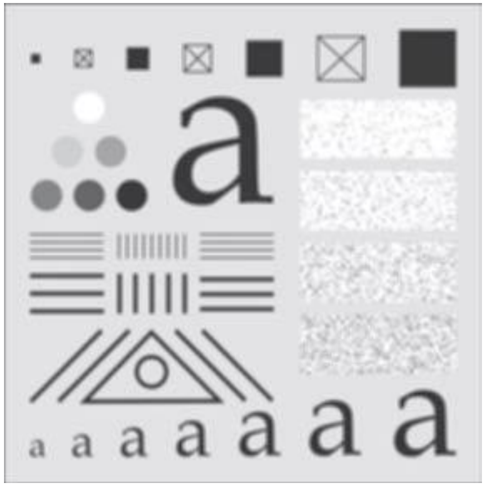


Test pattern of size 1024 x 1024 pixels

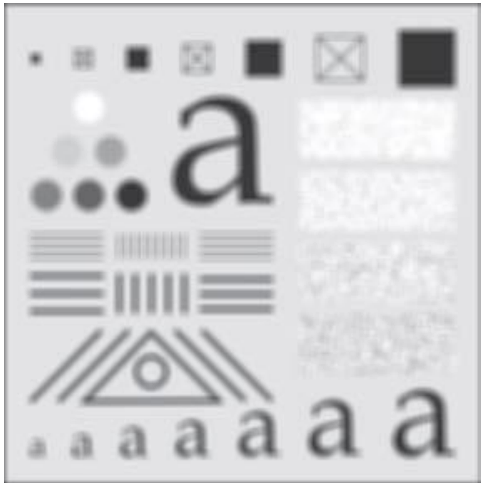


Result-Box kernel of size 3 × 3

Zero padding is used



Result-Box kernel of size 11 × 11



Result-Box kernel of size 21 × 21

Box filter kernels - Limitations

1. *Poor approximations* to the blurring characteristics of lenses.

2. *Favor blurring* along perpendicular directions.

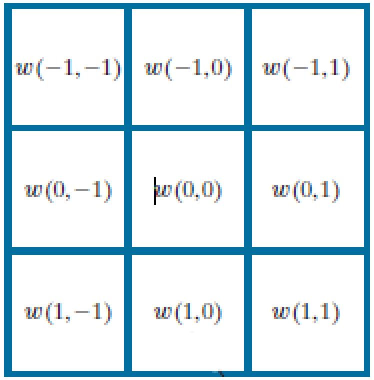
- **The kernels of choice in situations such as these** are ***circularly symmetric kernels*** (also called *isotropic*, meaning their response is independent of orientation) – **Gaussian Kernel**

Gaussian filter kernels

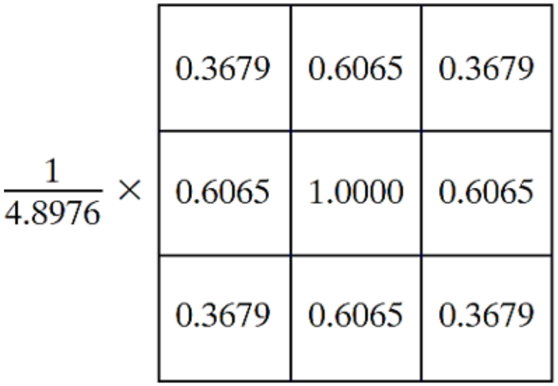
- *Gaussian kernels* of the following form are the only ***averaging*** *circularly symmetric kernels that are also separable*:

$$w(s,t) = G(s,t) = Ke^{-\frac{s^2+t^2}{2\sigma^2}}$$

where ***s*** and ***t*** are kernel coordinates, ***K*** is a constant and ***σ*** is the standard deviation.



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



$\frac{1}{4.8976} \times$

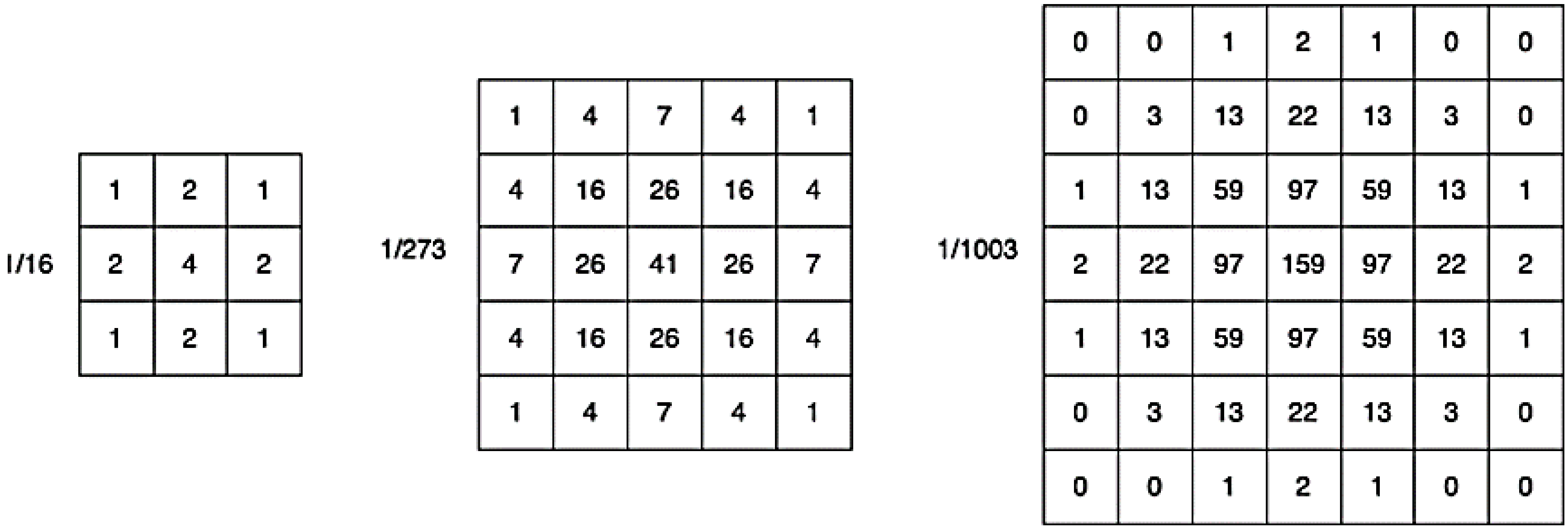
0.3679	0.6065	0.3679
0.6065	1.0000	0.6065
0.3679	0.6065	0.3679

K = ***σ*** = 1

Gaussian filter kernels

A Gaussian kernel gives less weight to pixels further from the center of the window and vice-versa.

$$w(s,t) = G(s,t) = Ke^{-\frac{s^2 + t^2}{2\sigma^2}}$$



I/16

1	2	1
2	4	2
1	2	1

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

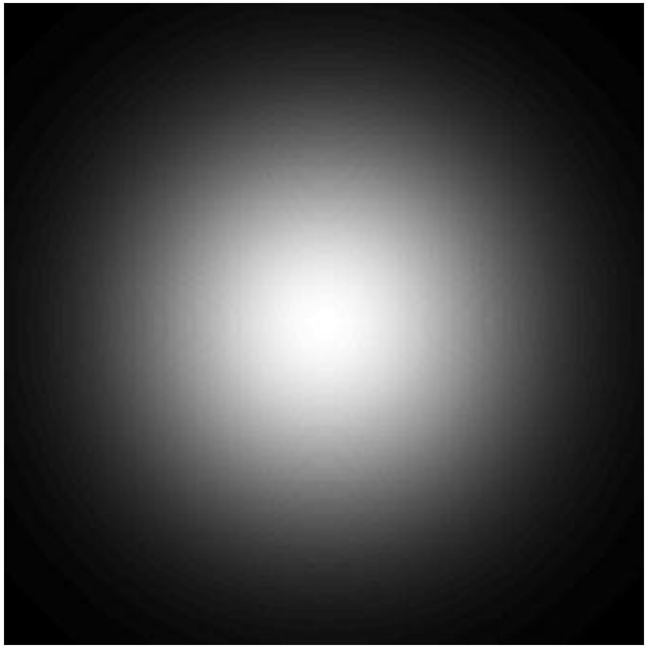
1/1003

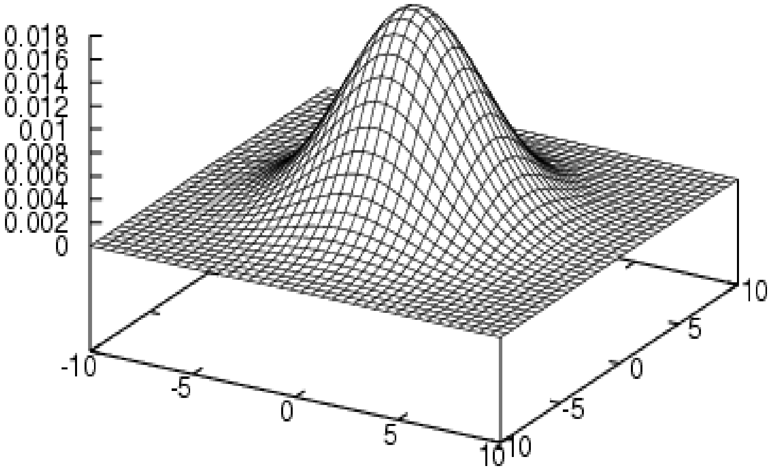
0	0	1	2	1	0	0
0	3	13	22	13	3	0
1	13	59	97	59	13	1
2	22	97	159	97	22	2
1	13	59	97	59	13	1
0	3	13	22	13	3	0
0	0	1	2	1	0	0

Gaussian filter kernels

A Gaussian kernel gives less weight to pixels further from the center of the window and vice-versa.

$$w(s,t) = G(s,t) = Ke^{-\frac{s^2+t^2}{2\sigma^2}}$$

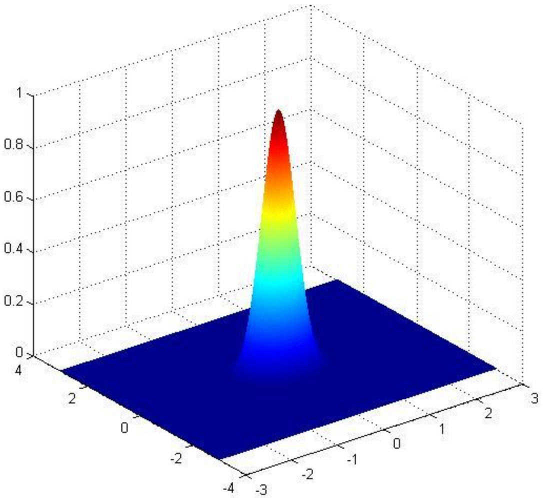




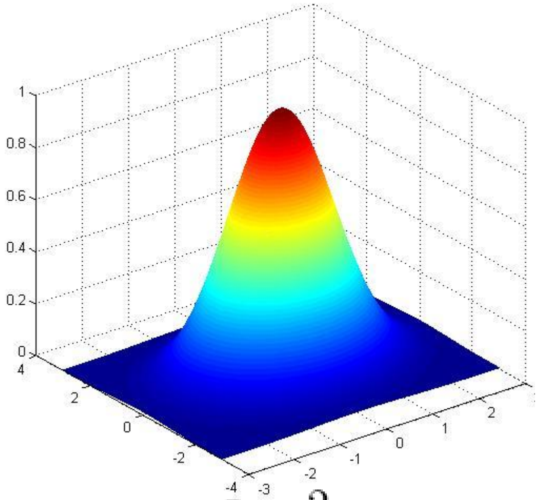
Gaussian filter kernels

$$w(s,t) = G(s,t) = Ke^{-\frac{s^2+t^2}{2\sigma^2}}$$

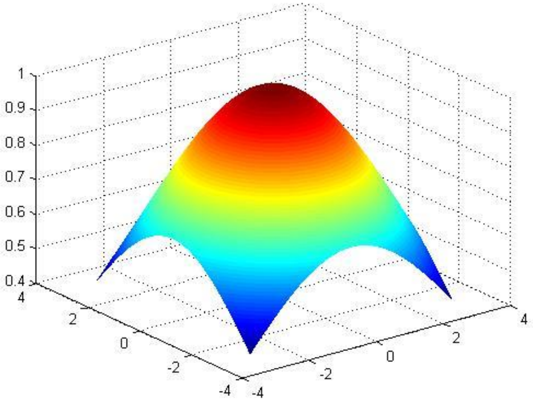
The σ term controls the “tightness” of the function



$\sigma = 0.2$



$\sigma = 2$

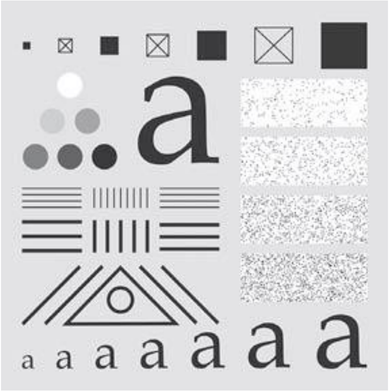


$\sigma = 20$

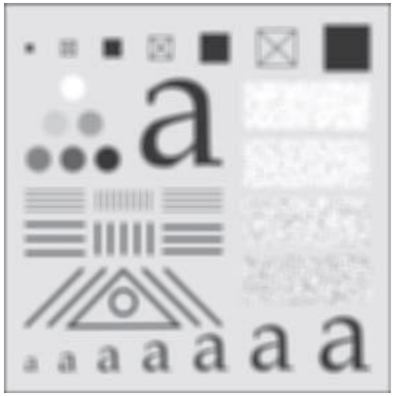
Low pass filtering with a Gaussian kernel

- Gaussian kernels have to be larger than box filters to achieve the same degree of blurring.

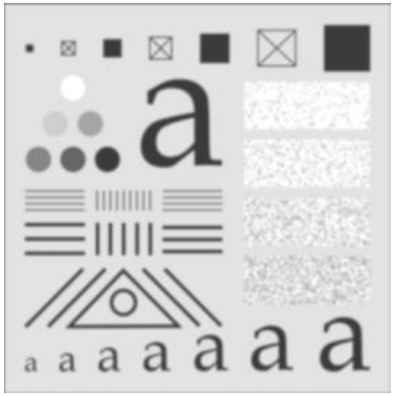
- This is because, whereas a **box kernel** assigns the same weight to all pixels, the values of **Gaussian kernel** coefficients (and hence their effect) decreases as a function of distance from the kernel center.



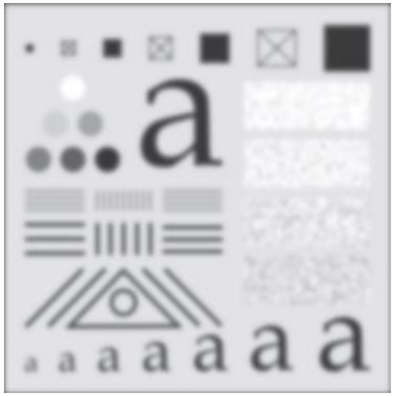
Test pattern of size 1024 x 1024 pixels



Box kernels of sizes 21 × 21



Gaussian kernels of sizes 21 × 21 and σ=3.5



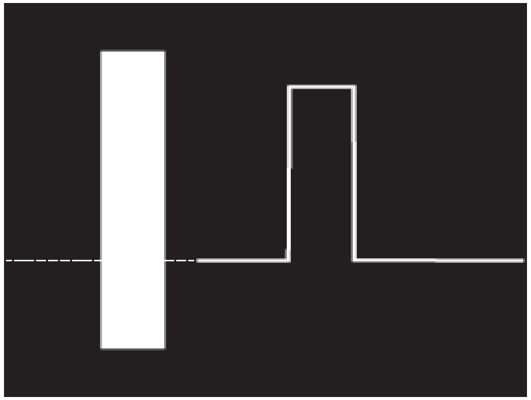
Gaussian kernels of sizes 43 × 43 and σ=7

K=1 and **Zero padding** is used in all cases

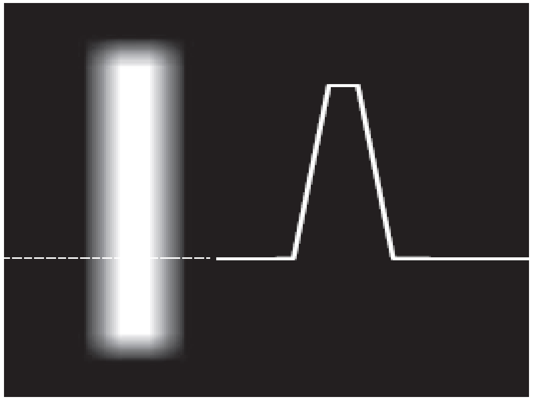
Comparison of Gaussian and Box filter smoothing characteristics

- Box filter:
  - The box filter produces linear smoothing, with the transition from black to white (i.e., at an edge) having the shape of a ramp.
  - The important features here are hard transitions at the onset and end of the ramp.
  - We would use this type of filter when less smoothing of edges is desired.

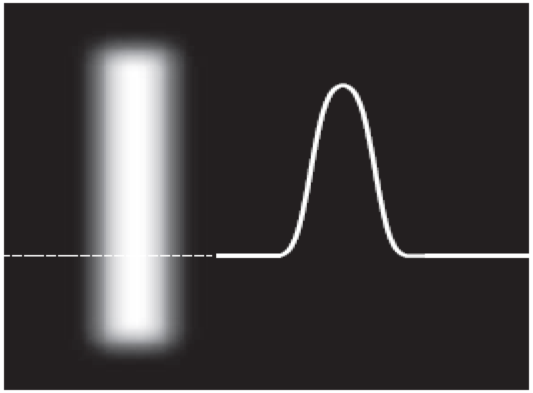
- Gaussian filter:
  - The Gaussian filter yields significantly smoother results around the edge transitions.
  - We would use this type of filter when generally uniform smoothing is desired.



Test image



Box kernel of size 71 × 71



Gaussian kernel of size 151 × 151, with K = 1 and σ = 25.

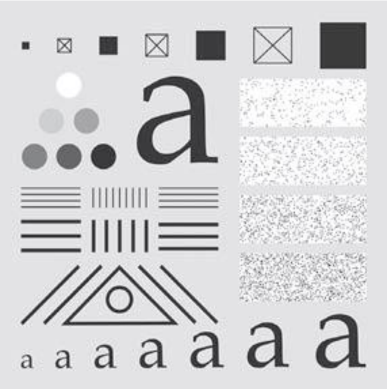
Comparison of Zero/Mirror/Replicate padding

- *Replicate padding* is useful when the *areas near the border of the image are constant*.

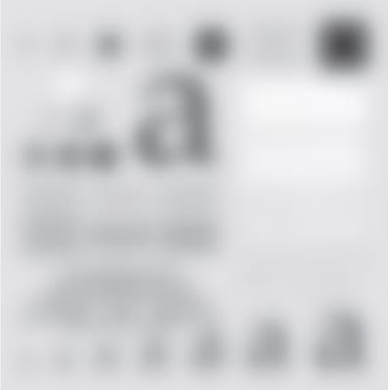
- *Mirror padding* is more applicable when the *areas near the border contain image details*.

- These two types of padding attempt to **“extend”** the characteristics of an image past its borders.

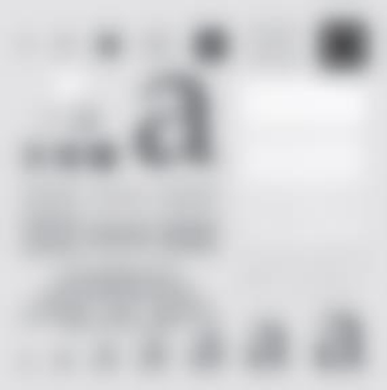
- *Zero padding* introduces **dark borders** after filtering which can be eliminated by other two types of padding.



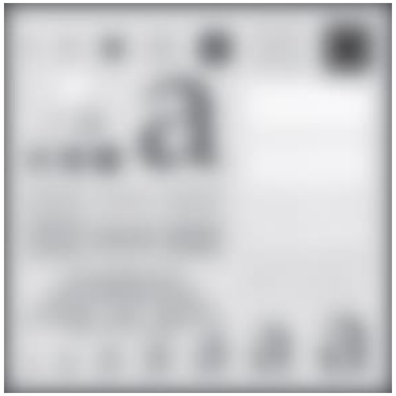
Test image



Replicate padding



Mirror padding



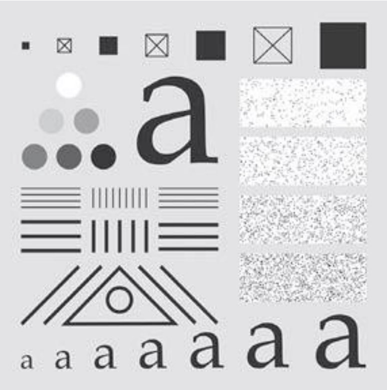
Zero padding

A **Gaussian kernel** of size **187 x 187**, with ***K* = 1** and **σ = 31** was used in all three cases

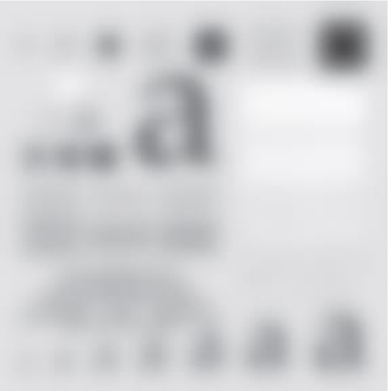
Smoothing performance as a function of kernel and image size

- The amount of relative blurring produced by a smoothing kernel of a given size depends directly on image size.

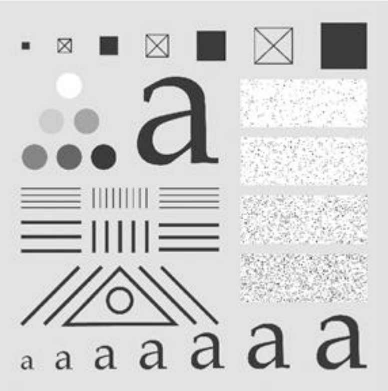
- Not understanding the relationship between kernel size and the size of objects in an image can lead to ineffective performance of spatial filtering algorithms.



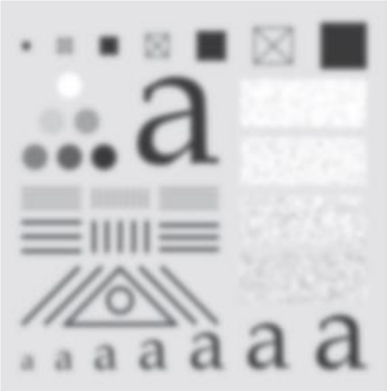
Test image of size 1024 x 1024 (a)



Gaussian kernel of size **187 x 187**, with ***K* = 1** and **σ = 31** (b)



Test image of size 4096 x 4096 (c)

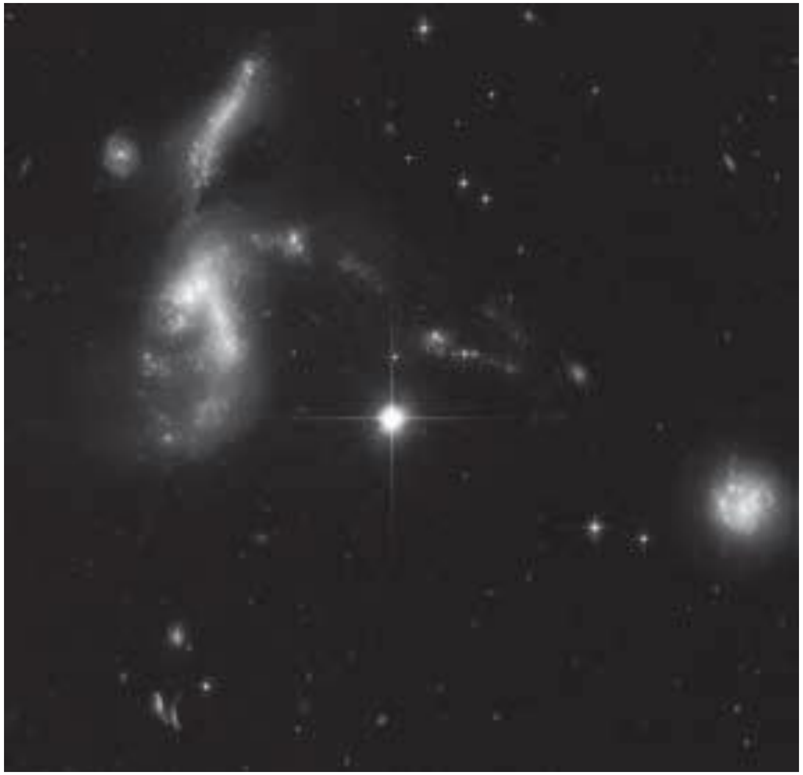


Gaussian kernel of size **187 x 187**, with ***K* = 1** and **σ = 31** (d)

To obtain results that are comparable to Fig. (b) we have to **increase the size and standard deviation** of the Gaussian kernel by **four**, the same factor as the increase in image dimensions (i.e. **745 × 745** (with ***K* = 1** and **σ = 124**).

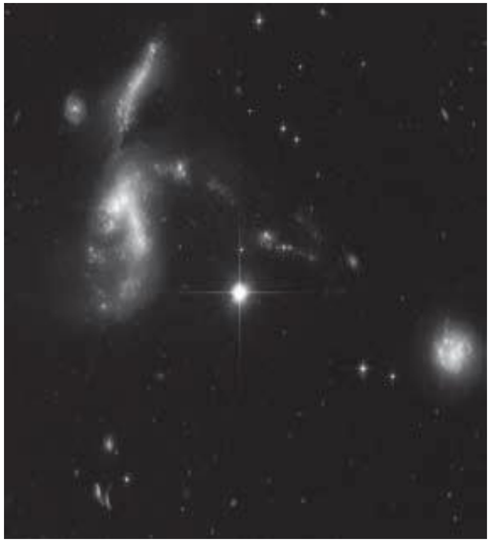
Using lowpass filtering and thresholding for region extraction

- *Lowpass filtering* can be combined with *Intensity thresholding* for eliminating ***irrelevant detail*** in this image. In the present context, "irrelevant" refers to *pixel regions that are small compared to kernel size*.



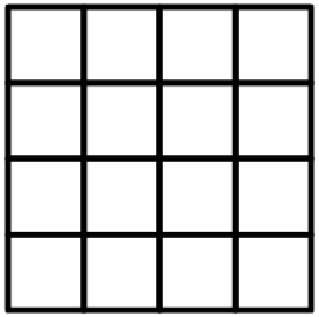
A 2566 × 2758 image from the Hubble telescope

Using lowpass filtering and thresholding for region extraction



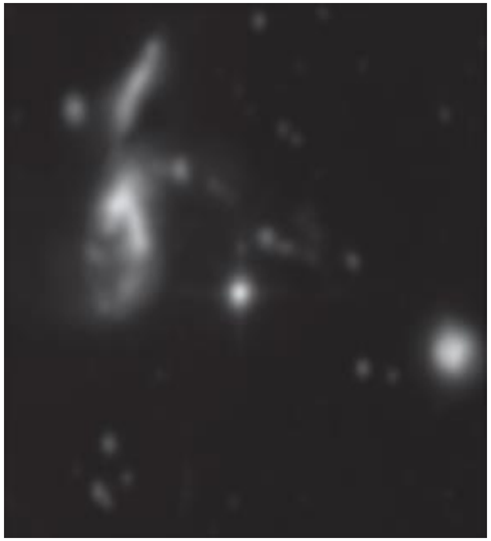
2566 × 2758 Image from the Hubble telescope

★



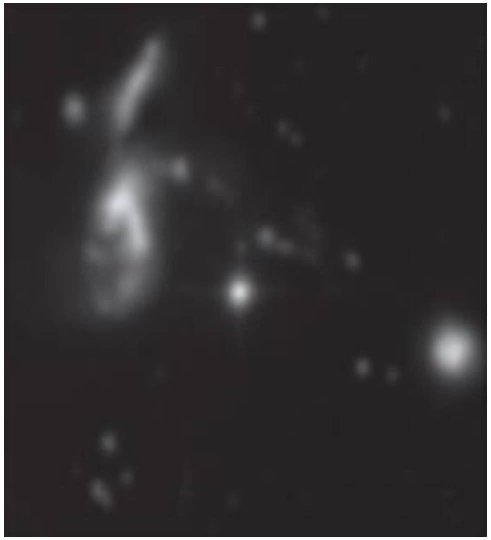
Gaussian kernel of size **151 × 151**, **K=1**, ***σ*** = 25.

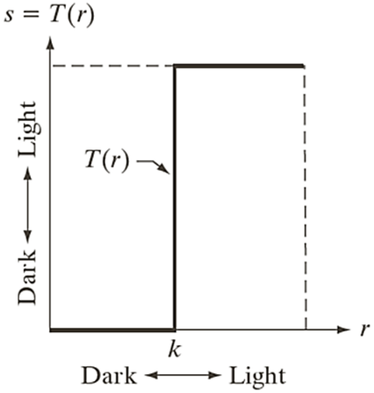
=



Result of blurring

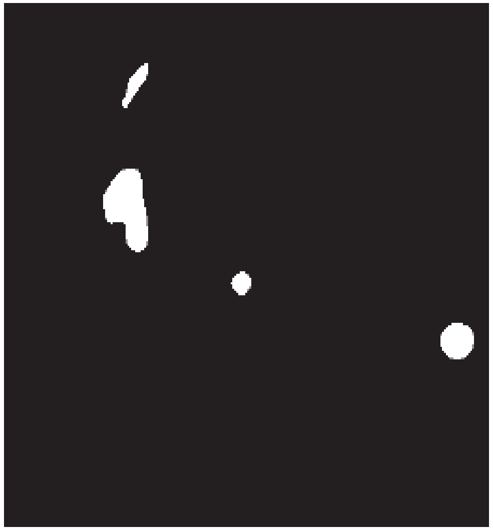
Using lowpass filtering and thresholding for region extraction



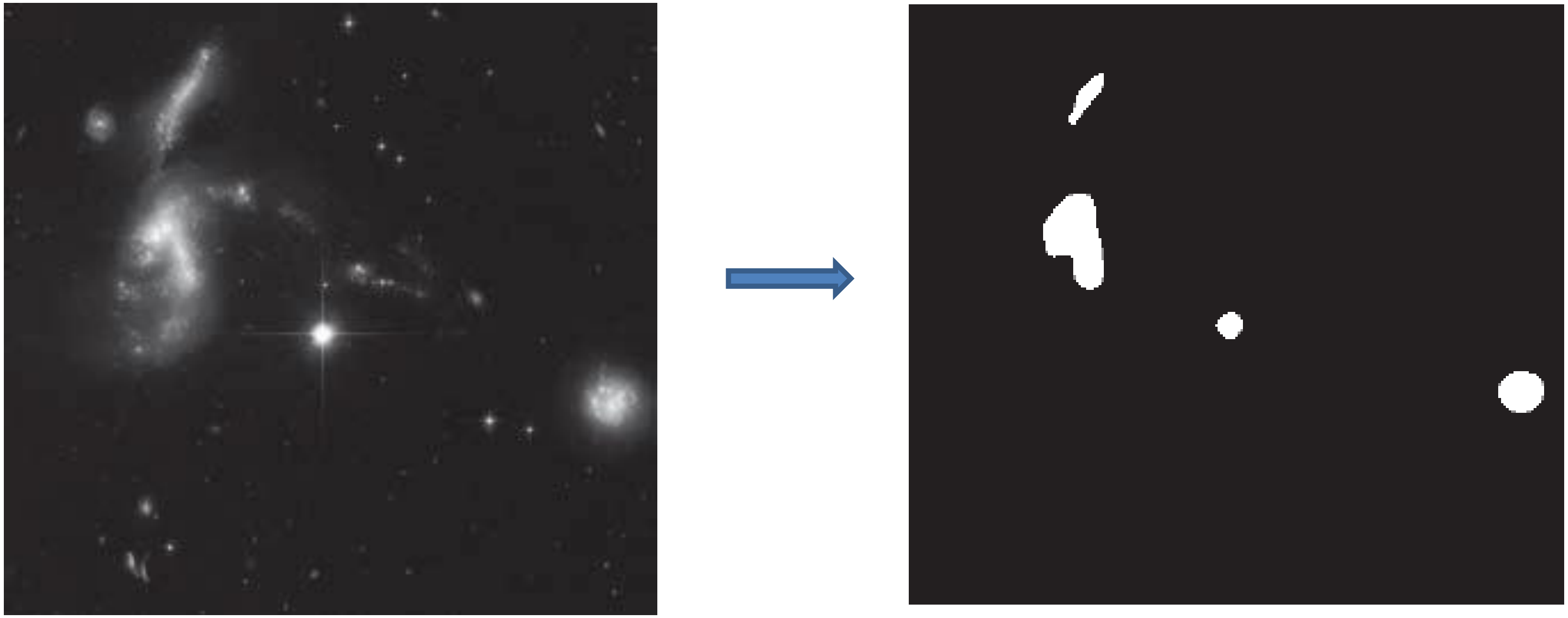


T=25%×max[f(x,y)]

T = 0.4



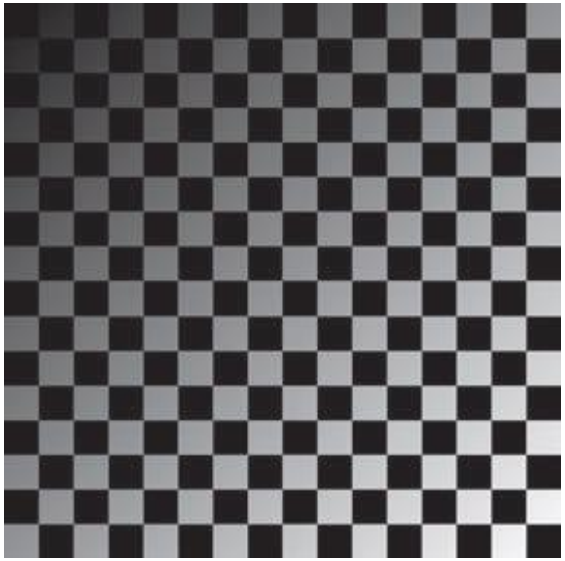
Using lowpass filtering and thresholding for region extraction



Shading correction using lowpass filtering

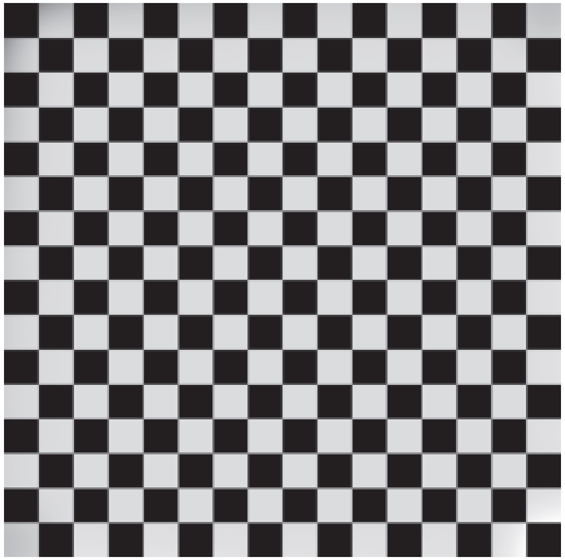
- When shading pattern is unknown, we have to estimate the pattern directly from available samples of shaded images.

- *Lowpass filtering* is a simple method for *estimating shading patterns*.



2048 × 2048 Image shaded by a shading pattern oriented in the **−45°** direction (a)

Estimate of the shading patterns using a **512 × 512** Gaussian kernel (four times the size of the squares), $K = 1$, and $\sigma = 128$ (b)



Result of dividing (a) by (b)

Smoothing Non-linear Filters (Order-static Non-linear Filters)

Order Statistics

- The **kth** **order statistic** of a sequence of intensity values is equal to its **kth smallest value**.

- **E.g.** $[5,7,6,4,2]$

  Order statistics: $\mathbf{x_{(1)}=2, x_{(2)}=4, x_{(3)}=5, x_{(4)}=6, x_{(5)}=7}$

- The **first** order statistic is the minimum.

- The **nth** order statistic is the maximum.

Order-Statistic Filters

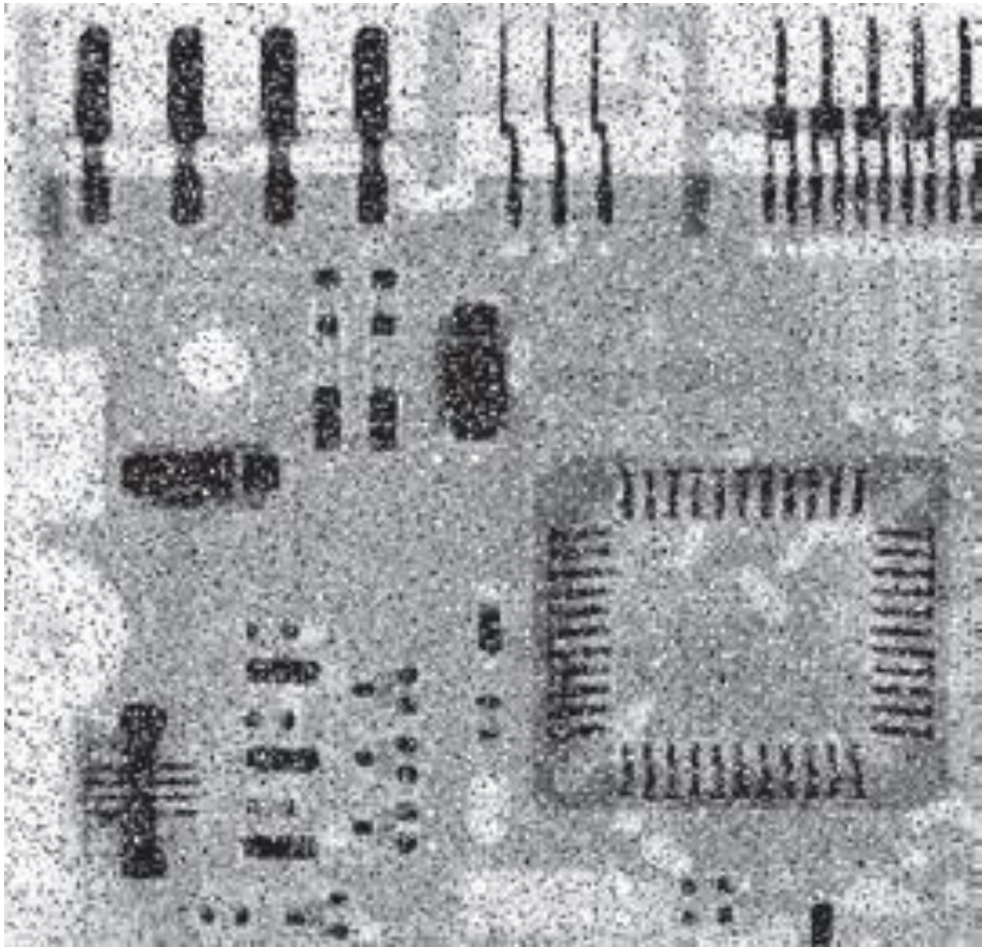
- *Order-statistic filters* are **nonlinear spatial filters** whose response is based on **ordering** (ranking) the pixels contained in the region encompassed by the filter.

- Median filter:
  - Replaces the value of the center pixel by the **median** of the *intensity values in the neighborhood of that pixel* (the value of the center pixel is included in computing the median).
  - Particularly effective in the presence of **impulse noise** (salt and pepper noise).
  - We first sort the values of the pixels in the neighborhood, determine their median, and assign that value to the pixel in the filtered image corresponding to the center of the neighborhood.
  - When several values in a neighborhood are the same, all equal values are grouped.

Order-Statistic Filters

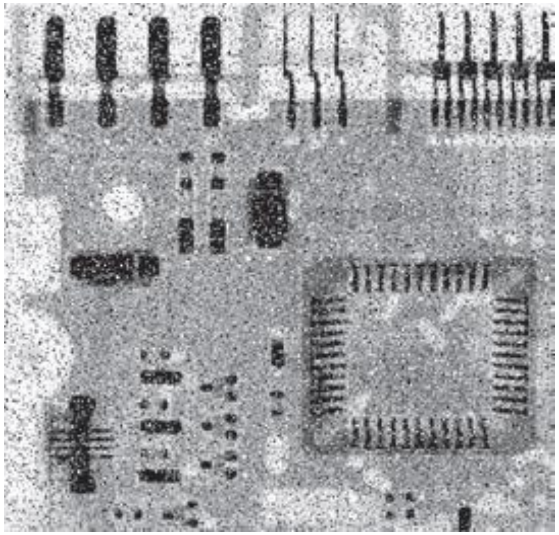
- Median filter:
  - For example, suppose that a 3 × 3 neighborhood has values (10, 20, 20, 20, 15, 20, 20, 25, 100). These values are sorted as (10, 15, 20, 20, 20, 20, 20, 25, 100), which results in a **median of 20**.
  - The main function of median filter is to *force points to be more like their neighbors*.
- Max filter:
  - Useful for finding the **brightest points** in an image or for **eroding** dark areas adjacent to bright regions.
- Min filter:
  - Useful for finding the **darkest points** in an image or for **eroding** bright areas adjacent to dark regions.

Median Filter Example

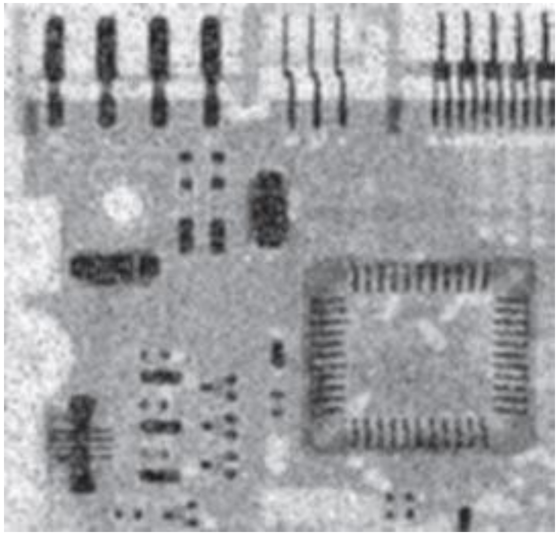


X-ray image of circuit board corrupted by impulse noise (salt-pepper noise)

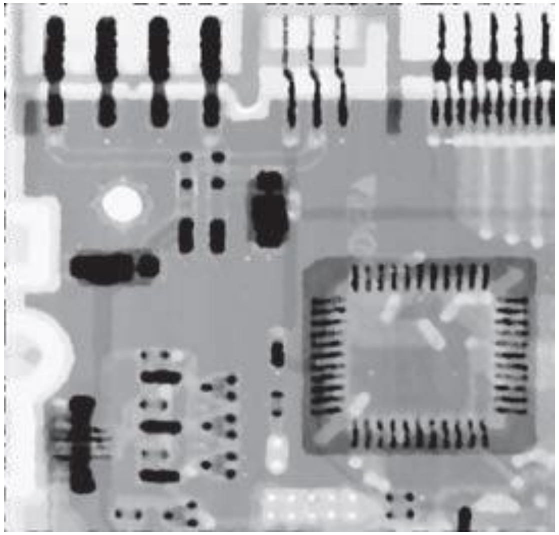
Median Filter Example



X-ray image of circuit board corrupted by impulse noise



A **19 × 19** Gaussian filter with ***K*** **= 1**, and $\boldsymbol{\sigma}$ **= 3**



A **7 × 7** Median filter

Next Lecture

- Sharpening spatial filters
  - Foundation of image sharpening
  - Using second-order derivative for image sharpening (Laplacian)
  - Unsharp masking and high boost filtering
  - Using first-order derivative for image sharpening (Gradient)
  - Highpass, Bandreject, and Bandpass filters from lowpass filters
  - Combining special enhancement methods