CSE4227 Digital Image Processing

Chapter 03 – **Smoothing Spatial Filtering**

Dr. Kazi A Kalpoma

Professor, Department of CSE

Ahsanullah University of Science & Technology (AUST)

Contact: kalpoma@aust.edu

Today’s Contents

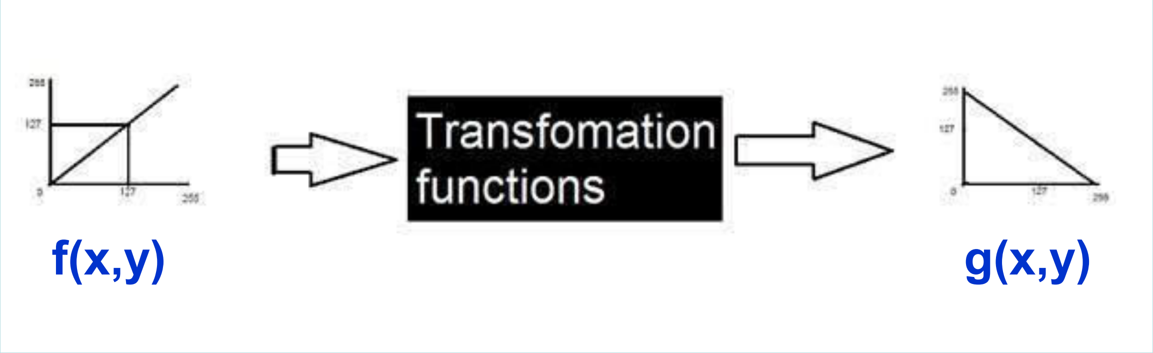
- Convolution Process
- What is Spatial filtering? Linear and Nonlinear filter.
- Why filters are used?
- Types of filters
- Smoothing operations
- Applications of Coefficient filtering

Chapter 3 from R.C. Gonzalez and R.E. Woods, Digital Image Processing (3rd Edition), Prentice Hall, 2008 [**Section 3.4, 3.5**]

Where have we reached until now!

Till now we have discussed two important methods to manipulate images.

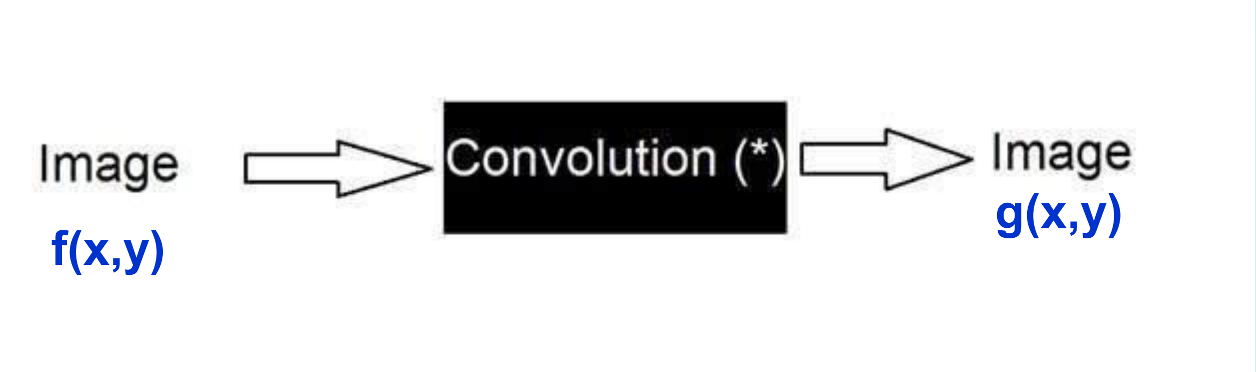
1. **Transformation Functions**



2. **Histogram Processing**

Another way of dealing images

So now we are going to use this third method, known as convolution. It can be represented as



It can be mathematically represented **as two ways**

g(x,y) = h(x,y) * f(x,y)

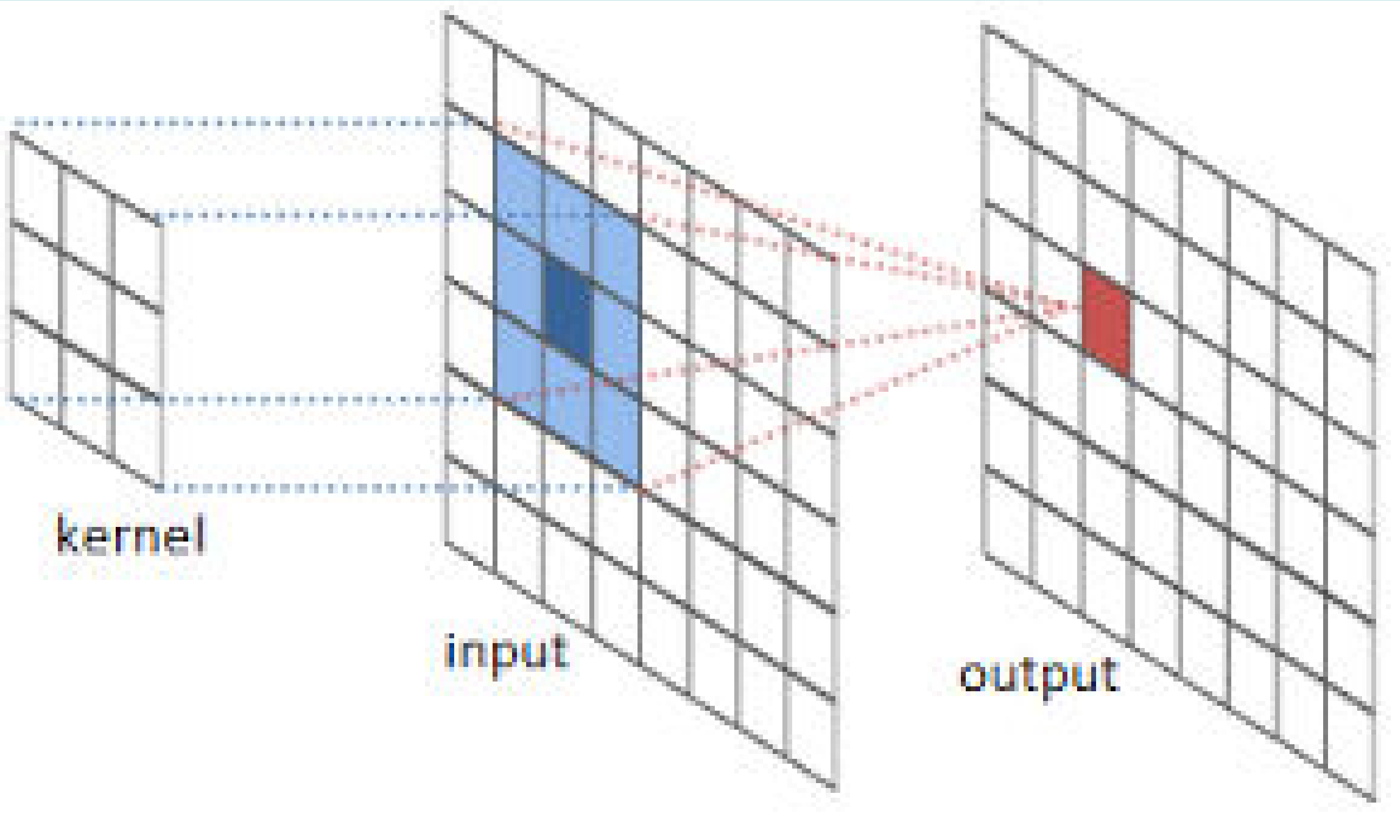
It can be explained as the "mask convolved with an image".

Or

g(x,y) = f(x,y) * h(x,y)

It can be explained as "image convolved with mask".

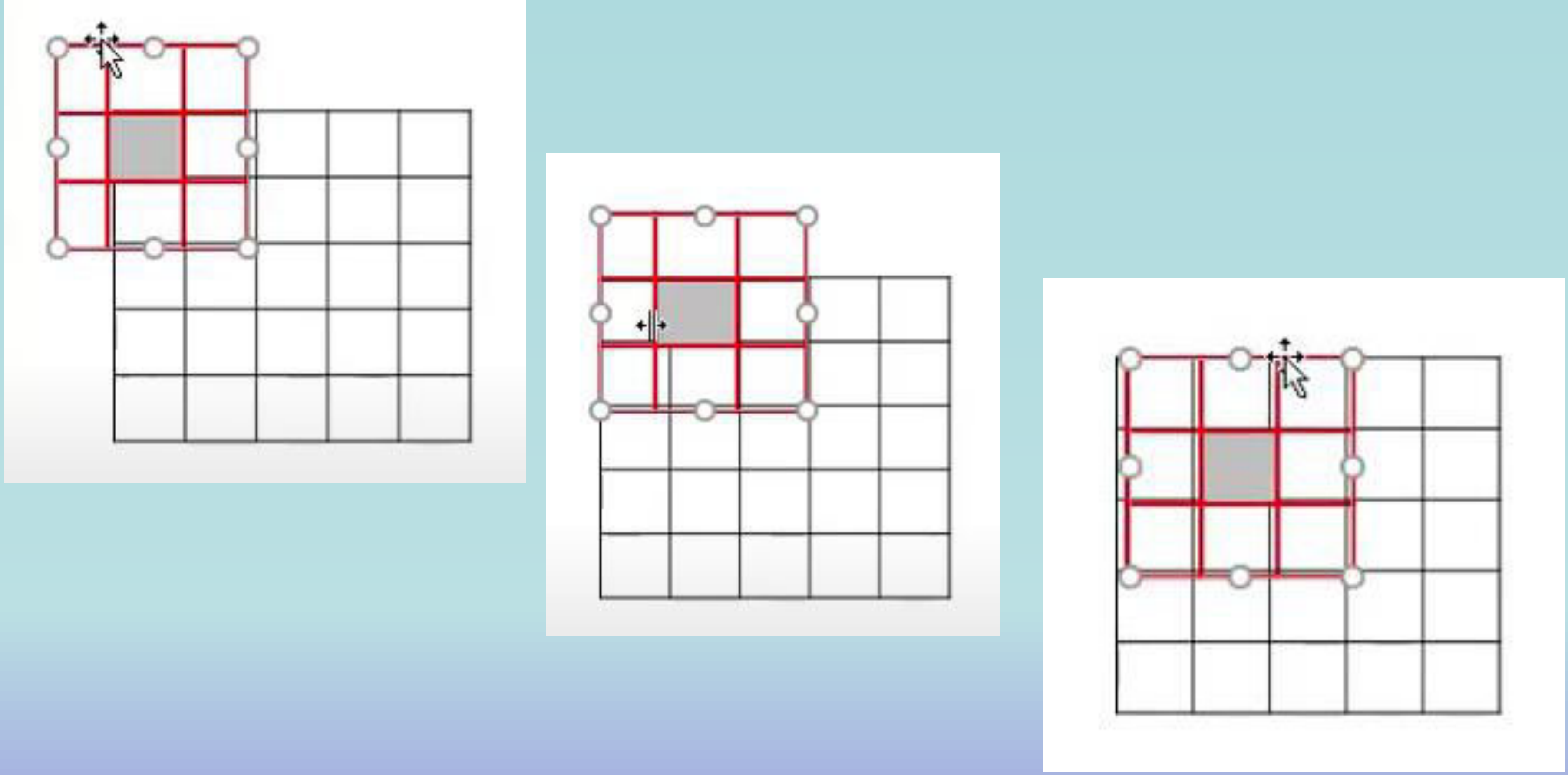
Kernel Convolved with an Image



What is kernel/mask?

- It can be represented by a two dimensional matrix.
- The mask is usually of the order of 1x1, 3x3, 5x5, 7x7 .
- A mask should always be in odd number, because other wise you cannot find the mid of the mask.
- It also known as filter/mask/window/kernel/box

Convolution is the process of moving a filter/mask/window/kernel **over** the image



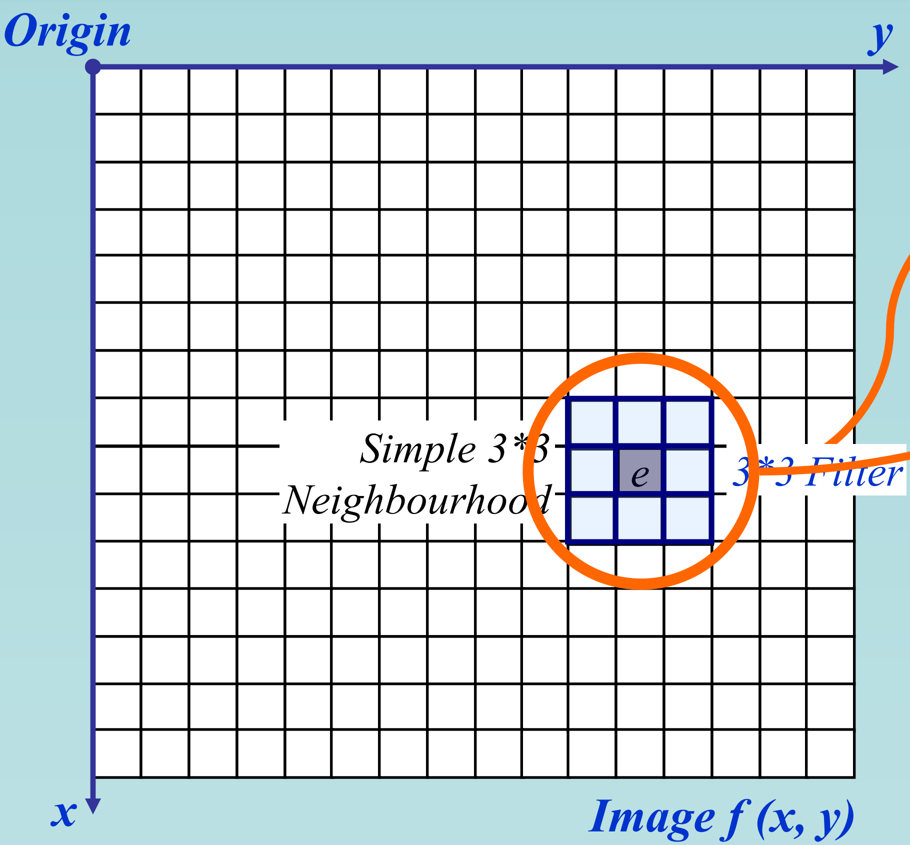
How to perform Convolution?

Matlab's im2conv() function

In order to perform convolution on an image, following steps should be taken.

- **Multiply** the corresponding elements and then
- **Add** them to computing the ***sum of product.***
- **Repeat** this procedure until all values of the image has been calculated.

Convolution Process



a	b	c
d	e	f
g	h	i

Original Image Pixels

$*$

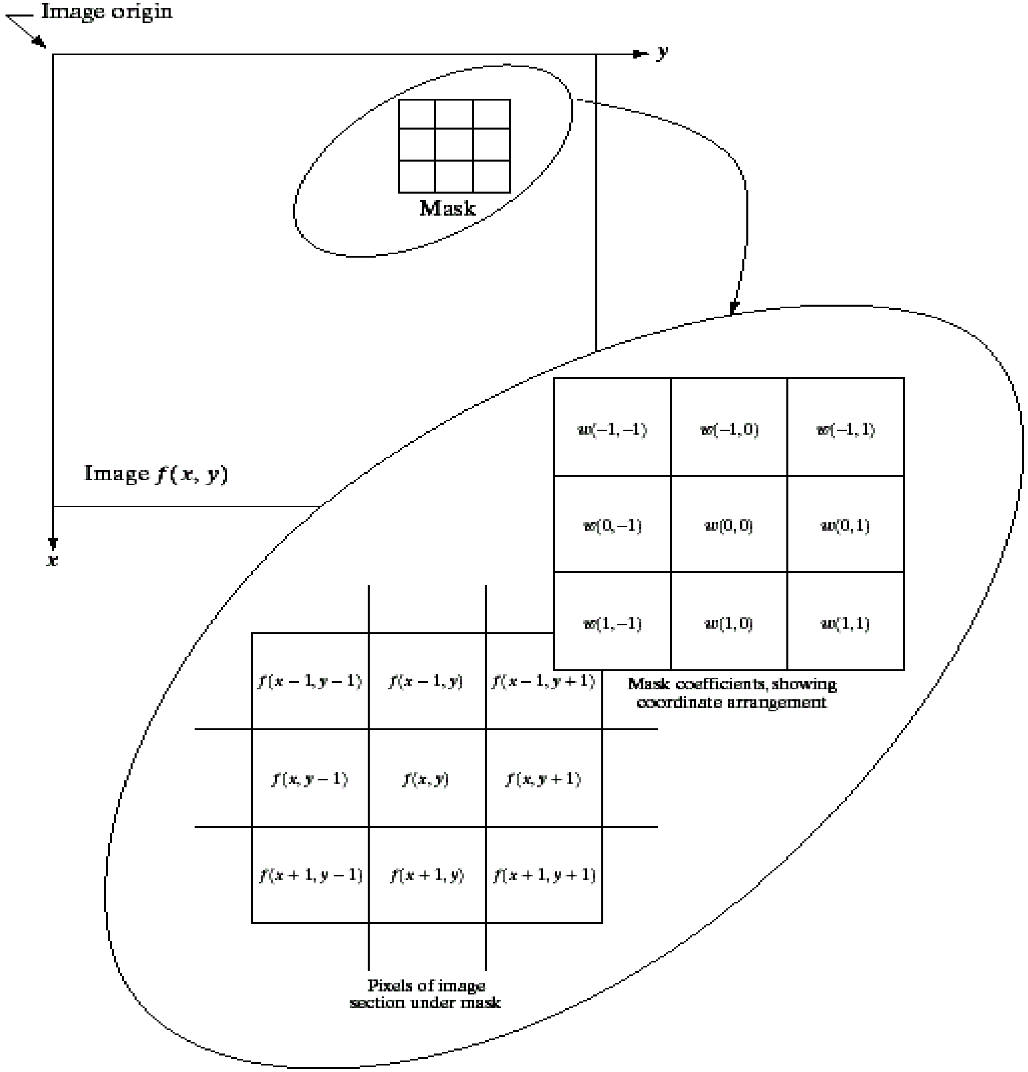
r	s	t
u	v	w
x	y	z

Filter

$$
\begin{aligned}
e_{processed} = {} & v*e + \\
& r*a + s*b + t*c + \\
& u*d + w*f + \\
& x*g + y*h + z*i
\end{aligned}
$$

Sum of Product

Equation Form



This can be given in equation form as shown in the next slide.

Equation Form

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

where $a = \dfrac{m-1}{2}$, $b = \dfrac{n-1}{2}$

$x = 0,1,2,....,M-1,\ y = 0,1,2,...,N-1$

f = Image

h = Kernel

g(x,y) = h(s,t) * f(x,y)

f

f_1	f_2	f_3
f_4	f_5	f_6
f_7	f_8	f_9

$\otimes$

h

h_1	h_2	h_3
h_4	h_5	h_6
h_7	h_8	h_9

Sum of Product

$$\begin{aligned} f * h = {} & f_1 h_1 + f_2 h_2 + f_3 h_3 \\ & + f_4 h_4 + f_5 h_5 + f_6 h_6 \\ & + f_7 h_7 + f_8 h_8 + f_9 h_9 \end{aligned}$$

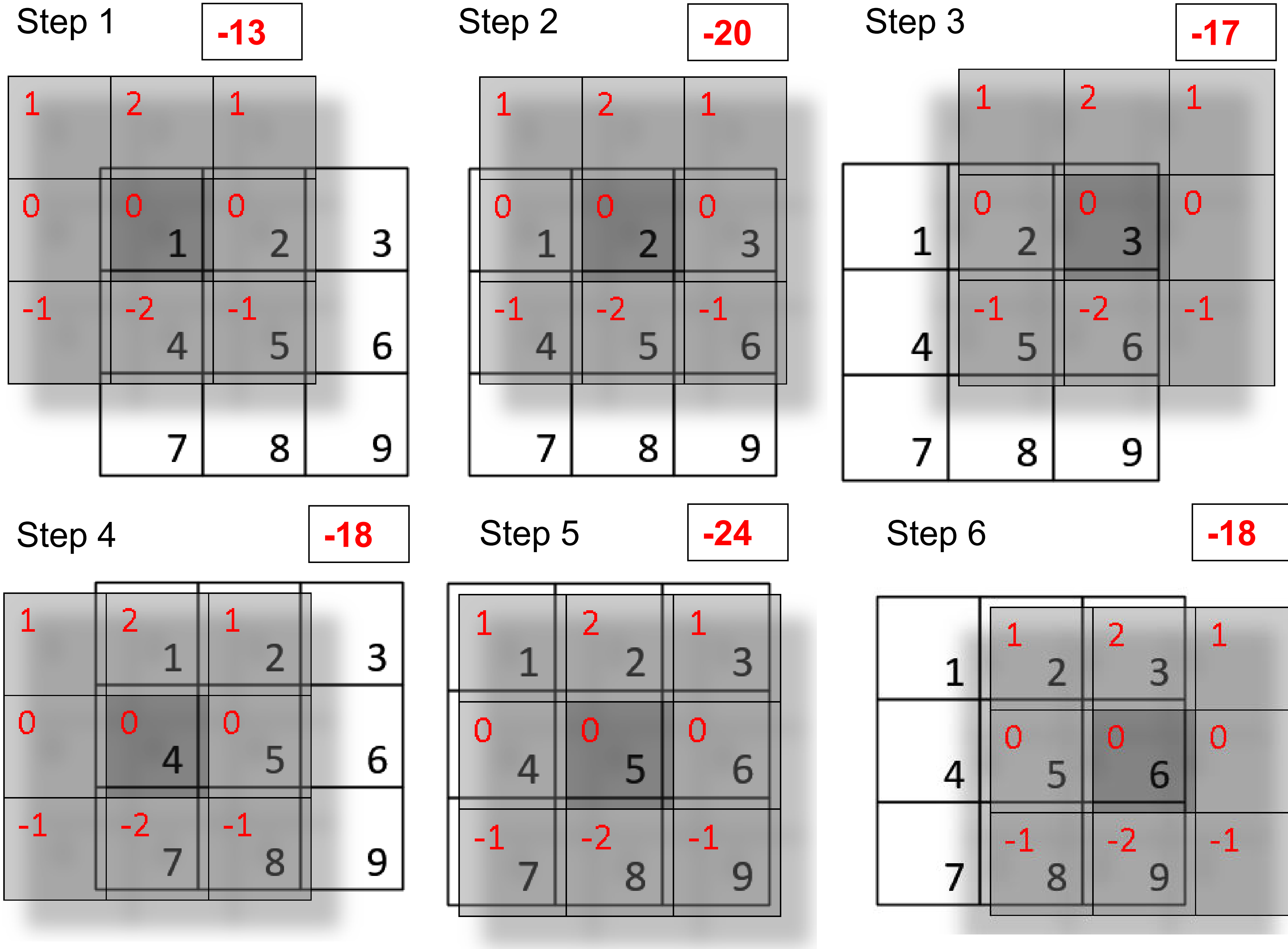
Example of Convolution

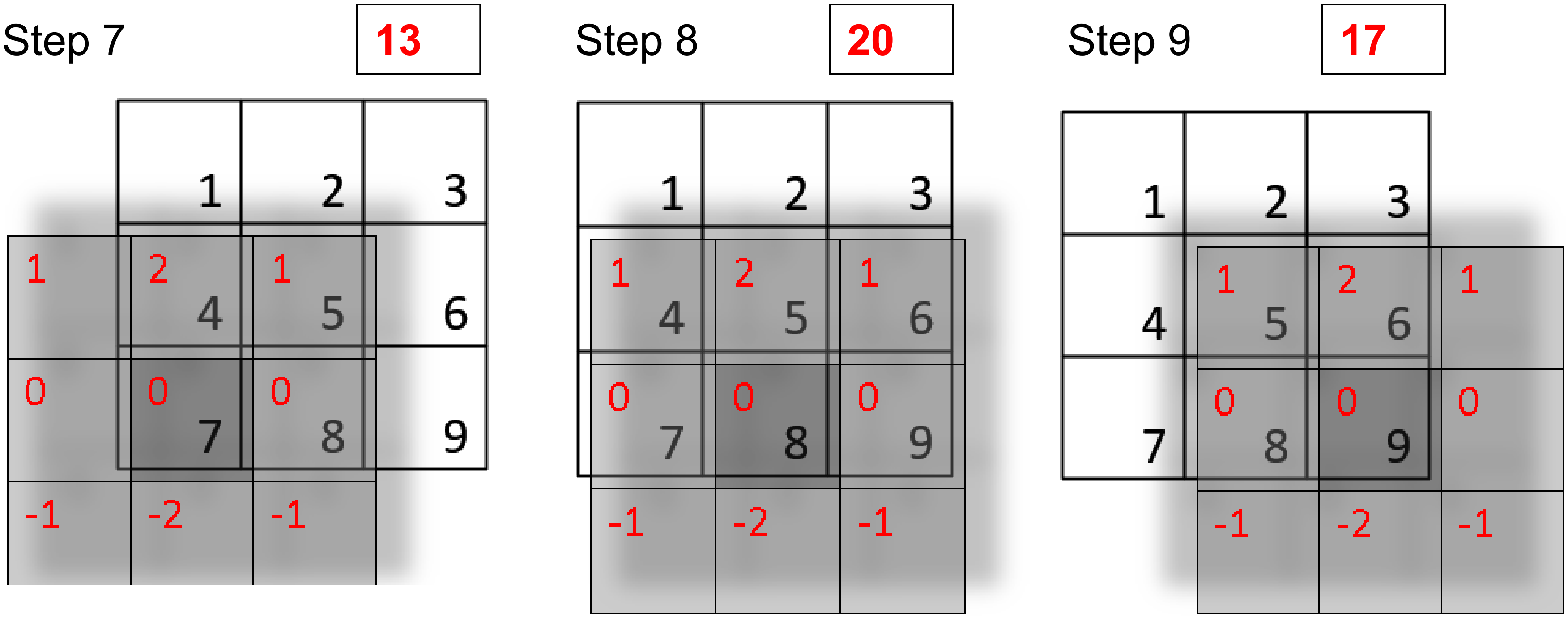
1	2	3
4	5	6
7	8	9

Input image

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

Kernel


Step 1
-13
Step 2
-20
Step 3
-17
Step 4
-18
Step 5
-24
Step 6
-18



1	2	3
4	5	6
7	8	9

Input image

-13	-20	-17
-18	-24	-18
13	20	17

Output image

Spatial Filtering

- The filtering we will talk about so far is referred to as ***convolution.***

- A spatial filter consists of
  - (i) a **neighborhood**, and
  - (ii) a **predefined operation** (filter/mask/kernel)

Filtering in the Spatial Domain

- ***There are two types of filtering usually***

1. Smoothing Spatial filters [low pass].
2. Sharpening Spatial Filters[high pass].

Why filters are used?

- ❑ Filters are applied on image for multiple purposes.

- ❑ Most common uses are as following:
  - ❑ Filters are used for Blurring and noise reduction
  - ❑ Filters are used for edge detection and sharpness

- ❑ Really important!
  - **Enhance images**
    - De-noise, resize, increase contrast, etc.
  - **Extract information from images**
    - Texture, edges, distinctive points, etc.
  - **Detect patterns**
    - Template matching

Smoothing Spatial filters

Use: for blurring and noise reduction.

Type of smoothing filters:

1. Standard average/Mean
2. Weighted average.

linear

3. Median filter

Order statistics (Nonlinear)

Smoothing Spatial filters

Average/Mean filter

- Mean filter is also known as Box filter and Average filter. A mean filter has the following **properties**.

➢It must be odd ordered.

➢The sum of all the elements should be **1.**

➢All the elements should be same.

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

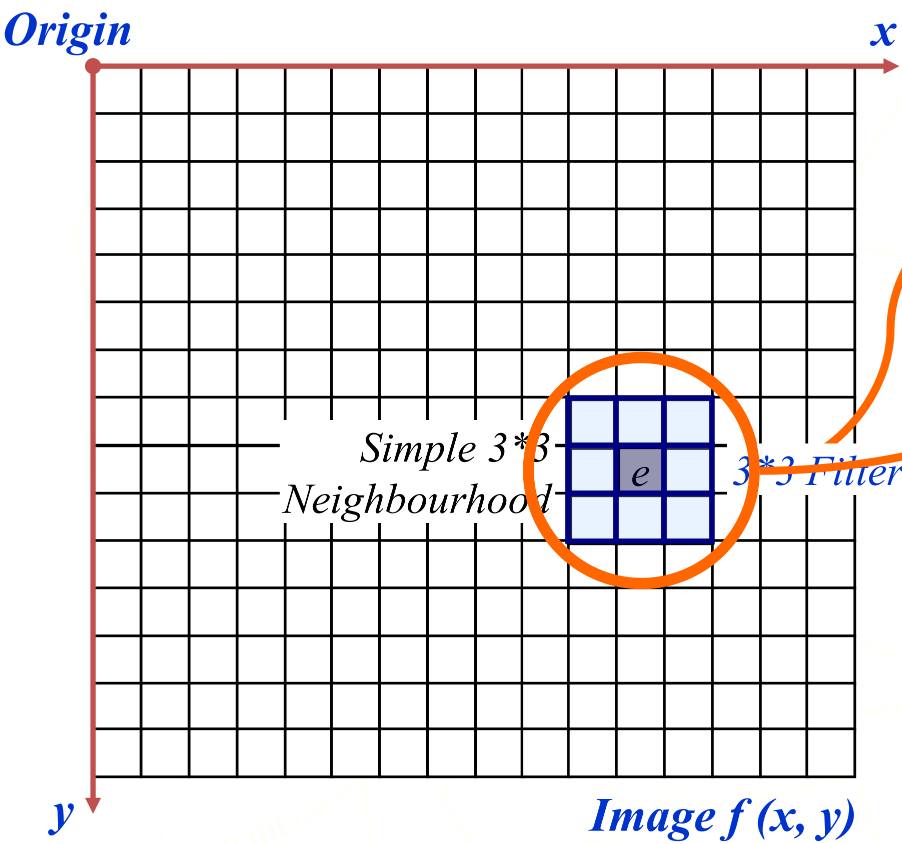
➢If we follow this rule, then for a mask of 3x3. We get the following result.

Since it is a 3x3 mask, that means it has 9 cells. The condition that all the element sum should be equal to 1 can be achieved by dividing each value by 9. As

$$1/9(1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1) = 9/9 = 1$$

Smoothing Spatial filters

Average/Mean filtering Process



104	100	108
99	106	98
95	90	85

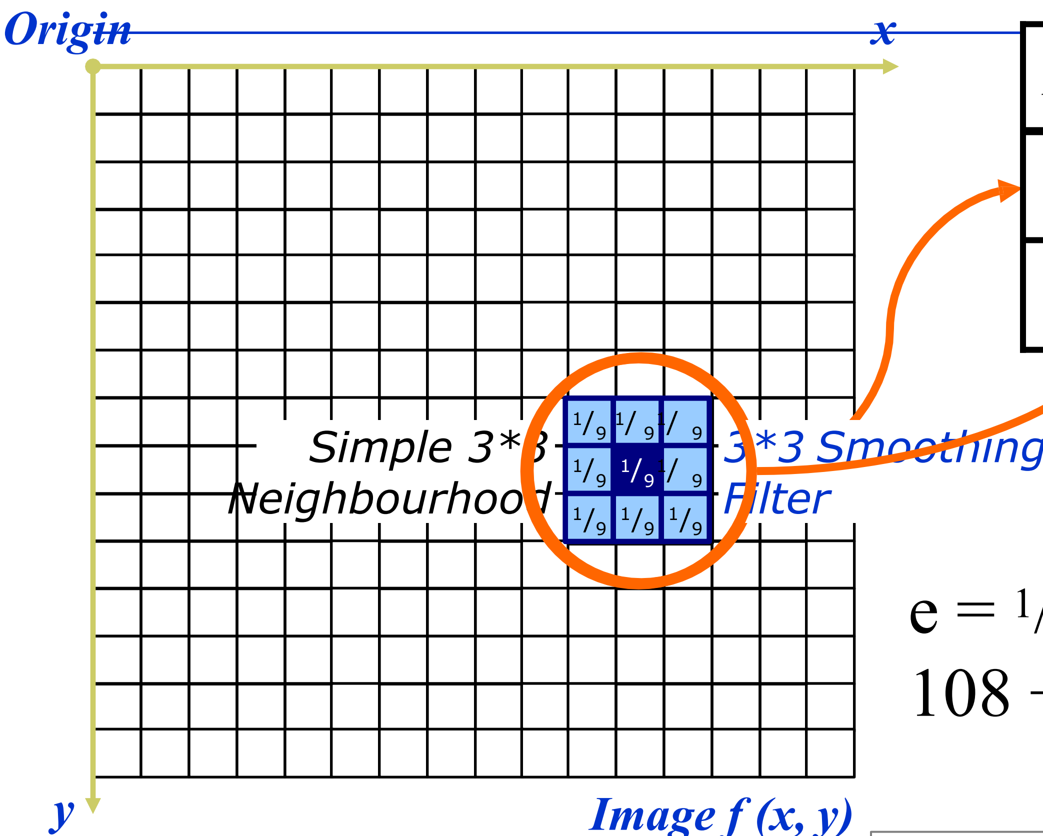
Original Image Pixels

*

$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$

Filter

Average/Mean filtering Process



104	100	108
99	106	98
95	90	85

Original Image Pixels

*

$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$

Filter

$$e = {}^1/_9 * (106 + 104 + 100 + 108 + 99 + 98 + 95 + 90 + 85) = 98.3333$$

106 will be replaced by 98 at g(x,y) image

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

Smoothing Spatial filters

Average/Mean filtering Process

- The mask is moved from point to point in an image.
- At each point (*x*,*y*), the response of the filter is calculated

110	120	90	130
91	94	98	200
90	91	99	100
82	96	85	90

$\frac{1}{9} \times$

1	1	1
1	1	1
1	1	1

Standard averaging filter:

(110 +120+90+91+94+98+90+91+99)/9 =883/9 = 98.1

Filtered image generation (Mean Filter Example)

$$h[.,.] \quad \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$f[.,.]$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$g[\cdot\,,\cdot\,]$$

- **The mask is moved from point to point in an image.**
- **At each point (*x*,*y*), the response of the filter is calculated**

Filtered image generation (Mean Filter Example)

$$h[\cdot,\cdot] \quad \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$f[\cdot,\cdot]$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$g[\cdot,\cdot]$$

	0	10							

g(x,y) = 1/9 x (0+0+0+0+0+0+0+0+90) = 10

Filtered image generation (Mean Filter Example)

$$h[.,.] \quad \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$f[.,.]$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$g[\cdot,\cdot]$$

	0	10	20						

g(x,y) = 1/9 x (0+0+0+0+0+0+0+90+90) = 20

Filtered image generation (Mean Filter Example)

$$h[.,.] \quad \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g[\cdot,\cdot]$

	0	10	20	30					

g(x,y) = 1/9 x (0+0+0+0+0+0+90+90+90) = 30

Filtered image generation (Mean Filter Example)

$$g[\cdot,\cdot]\ \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30	30				

g(x,y) = 1/9 x (0+0+0+0+0+0+90+90+90) = 30

Filtered image generation (Mean Filter Example)

$h[.,.]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g[\cdot,\cdot]$

	0	10	20	30	30				
				?					

g(x,y) = 1/9 x (0+0+0+0+90+90+90+90+90) = 50

Filtered image generation (Mean Filter Example)

$h[.,.]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g[\cdot,\cdot]$

	0	10	20	30	30				
						?			
				50					

g(x,y) = 1/9 x (90+90+90+90+90+90+90+90+90) = 90

Filtered image generation (Mean Filter Example)

$h[.,.]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

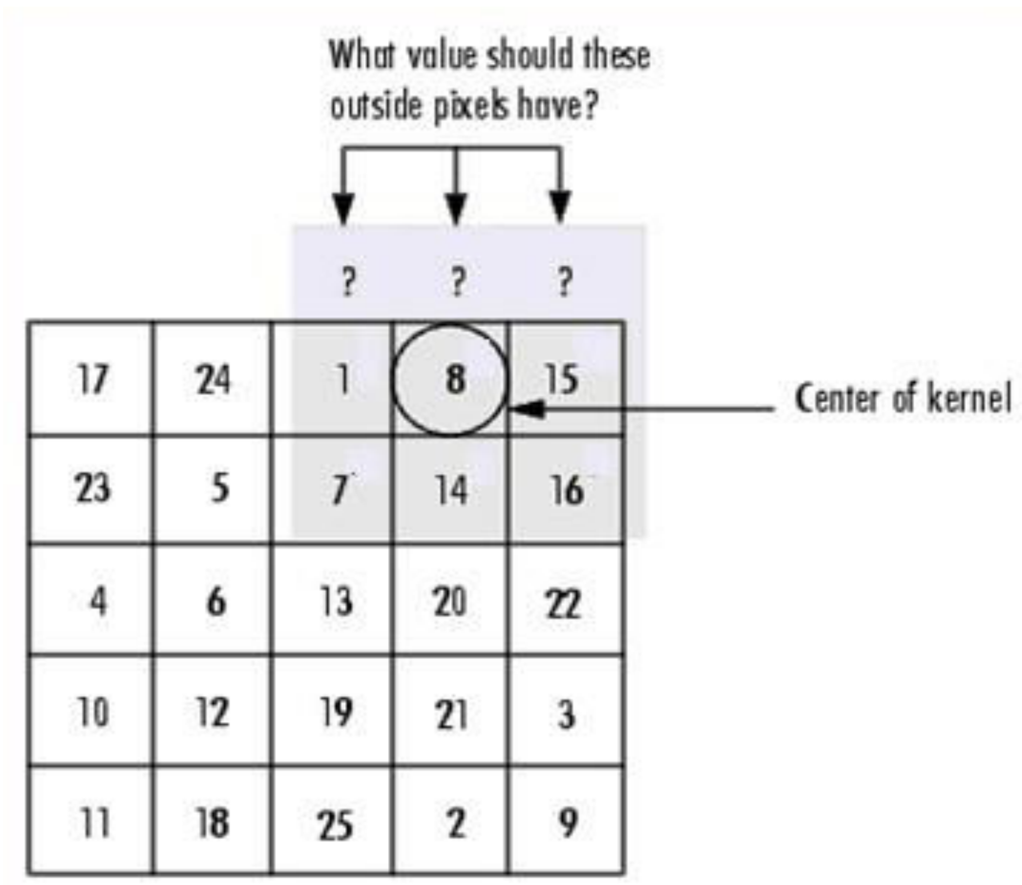
Input image f(x,y)

$g[\cdot,\cdot]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

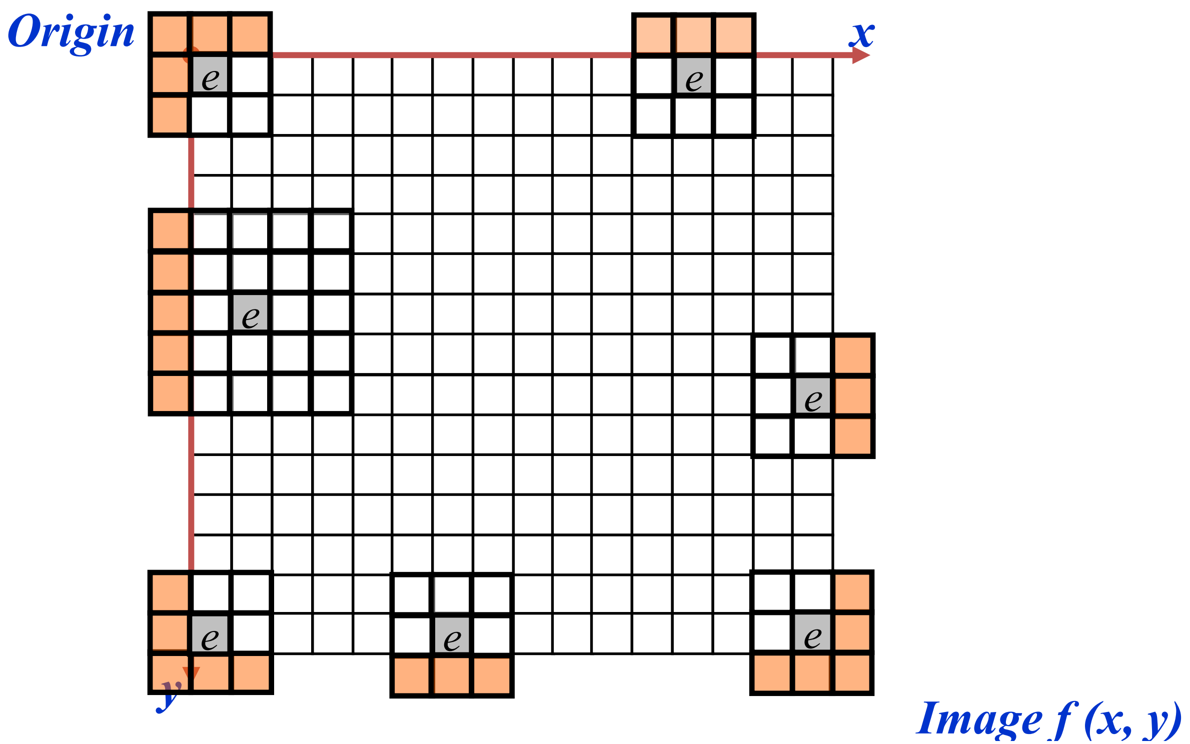
Filtered output image g(x,y)

What happens when the Values of the Kernel Fall Outside the Image??!

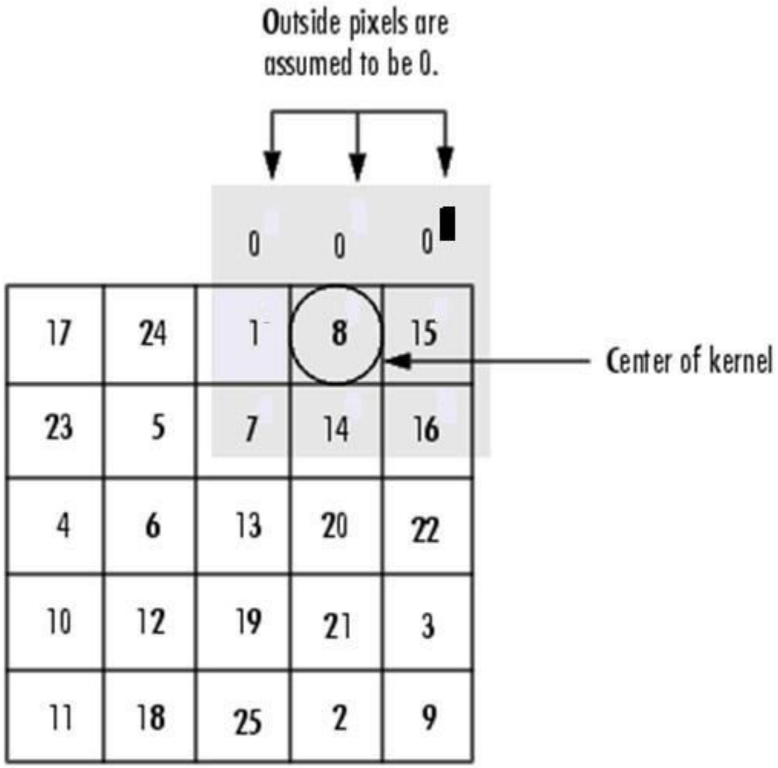


Strange Things Happen At The Edges!

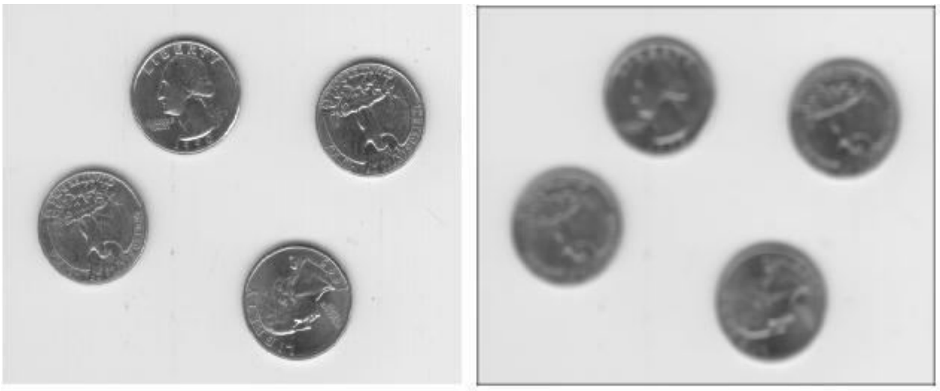
Mask operation near the image border:
Problem arises when part of the mask is located outside the image plane



First solution : Zero padding



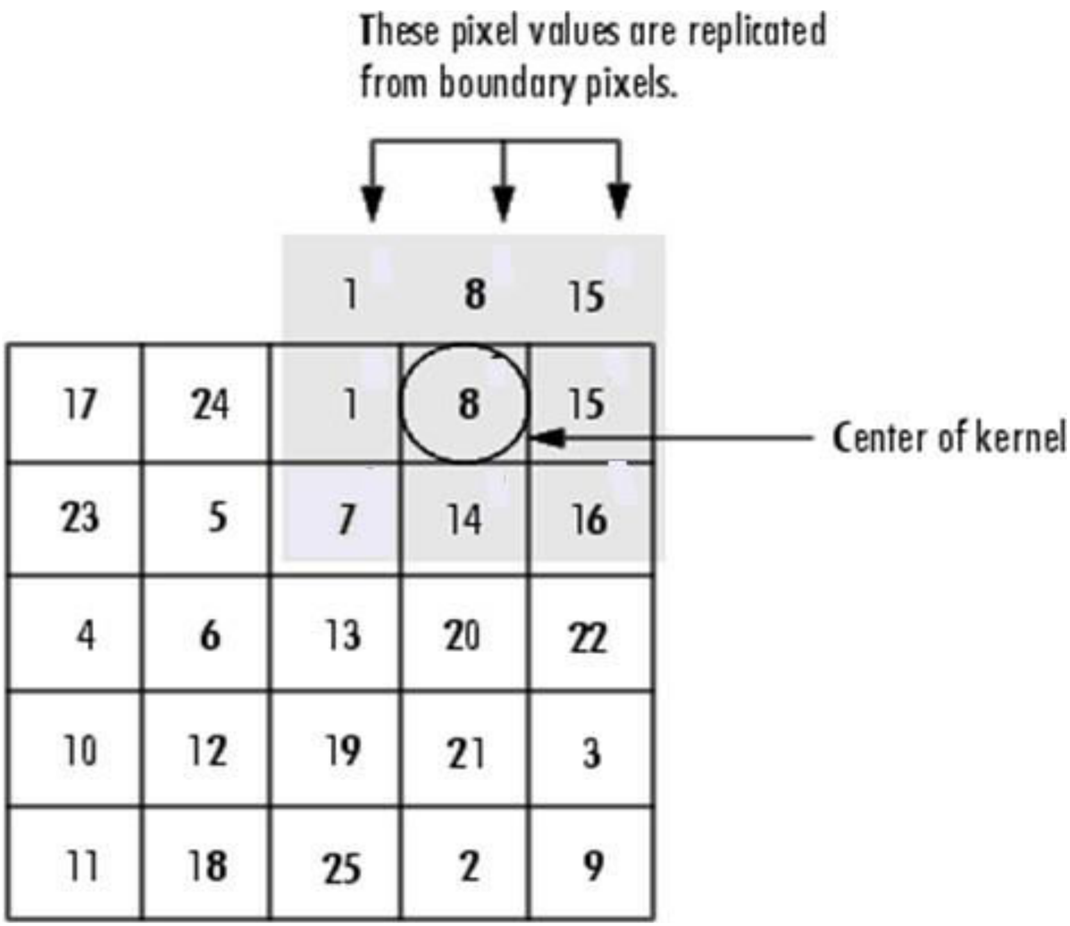
-ve: black border



Original Image

Filtered Image with Black Border

Another solution : Border padding /Pixel replication



Another solution :

Discard the problem (border) pixels.

(e.g. 512x512 input 510x510 output if mask size is 3x3)

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

	5	7	14	
	6	13	20	
	12	19	21	

What does Mean or Box filter do?

kernel for a 3x3 mean filter

$$g[\cdot\,,\cdot\,]$$

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

- Replaces each pixel with an average of its neighborhood
- Achieve smoothing effect (remove sharp features)
- Adds a “softer” look to an image

Averaging effects: blurring + reducing noise

Original image

Size: 500x500

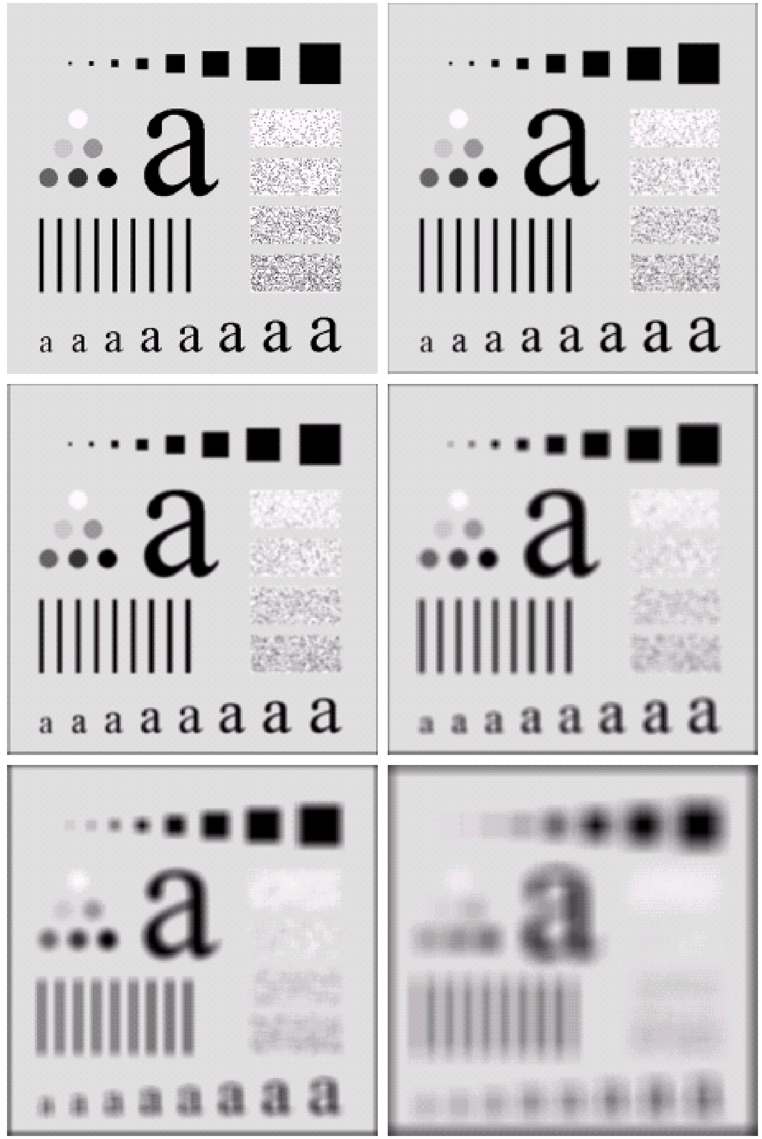
Smooth by 3x3 box filter

Smooth by 5x5 box filter

Smooth by 9x9 box filter

Smooth by 15x15 box filter

Smooth by 35x35 box filter



Notice how detail begins to disappear

Weighted Smoothing Filters

- More effective smoothing filters can be generated by allowing different pixels in the neighbourhood different weights in the averaging function.

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

weighted average filter

Weighted Average filter: Example

- The mask is moved from point to point in an image.
- At each point (x,y), the response of the filter is calculated

110	120	90	130
91	$\boxed{94}$	98	200
90	91	99	100
82	96	85	90

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

Weighted averaging filter:

(110 +2 x 120+90+2 x 91+4 x 94+2 x 98+90+2 x 91+99)/16

= (110 + 240 + 90 + 182 + 376 + 196 + 90 + 182 + 99)/16 = 97.25 = **97**

Gaussian Filter

- The Gaussian Smoothing Operator performs a weighted average of surrounding pixels based on the Gaussian distribution.
- It is used to remove Gaussian noise
- Good for reducing sharpness caused by random pixels- gives better performance than average filter.
- Discrete approximation of the Gaussian kernels 3x3, 5x5, 7x7

1/16

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

Order-Statistic (Nonlinear) Filtering

- Output is based on order of gray levels in the masked area.
- Replacing the value of the center pixel with the value determined by the ranking result.
- Some simple neighbourhood operations include:
  - **Min:** Set the pixel value to the minimum in the neighbourhood
  - **Max:** Set the pixel value to the maximum in the neighbourhood
  - **Median:** The median value of a set of numbers is the midpoint value in that set

Median filter

A **Median Filter** operates over a window by selecting the median intensity in the window.

Example of a 3x3 median filter:

110	120	90	130
91	94	98	200
90	95	99	100
82	96	85	90

becomes

110	120	90	130
91	95	98	200
90	95	99	100
82	96	85	90

Steps:

1. Sort the pixels in ascending order:

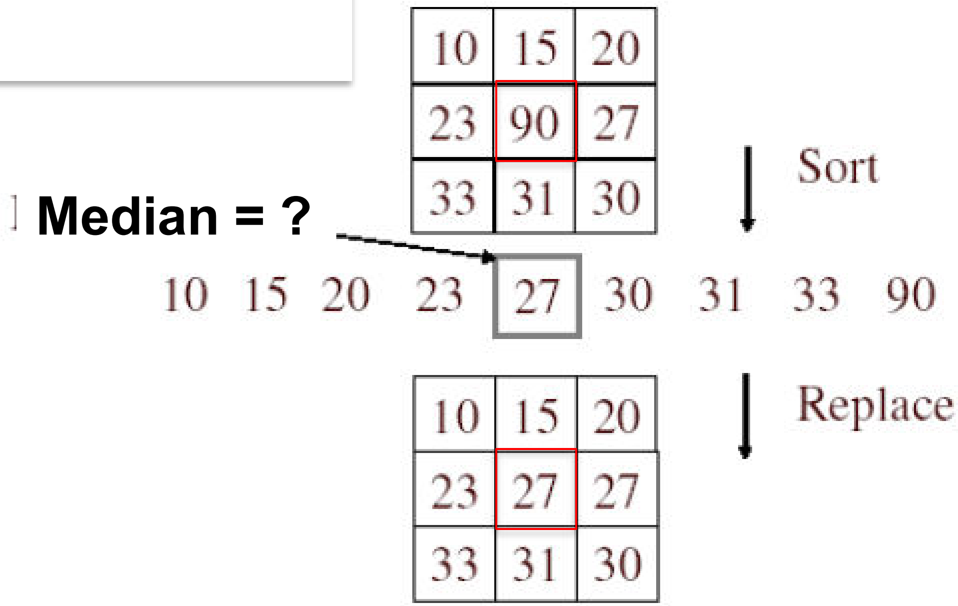
90,90, 91, 94, 95, 98, 99, 110, 120

90	**90**	**91**
94	**95**	**98**
99	**110**	**120**

2. replace the original pixel value by the median :

95

Median filter



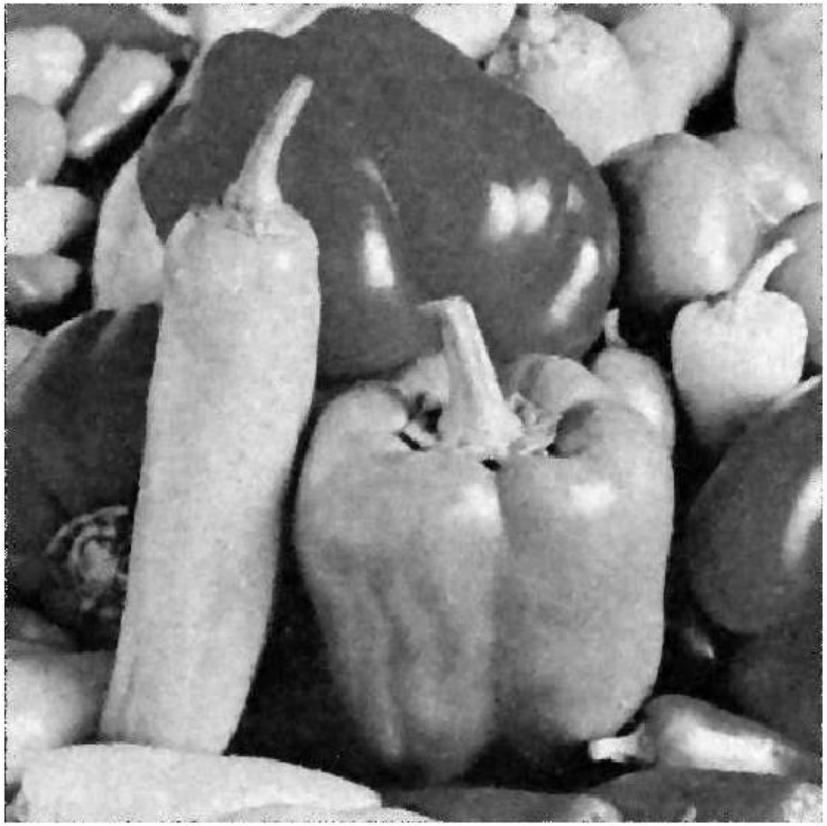
- No new pixel values introduced
- Never replace with largest or smallest value
- Removes spikes: good for impulse, salt & pepper noise
- Non-linear filter

- Particularly effective when
  - The noise pattern consists of strong impulse noise (salt-and-pepper)

Median filter effect: Blurring + Reduce Noise

Very effective for removing "salt and pepper" noise

Salt and pepper noise



Median filtered

Common types of noise

- **Salt and pepper noise**: random occurrences of black and white pixels
- **Impulse noise:** random occurrences of white pixels
- **Gaussian noise**: variations in intensity drawn from a Gaussian normal distribution

Original

Salt and pepper noise

Impulse noise

Gaussian noise

Filtering is useful for noise reduction...

Source: S. Seitz

Spatial Filtering:

with coefficient mask

- Given the 3×3 mask with coefficients: $w_1, w_2, \ldots, w_9$
- The mask cover the pixels with gray levels: $z_1, z_2, \ldots, z_9$

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

Sum of Product

$$z \longleftarrow z_1 w_1 + z_2 w_2 + z_3 w_3 + \cdots + z_9 w_9 = \sum_{i=1}^{9} z_i w_i$$

- z gives the output intensity value for the processed image (to be stored in a new array) at the location of z_5 in the input image

Predict the outputs using coefficient filtering

$$* \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = ?$$

$$* \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} = ?$$

$$* \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right) = ?$$

Practice with linear filters

Original

0	0	0
0	1	0
0	0	0

?

Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	1	0
0	0	0

Filtered
(no change)

Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	0	1
0	0	0

?

Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	0	1
0	0	0

Shifted left
By 1 pixel

Source: D. Lowe

Practice with linear filters

Original

$$* \left(\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 2 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} - \frac{1}{9} \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \right) \qquad ?$$

Source: D. Lowe

Practice with linear filters

$$* \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right)$$

Original

Sharpening filter

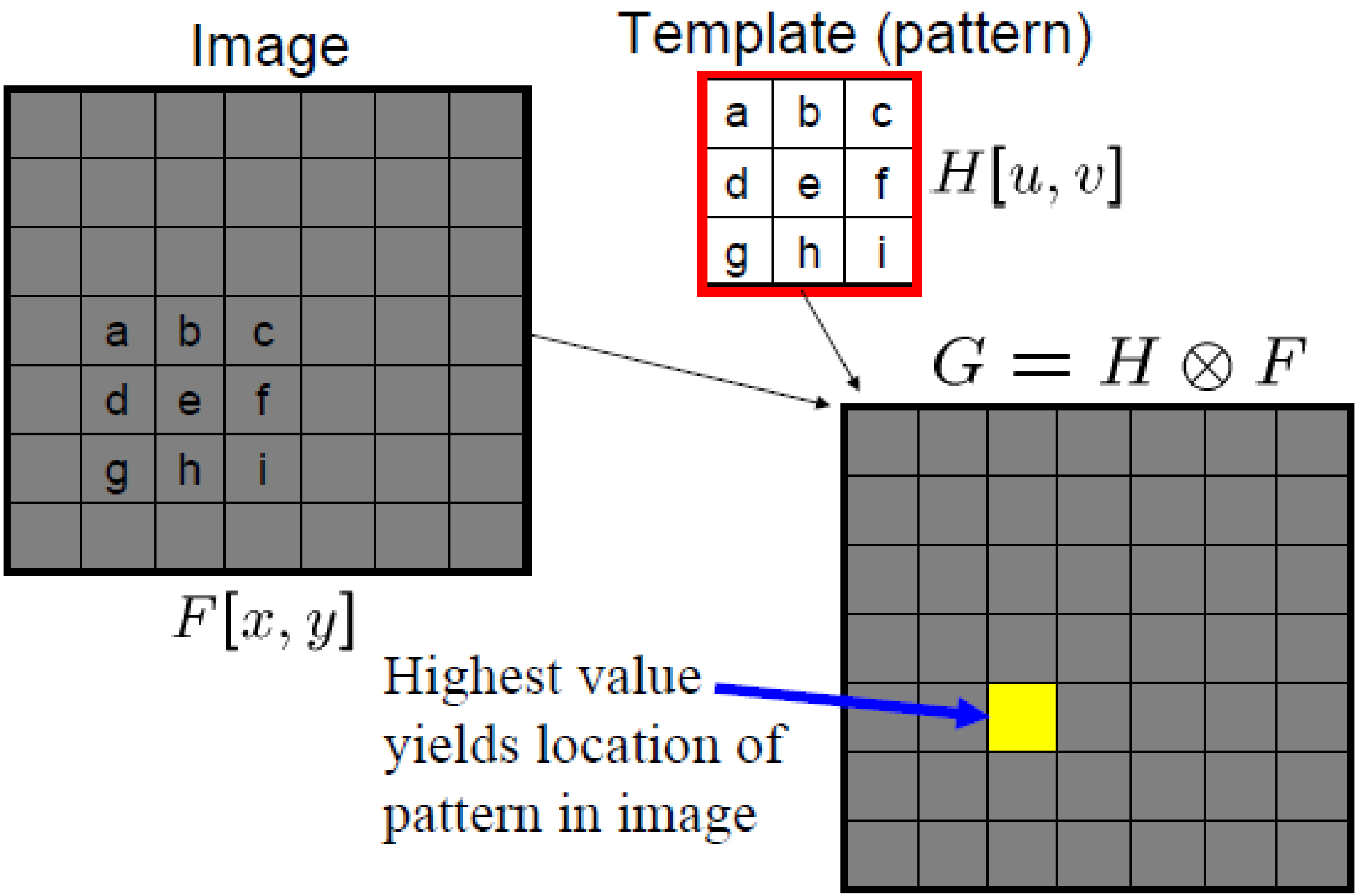
-emphasize differences with local average

Source: D. Lowe

More Applications of filtering

Cross-correlation and template matching

Cross-correlation is useful for *template matching* (locating a given pattern in an image)



Filtering an impulse signal

What is the result of filtering the impulse signal (image) *F* with the arbitrary kernel *H*?

Impulse signal

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

$F[x, y]$

Kernel

a	b	c
d	e	f
g	h	i

$H[u, v]$

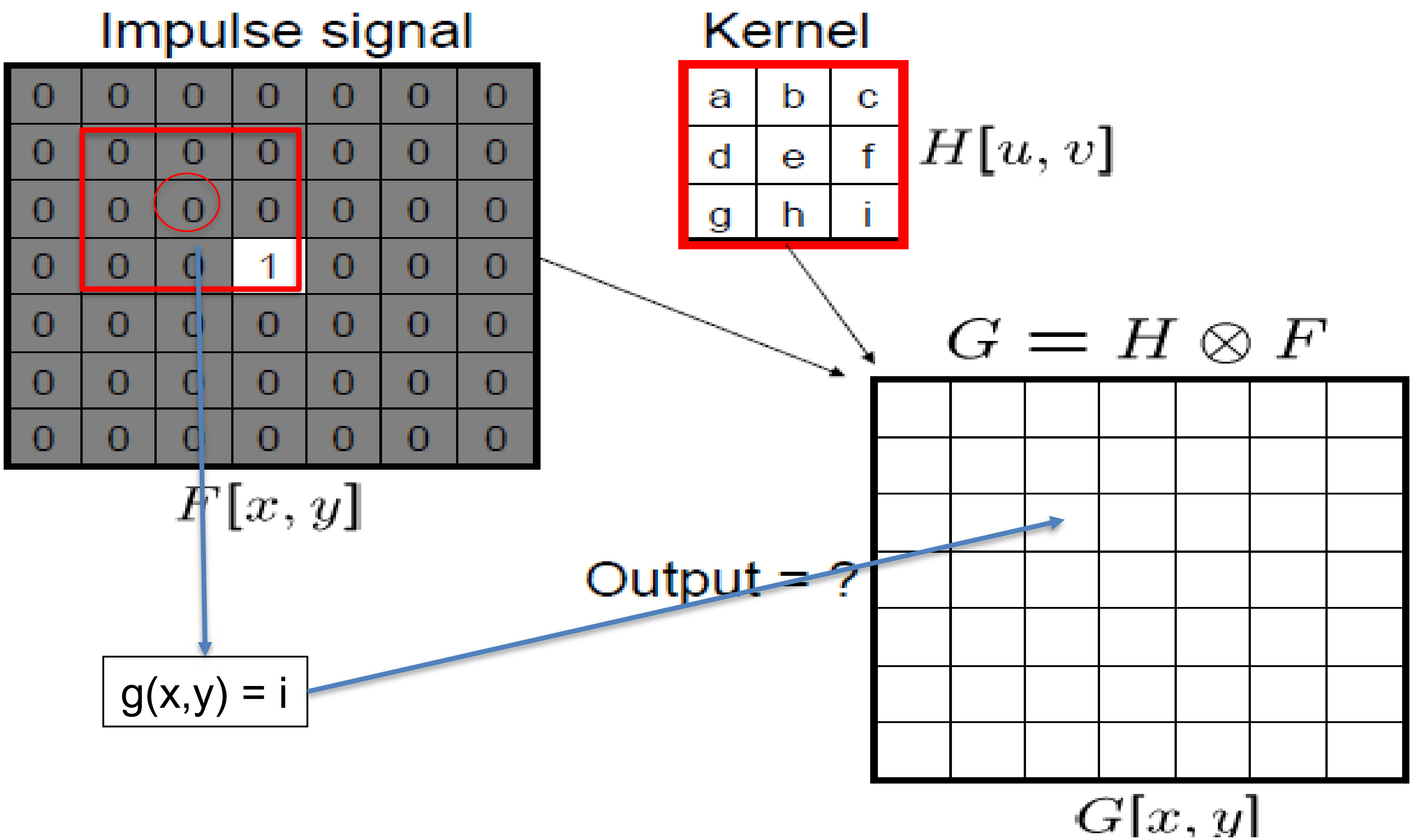
$$G = H \otimes F$$

Output = ?

$G[x, y]$

Filtering an impulse signal

What is the result of filtering the impulse signal (image) *F* with the arbitrary kernel *H*?



Filtering an impulse

Impulse signal

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

$F[x, y]$

Filter Kernel

a	b	c
d	e	f
g	h	i

$H[u, v]$

$$G = H \otimes F$$

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	i	h	g	0	0
0	0	f	e	d	0	0
0	0	c	b	a	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

$G[x, y]$

Filtering an impulse

Impulse signal

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

$F[x, y]$

Filter Kernel

a	b	c
d	e	f
g	h	i

$H[u, v]$

$$G = H \otimes F$$

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	i	h	g	0	0
0	0	f	e	d	0	0
0	0	c	b	a	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

$G[x, y]$

Output is equal to filter kernel flipped horizontally & vertically

What if we want to get an output that looks exactly like the filter kernel?

Flipping kernels

Impulse signal

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

$F[x, y]$

Filter Kernel

a	b	c
d	e	f
g	h	i

Flipped Kernel

i	h	g
f	e	d
c	b	a

$$G = H \otimes F$$

Output is equal to filter kernel!

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	a	b	c	0	0
0	0	d	e	f	0	0
0	0	g	h	i	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

$G[x, y]$