
ABV-Indian Institute of Information Technology and Management Gwalior

==========

# Image Filtering
# (ITIT-9507)

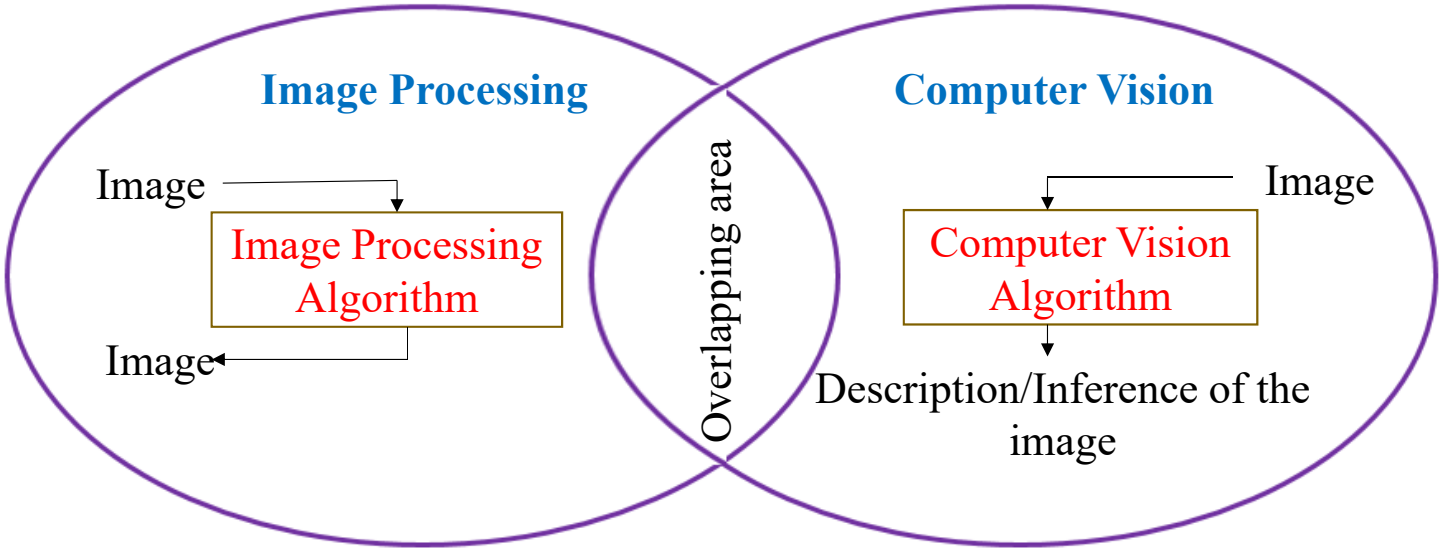
==========

Instructor – Dr. Sunil Kumar

Office – 206, F-Block (V), Tel No – 0751-2449710 (O), Email - snk@iiitm.ac.in

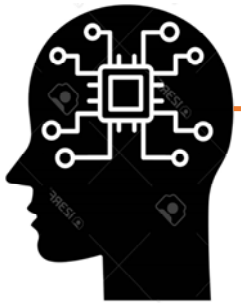
Mob - 8472842090

# Image Processing Vs. Computer Vision



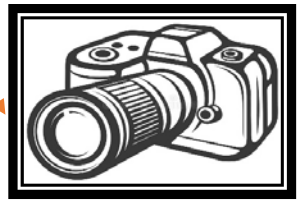
# Vision System Vs. Computer Vision

Image



Sensing Device

Interpreting Device



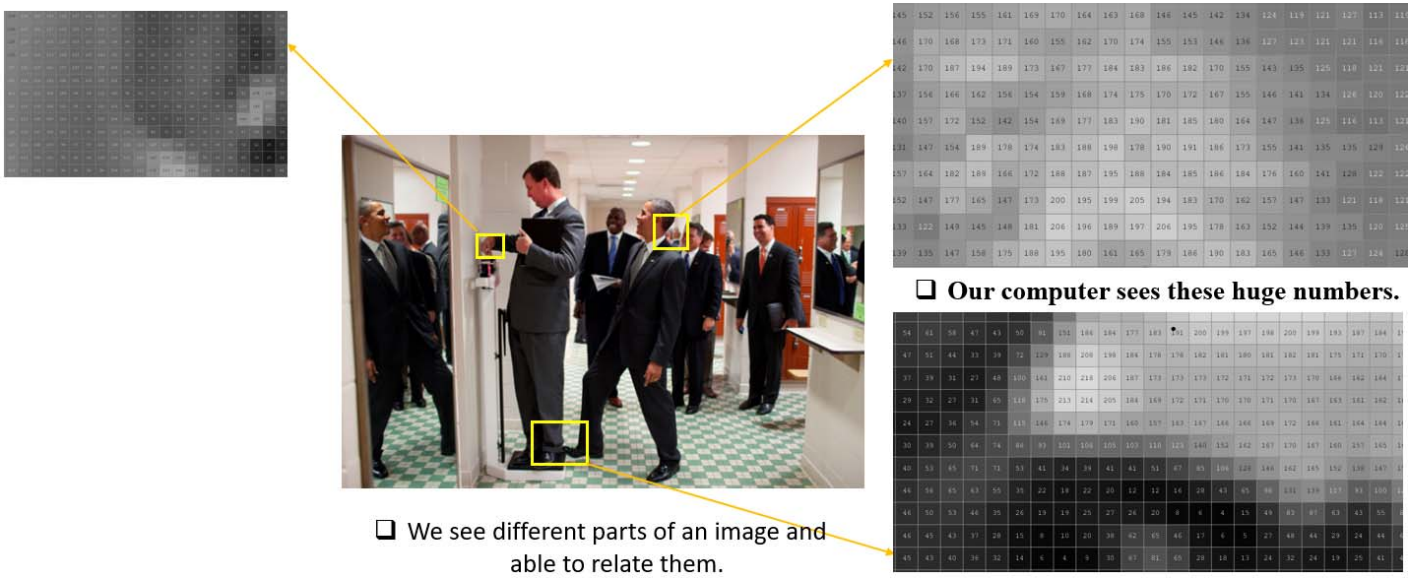
❑ **Inferences**

- ❖ It can answers many questions
  - ❖ How many persons are there in image?
  - ❖ Where is Mr. Barack obama and what is he doing?
  - ❖ Many more.
- ❖ In fact, one can write a story on this image.

- ❖ Computer vision enables computer to do all such tasks that a human can do.
- ❖ Still, it has to do a lot many ….

# Goal of Computer Vision

- To bridge the gap between "image pixels" and its "sematic meaning"




- In other words: The goal of Computer Vision is "to enable computers to extract meaning of every pixel in a image (frames of a given video) , relate them with every other pixels in order to interprets the image or a video.
- So, Computer Vision mimics the human perception.

# Digital Image Representation

Figure- 1 : ABV-IIITM Gwalior sports

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | • • • | N-1 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 5 | 20 | 3 | 0 | 88 | 47 | 255 | 156 | 20 | 90 |
| 1 | 50 | 210 | 30 | 10 | 128 | 40 | 27 | 61 | 17 | 25 |
| 2 | 7 | 23 | 43 | 137 | 18 | 51 | 43 | 29 | 89 | 86 |
| 3 | 42 | 42 | 63 | 96 | 128 | 48 | 127 | 161 | 127 | 203 |
| 4 | 50 | 210 | 30 | 10 | 128 | 40 | 27 | 61 | 17 | 26 |
| 5 | 56 | 0 | 30 | 77 | 256 | 52 | 224 | 58 | 66 | 83 |
| ⋮ | | | | | | | | | | |
| M-1 | 15 | 21 | 14 | 53 | 128 | 40 | 27 | 61 | 17 | 222 |

y-axis

x-axis

# Types of a Digital Image

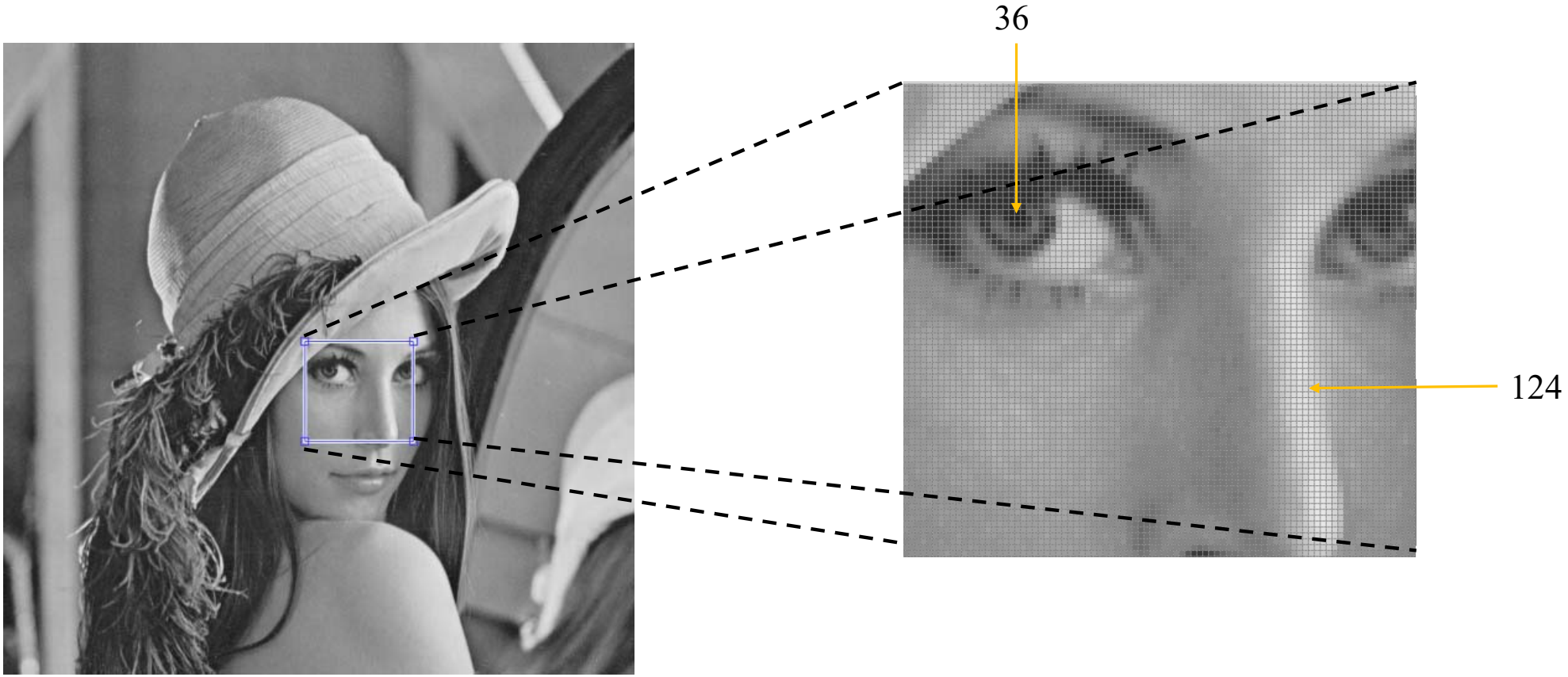
Grayscale image : 0 to 255 for 8-bits

Color image : [R-plane, G-plane, B-plane]

- Grayscale image with intensity value 0 (black) and 1 (white) Called “black and white image”.

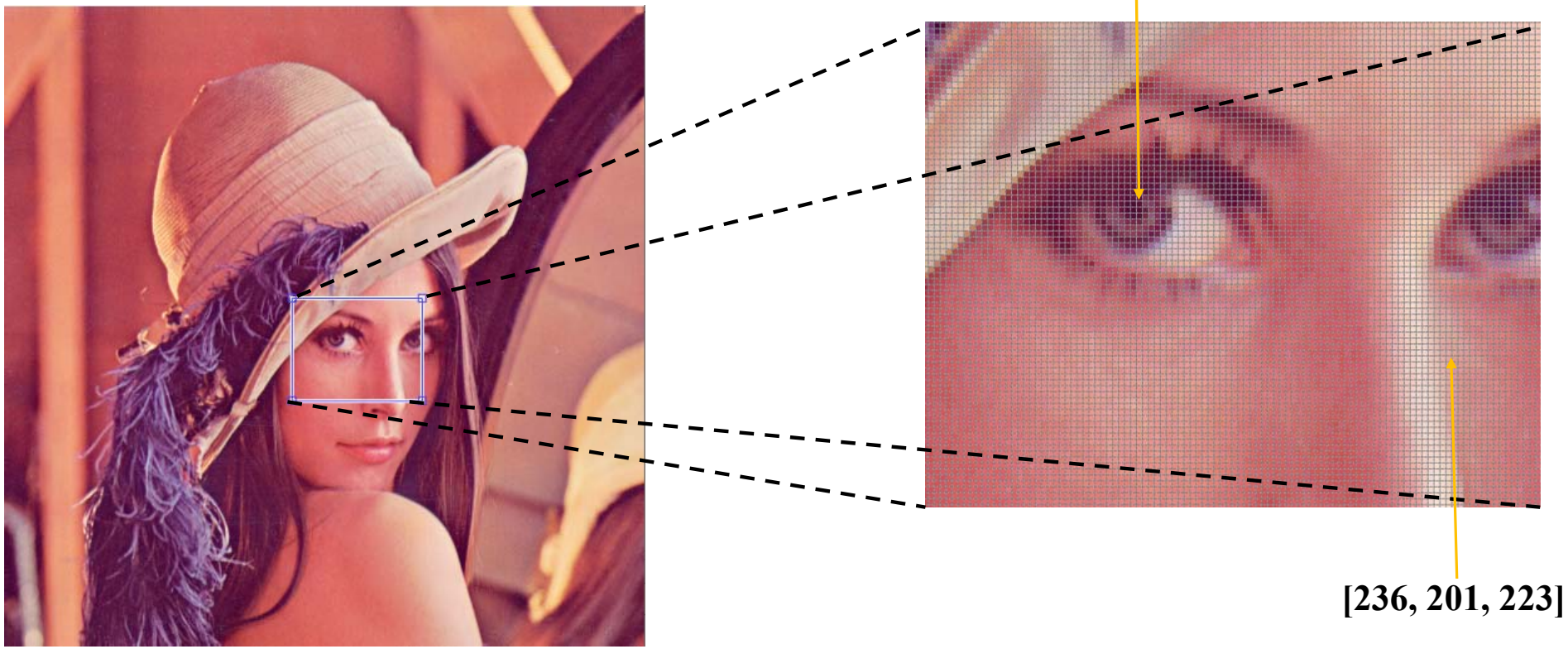
# Types of a Digital Image

- Grayscale image : 0 to 255 for 8-bits
  - Grayscale or intensity value [0, 255]



# Types of a Digital Image

- Color image : [R-plane, G-plane, B-plane]
  - Color : [R,G,B] or [L, a, b] or [H, S, V]



# Types of a Digital Image

RGB Color image : [R-plane, G-plane, B-plane]

R

G

B

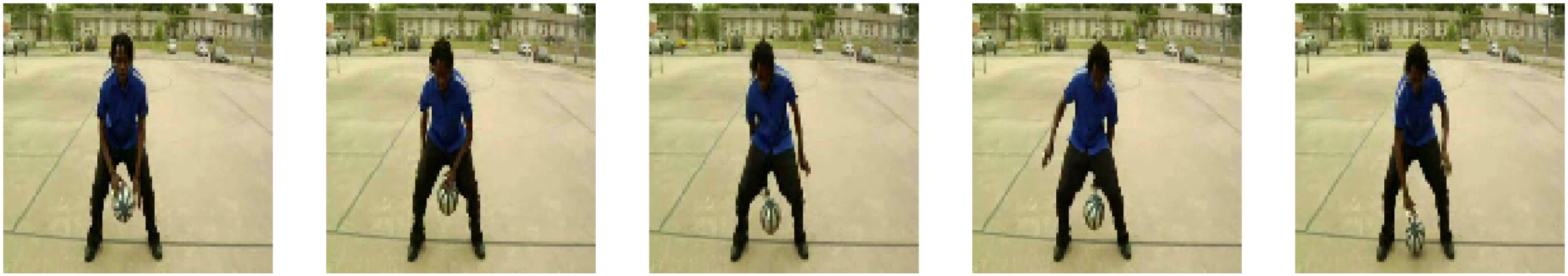
R-plane of the image

B-plane of the image

# Video

- Defined by a sequence of images (also called frames).
- Frame rate : 30 frames/second

Class: dribble



Class: kick_ball

# Image Filtering

- In general, filtering is an operation used to extract some useful (desired) information from the input signal (1D or 2D or ND).
- In image, it may be the "edges", "boundaries of the object", "denoised image", and so on.
- Image filtering is done in spatial domain and so it also known as "*Spatial filtering*".
- Spatial filtering is done by convolving a 2D-filter (also knows as mask) with the input image.
- The output image is known as filtered image.

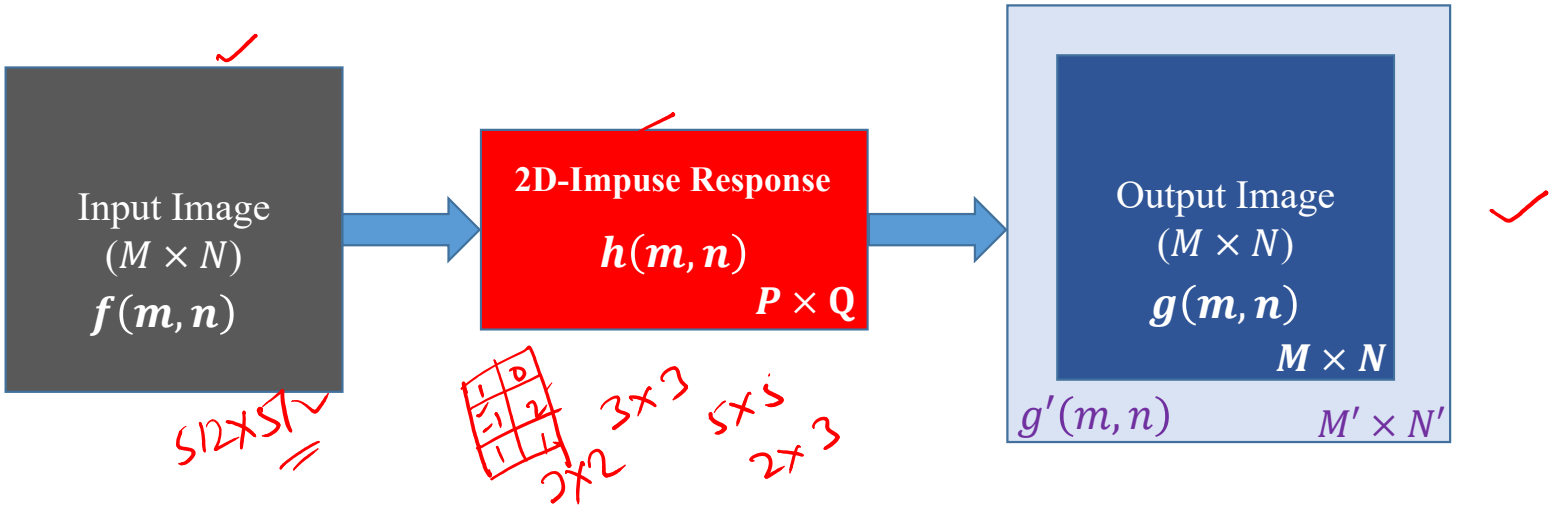



**Figure** : Spatial filtering operation on hypothetical i/p image.

# Background of Spatial Filtering

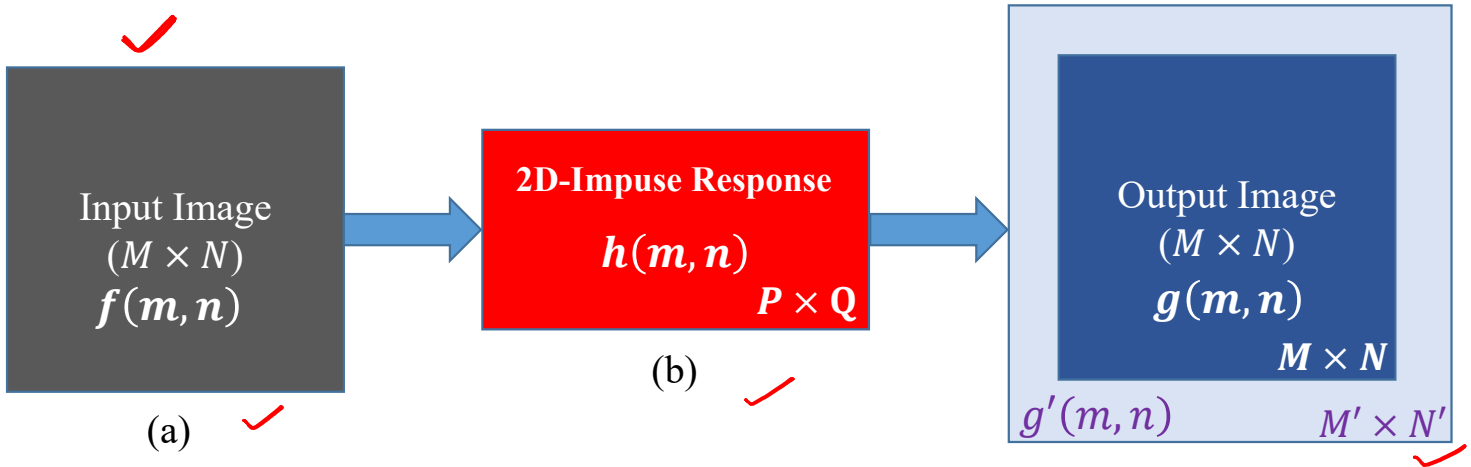



**Figure** : Spatial filtering operation on hypothetical i/p image.

- In the above representation :
  - (a) is called the input image, and it is represented in a matrix form.
  - (b) is your filter or mask or kernel, and it is also represented by a matrix of size "$3 \times 3$", or "$5 \times 5$", or "$7 \times 7$", and so on. Even the filter can be of size "$1 \times 1$".
  - (c) is the output of the system, also known as filtered output. The output image $g(m,n)$ is mathematically represented in the following ways (we will see in next slide).

# Background of Spatial Filtering

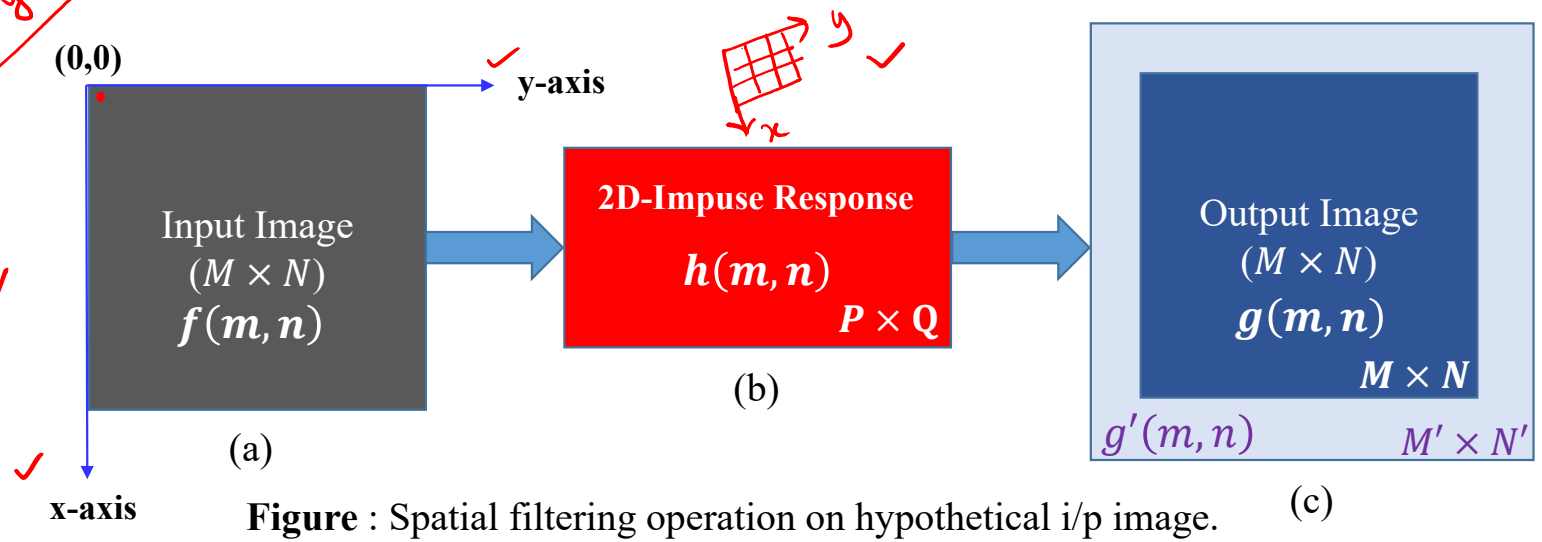



**Figure** : Spatial filtering operation on hypothetical i/p image.

**Case-1: Here both, image as well as filter (mask) are causal**

- In the above representation :
  - (c) is the output of the system, also known as filtered output. The output image $g(m,n)$ is mathematically represented as:

$$g(m,n) = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} f(k,l)\,h(m-k,n-l) = f(m,n) * h(m,n)$$

  - **Here both, image as well as filter (mask) are causal**

# Background of Spatial Filtering

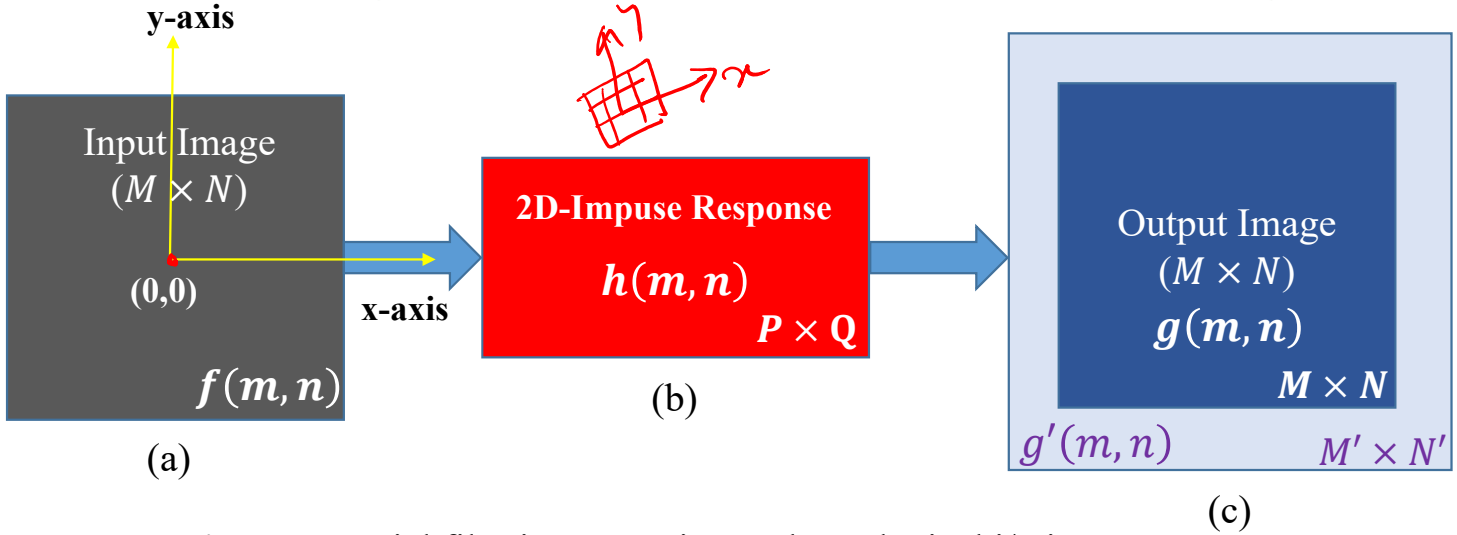



**Figure** : Spatial filtering operation on hypothetical i/p image.

- **Case-2: Here both, image as well as filter (mask) may be causal/non-causal**
- In the above representation :
  - (c) is the output of the system, also known as filtered output. The output image $g(m,n)$ is mathematically represented as:

$$g(m,n) = \sum_{k=-\infty}^{\infty} \sum_{l=-b}^{\infty} f(k,l)\,h(m-k,n-l) = f(m,n) * h(m,n)$$

# Filtering by Sliding Mask/Filter/Kernel

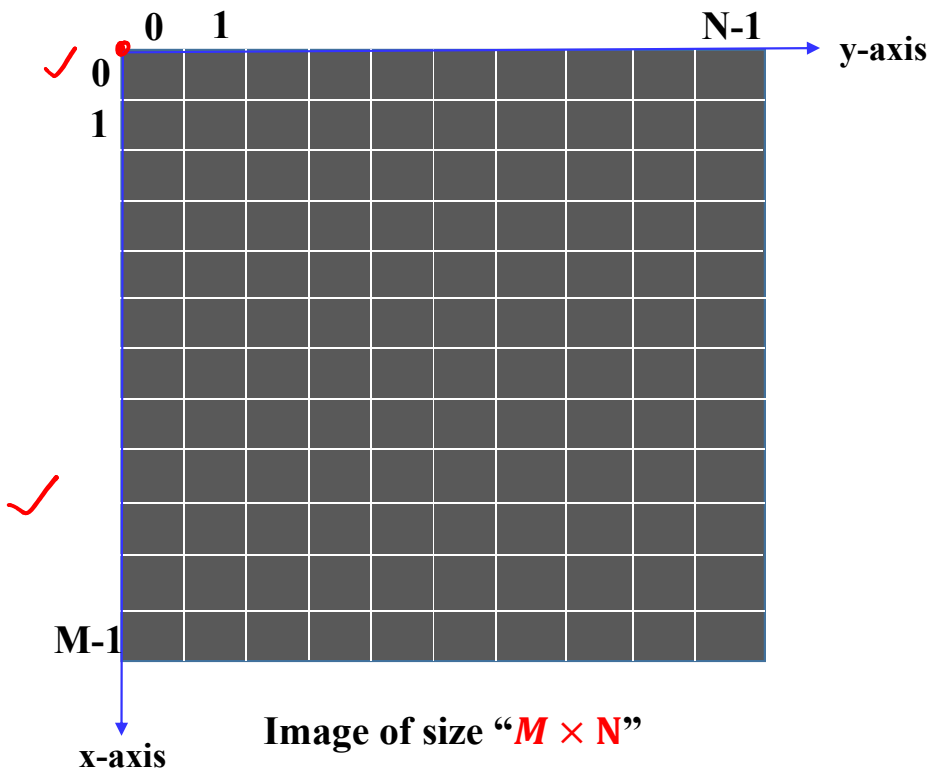



Image of size "$M \times N$"

$*$

| | | |
|---|---|---|
| $w_1$ | $w_2$ | $w_3$ |
| $w_4$ | $w_5$ | $w_6$ |
| $w_7$ | $w_8$ | $w_9$ |

y-axis (n)

x-axis (m)

Filter of size "$3 \times 3$"

$h(m, n)$

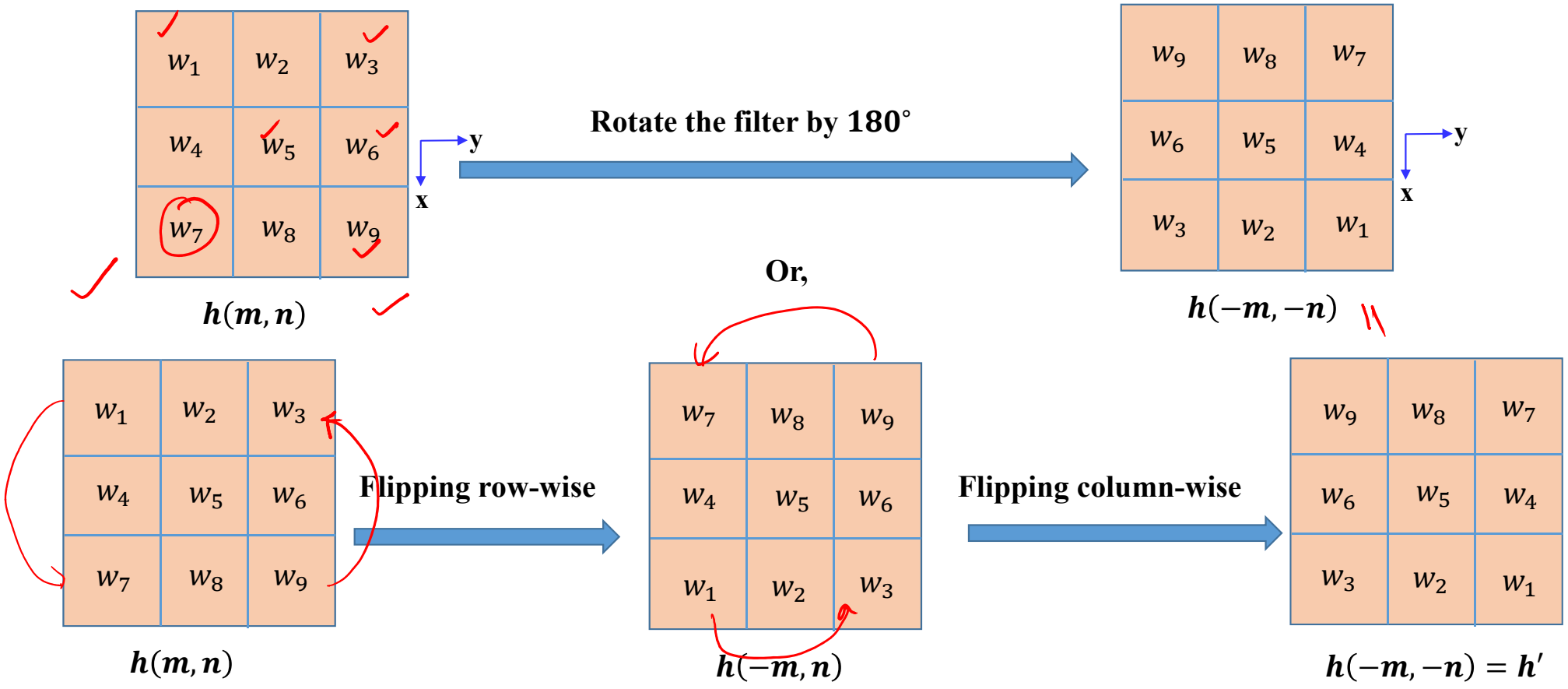
$h(-m, -n)$

Task is find $g(m,n) = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} f(k,l) h(m-k, n-l) = f(m,n) * h(m,n)$

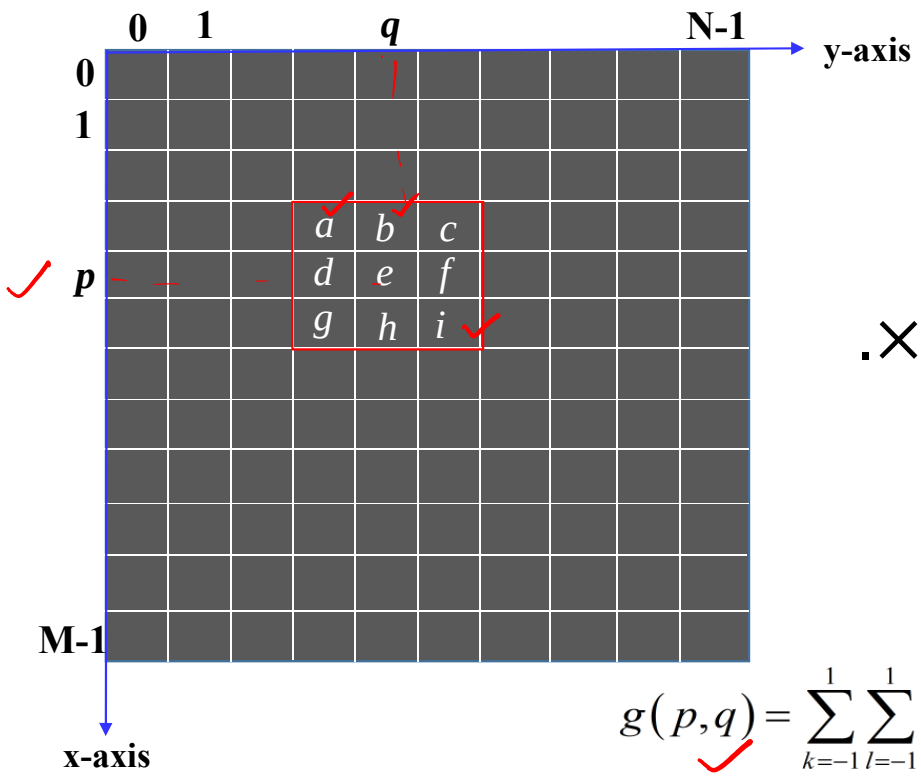
using sliding approach.

# Filtering by Sliding Mask/Filter/Kernel

**Step-1**: Rotate the filter by 180° about the origin **or** flip the filter firstly row-wise and then flip it column-wise, the result obtained in both the cases will be same.



# Filtering by Sliding Mask/Filter/Kernel




$\cdot \times / \Sigma$

| | | |
|---|---|---|
| $w_9$ | $w_8$ | $w_7$ |
| $w_6$ | $w_5$ | $w_4$ |
| $w_3$ | $w_2$ | $w_1$ |

$\boldsymbol{h(-m,-n) = h'}$

| | | |
|---|---|---|
| $w_1$ | $w_2$ | $w_3$ |
| $w_4$ | $w_5$ | $w_6$ |
| $w_7$ | $w_8$ | $w_9$ |

y-axis

x-axis

**Given filter : $\boldsymbol{h(m,n)}$**

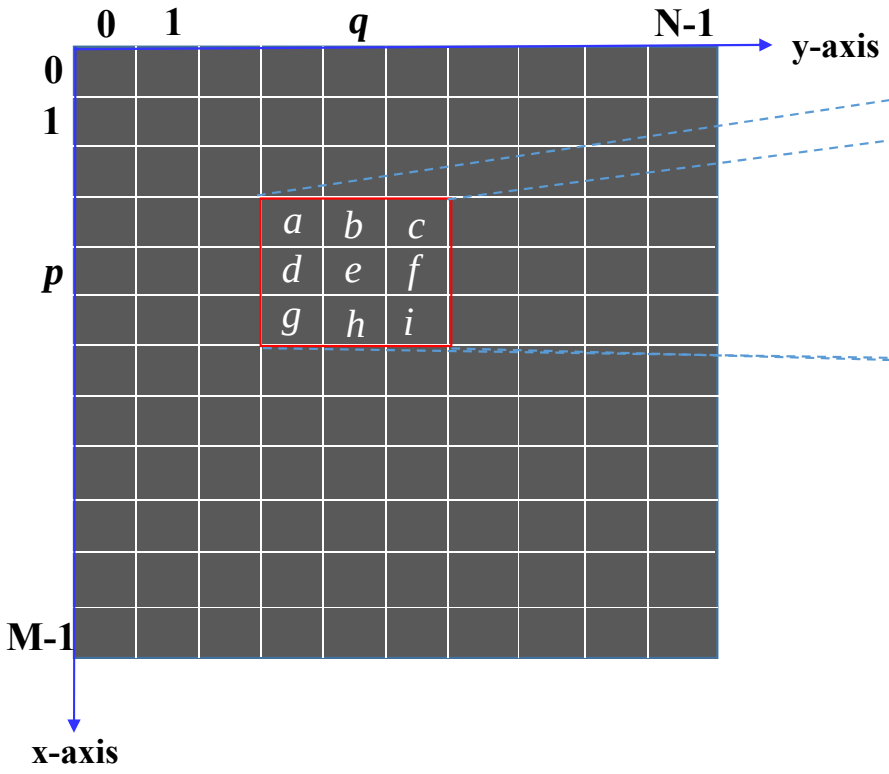
$$g(p,q) = \sum_{k=-1}^{1}\sum_{l=-1}^{1} f(k,l)h(p-k,q-l)$$

$$= f(-1,-1)h(p+1,q+1) + f(-1,0)h(p+1,q) + f(-1,1)h(p+1,q-1) + \cdots + f(1,1)h(p-1,q-1)$$

$$= f(-1,-1)w_9 + f(-1,0)w_8 + f(-1,1)w_7 + \cdots + f(1,1)w_1$$

$$= a \times w_9 + b \times w_8 + c \times w_7 + d \times w_6 + \cdots + h \times w_2 + i \times w_1$$

# Filtering by Sliding Mask/Filter/Kernel




| | | |
|---|---|---|
| $a$ | $b$ | $c$ |
| $d$ | $e$ (p, q) | $f$ |
| $g$ | $h$ | $i$ |

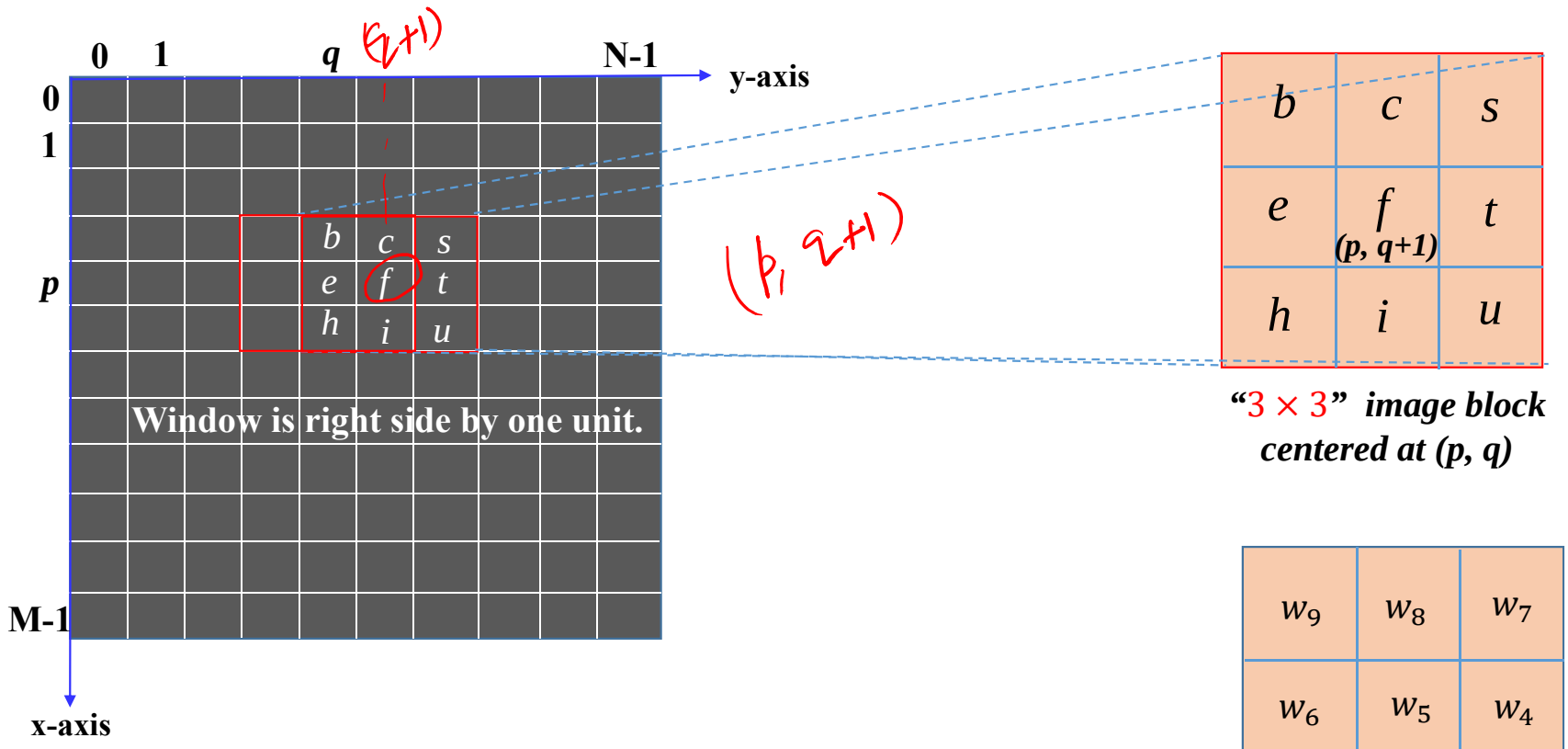
***"3 × 3" image block centered at (p, q)***

| | | |
|---|---|---|
| $w_9$ | $w_8$ | $w_7$ |
| $w_6$ | $w_5$ | $w_4$ |
| $w_3$ | $w_2$ | $w_1$ |

$$g(p,q) = a \times w_9 + b \times w_8 + c \times w_7 + d \times w_6 + \cdots + h \times w_2 + i \times w_1$$

$$\boldsymbol{h(-m,-n) = h'}$$

# Filtering by Sliding Mask/Filter/Kernel




*"3 × 3" image block centered at (p, q)*

| $w_9$ | $w_8$ | $w_7$ |
|---|---|---|
| $w_6$ | $w_5$ | $w_4$ |
| $w_3$ | $w_2$ | $w_1$ |

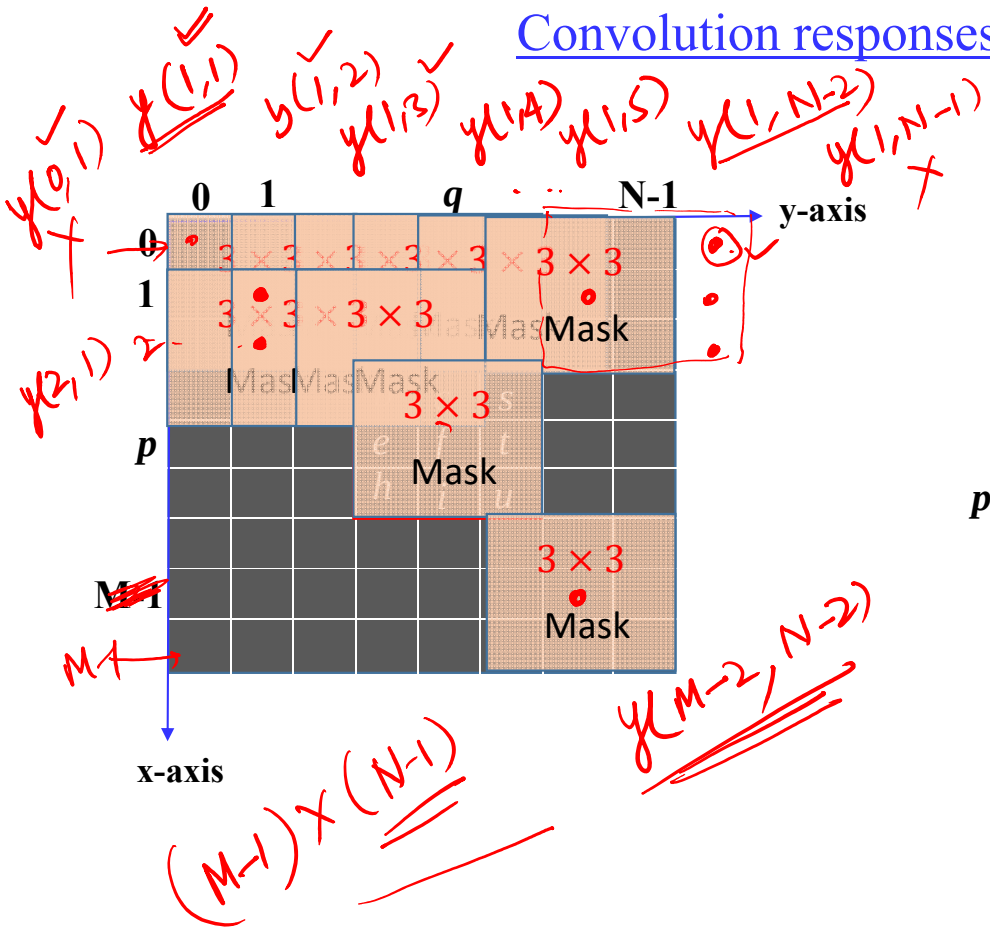
$$g(p, q+1) = b \times w_9 + c \times w_8 + s \times w_7 + e \times w_6 + \cdots + i \times w_2 + u \times w_1$$

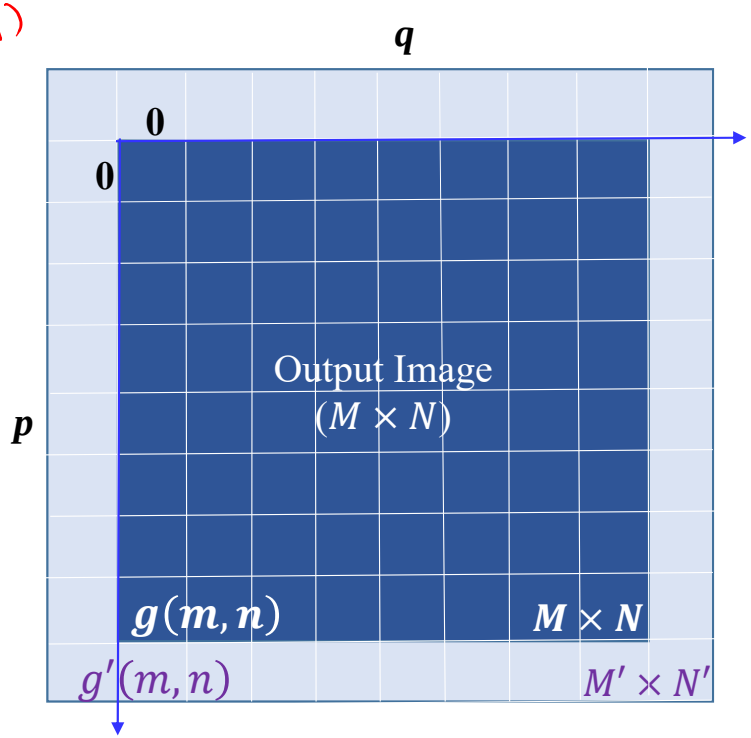
$$\boldsymbol{h(-m, -n) = h'}$$

- Similarly, we can calculate the responses for other points.

# Filtering by Sliding Mask/Filter/Kernel

Convolution responses except boundary points




| | | |
|---|---|---|
| $w_9$ | $w_8$ | $w_7$ |
| $w_6$ | $w_5$ | $w_4$ |
| $w_3$ | $w_2$ | $w_1$ |

$$h(-m, -n) = h'$$

$$\equiv$$

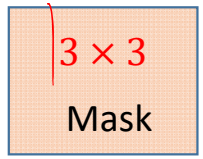
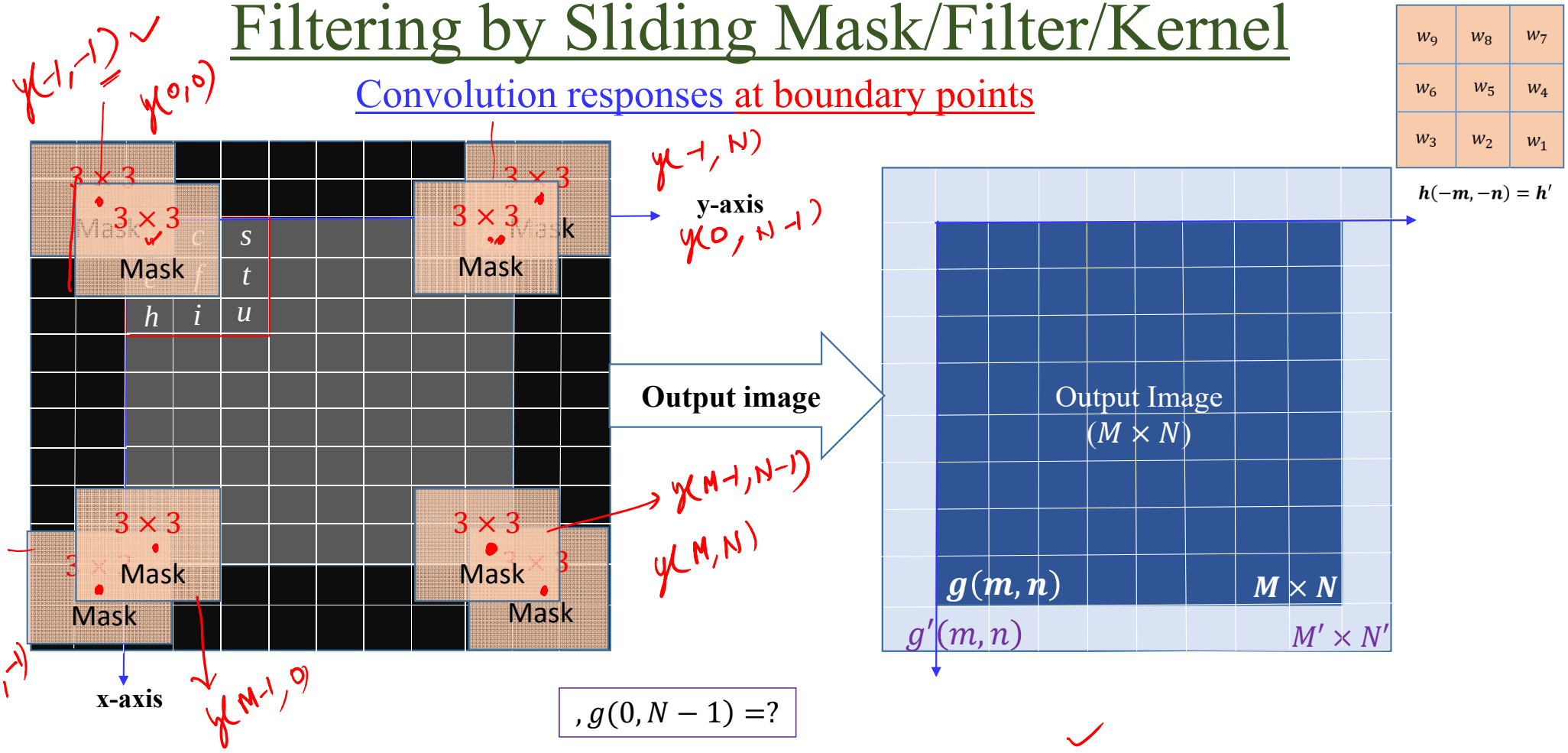



**Figure :** Sliding window approach to find convolution responses

# Filtering by Sliding Mask/Filter/Kernel

## Convolution responses at boundary points




$g(-1,-1) = b \times w_9$ ,

$g(0,0) = b \times w_5 + c \times w_4 + e \times w_2 + f \times w_1$

$, g(-1,N) = b \times w_3$

$, g(0,N-1) =?$

$, g(M-1,0) =?$

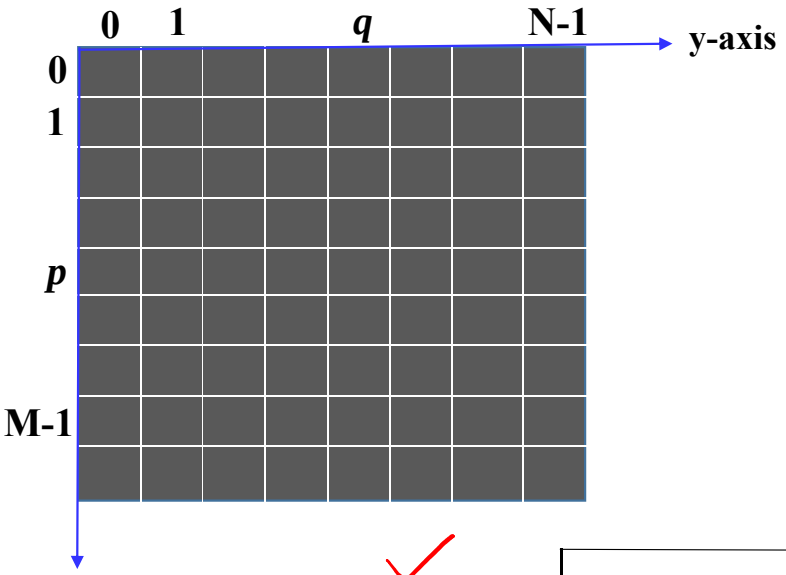
$, g(M,-1) =?$

$, g(M-1,N-1) =?$

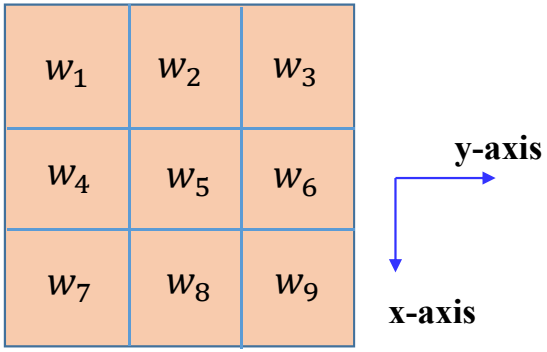
$, g(M,N) =?$

Similarly, you find outputs for all other boundary points.

# Representation of Output Filtered Image



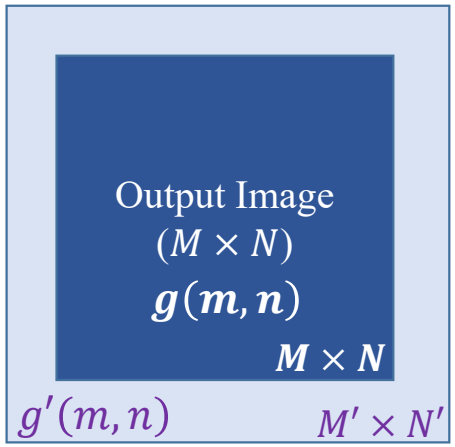

*




**Filter of size "3 × 3"**
$\boldsymbol{h(m,n)}$

| | Upper-left most pixel location | Lower-right most pixel location | Size |
|---|---|---|---|
| **Input Image** | $(a, b)$ | $(c, d)$ | $\boldsymbol{M \times N}$ |
| **Filter/Mask/Kernel** | $(e, f)$ | $(g, h)$ | $\boldsymbol{m \times n}$ |
| **Output Image** | $(a + e, b + f)$ | $(c + g, d + h)$ | $(M + m - 1,\ N + n - 1)$ |

# Representation of Output Filtered Image



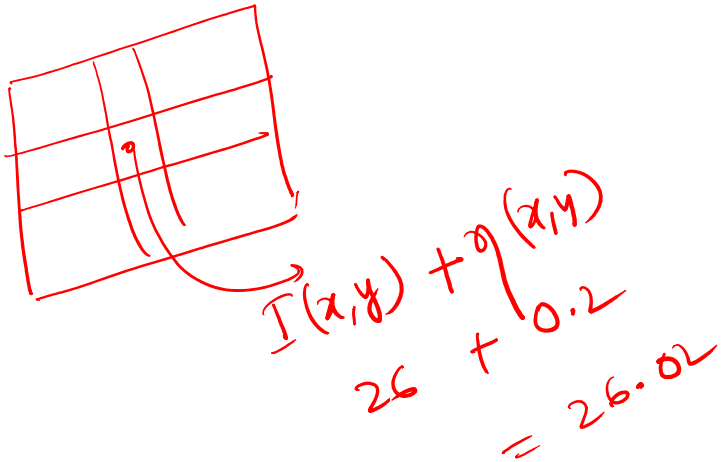

Here, $M' = \mathrm{M} + \mathrm{m} - 1$
and $N' = \mathrm{N} + \mathrm{n} - 1$

| | Upper-left most pixel location | Lower-right most pixel location | Size |
|---|---|---|---|
| **Input Image** | *(a, b)* | *(c, d)* | $\boldsymbol{M \times N}$ |
| **Filter/Mask/Kernel** | *(e, f)* | *(g, h)* | $\boldsymbol{m \times n}$ |
| **Output Image** | *(a + e, b + f)* | *(c + g, d + h)* | $(\mathrm{M} + \mathrm{m} - 1,\ \mathrm{N} + \mathrm{n} - 1)$ |

In our example : Image : *(a, b)= (0, 0); (c, d)=(M-1, N-1); size = M ×N*
Filter : *(e, f)=(-1,-1); (g, h)=(1, 1);* *size = 3 ×* **3**
Output Image : *(a + e, b + f) = (-1, -1)*
*(c + g, d + h)=(M, N); and size = (M+2, N+2)*

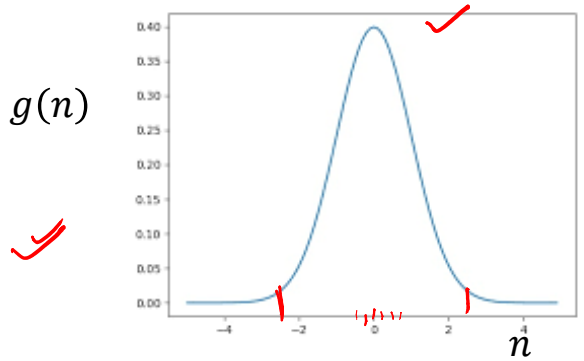
# Noise

- Noise in an image is mainly due to :
  - Image sensors
  - Light variation
  - Lens
  - Electrical fluctuation
  - …so on




- We do not much about how "noise" is varying so called "Noise".
- Described by probability density function (pdf).
- Generally, it is modelled as Gaussian distribution with zero mean and variance"$\sigma^2$"

$$\eta(x,y) \approx g(n) = e^{-\frac{n^2}{2\sigma^2}}$$



# Noisy image

$$\eta(x,y) \approx g(n) = e^{-\frac{n^2}{2\sigma^2}}$$

- Let $I_{original}(x,y)$: Actual pixel at location $(x,y)$
- $\eta(x,y)$ : noise at $(x,y)$
- $I_{observed}(x,y)$ : observed pixel at $(x,y) = I_{original}(x,y) + \eta(x,y)$ : Noise is additive
- $I_{observed}(x,y)$ : observed pixel at $(x,y) = I_{original}(x,y) \times \eta(x,y)$ : Noise is multiplicative

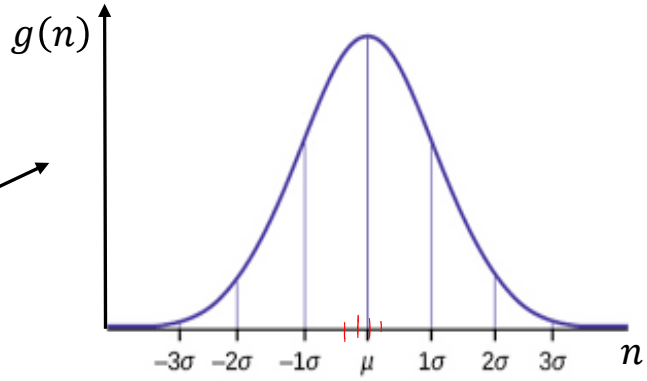
Original image

$I_{original}(x,y) + \eta(x,y)$

$I_{original}(x,y) \times \eta(x,y)$

# Noise

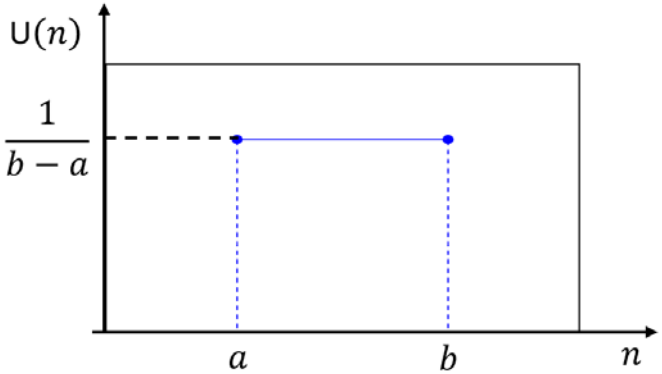
+




**Noise from Gaussian distribution**

$I_{original}(x, y) + \eta \sim g(0,0.02)$

+




**Noise from Uniform distribution**

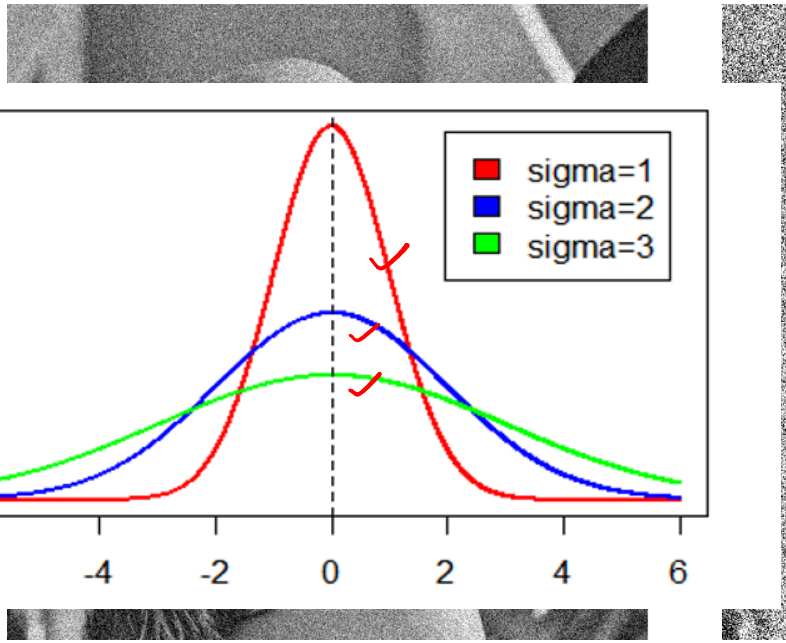
$I_{original}(x, y) + \eta \sim U(0,0.04)$

# Noisy image

- $I_{observed}(x, y)$ : observed pixel at $(x, y) = I_{original}(x, y) + \eta(x, y)$ : Noise is additive

$I_{original}(x, y) + \eta_1(x, y) \sim g(0, 0.005)$

$I_{original}(x, y) + \eta_2(x, y) \sim g(0, 0.01)$




$I_{original}(x, y) + \eta_3(x, y) \sim g(0, 0.02)$

# Image Corrupted by Different Types of noise

Original image I

a) I + Gaussian_noise

b) I + Poisson_noise

c) I + Salt and paper noise

d) I + Speckle noise : I + n*I

# Image Filtering

- Image filtering is used to manipulate image pixels.
  - to get rid of noise
  - Highlight edges
  - Texture features
  - …so on
- Analysis problem :
- **Case-1:** Linear filtering
  - Given an image *I(m,n)* and the filter/mask/kernel *h(m,n)*, find the output image *g(m,n)* using convolution sum expression.
- **Case-2:** Non-linear filtering
  - Compute some mathematical operations over the neighbourhood of pixel (x,y) in order to modify it.

$$g(m,n) = \sum_{k,l} I(k,l)h(m-k,n-l) = I(m,n) * h(m,n)$$

$f$ ( | $a$ | $b$ | $c$ | | $d$ | $e$ | $f$ | | $g$ | $h$ | $i$ | ) = | | | | | | $p$ | | | | |

- Example;
- $f(.)$ may be median over 3x3 block.
- $f(m,n) = \sum_{k,l} \alpha_k I^2(m-k,n-l)$

# Linear Filtering : Convolution

- **Case-1(a):** Linear filtering
  - Given an image *I(m,n)* and the filter/mask/kernel *h(m,n)*, find the output image *g(m,n)* using convolution sum expression.

$$I * h = \sum_k \sum_l I(k,l)\, h(-k,-l)$$

- Example : Let $I =$ $\begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \\ p_7 & p_8 & p_9 \end{bmatrix}$ and kernel $h =$ $\begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$

$\begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$ $h(k,l)$ —X-flip→ $\begin{bmatrix} h_3 & h_2 & h_1 \\ h_6 & h_5 & h_4 \\ h_9 & h_8 & h_7 \end{bmatrix}$ $h(-k,l)$ —Y-flip→ $\begin{bmatrix} h_9 & h_8 & h_7 \\ h_6 & h_5 & h_4 \\ h_3 & h_2 & h_1 \end{bmatrix}$ $h(-k,-l)$ $.\times/\Sigma$ $\begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \\ p_7 & p_8 & p_9 \end{bmatrix}$ $I$ $=$ $\begin{bmatrix} & & \\ & \mathbf{q} & \\ & & \end{bmatrix}$ Output = $\boldsymbol{I * h}$

$$\boldsymbol{I * h = h_9p_1 + h_8p_2 + h_7p_3 + h_6p_4 + h_5p_5 + h_4p_6 + h_3p_7 + h_2p_8 + h_1p_9 = q}$$

# Linear Filtering : Correlation

- **Case-1(b):** Linear filtering
  - Given an image *I(m,n)* and the filter/mask/kernel *h(m,n)*, the correlation output is defined as.

$$I \otimes h = \sum_k \sum_l I(k,l)\,h(k,l)$$

- Example : Let $I =$

| | | |
|---|---|---|
| $p_1$ | $p_2$ | $p_3$ |
| $p_4$ | $p_5$ | $p_6$ |
| $p_7$ | $p_8$ | $p_9$ |

and kernel $h =$

| | | |
|---|---|---|
| $h_1$ | $h_2$ | $h_3$ |
| $h_4$ | $h_5$ | $h_6$ |
| $h_7$ | $h_8$ | $h_9$ |

$I \otimes h =$

| | | |
|---|---|---|
| $h_1$ | $h_2$ | $h_3$ |
| $h_4$ | $h_5$ | $h_6$ |
| $h_7$ | $h_8$ | $h_9$ |

$h(k,l)$

$.\times/\Sigma$

| | | |
|---|---|---|
| $p_1$ | $p_2$ | $p_3$ |
| $p_4$ | $p_5$ | $p_6$ |
| $p_7$ | $p_8$ | $p_9$ |

$I$

$=$

| | | |
|---|---|---|
| | | |
| | **s** | |
| | | |

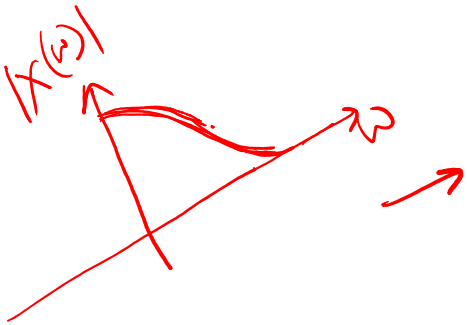
Output = $I \otimes h$

Note: $I * h\,(Conv) = I \otimes h\,(Corr)$
If $h(-k,-l) = h(k,l)$
i.e., when kernel is symmetric

$$I \otimes h = h_1p_1 + h_2p_2 + h_3p_3 + h_4p_4 + h_5p_5 + h_6p_6 + h_7p_7 + h_8p_8 + h_9p_9 = s$$

# Linear Filtering : Mean Filtering

- **Example -1 (a) :** Blurring: -
  - Example




| | | |
|---|---|---|
| 10 | 1 | 10 |
| 5 | 5 | 7 |
| 15 | 3 | 2 |

$I$

$* \frac{1}{9}$

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

$h$

$=$

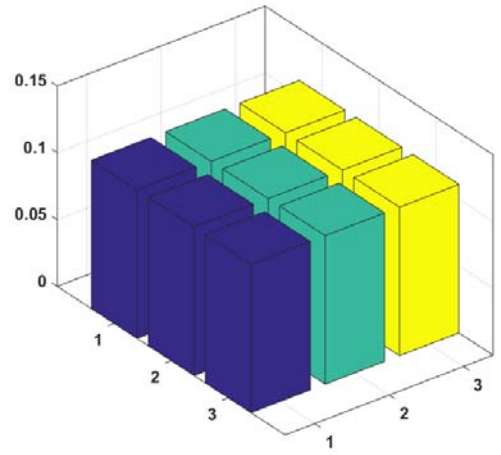
| | | |
|---|---|---|
| | | |
| | **q** | |
| | | |

Output = $\boldsymbol{I} * \boldsymbol{h}$

$q = \frac{1}{9}(10 + 1 + 10 + 5 + 5 + 7 + 15 + 3 + 2) = 6.4 \approx 6$

||

$*$

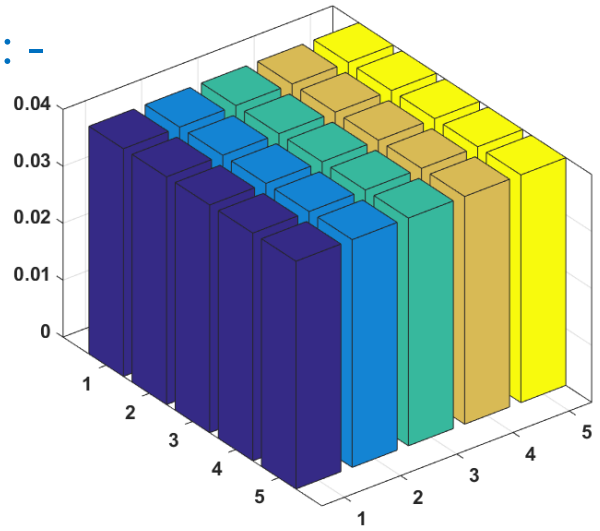


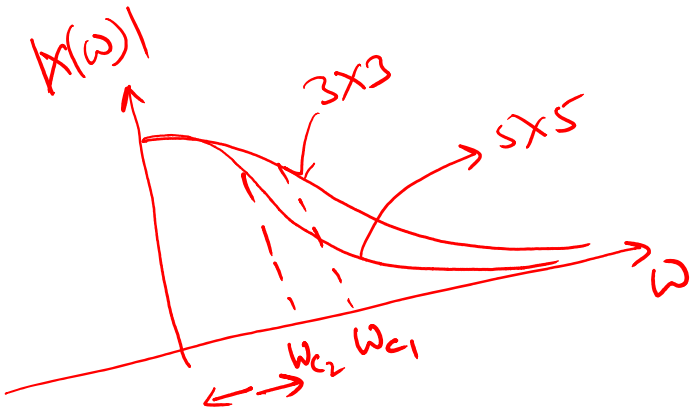

3x3 filter

$=$

# Linear Filtering : Mean Filtering

- **Example -1 (b):** Blurring: -



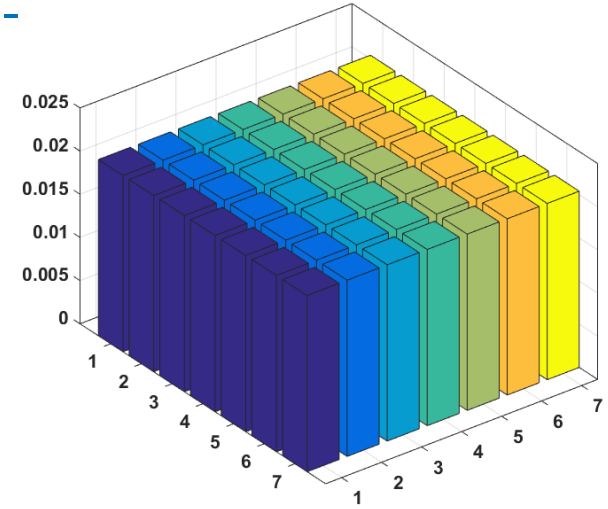




$$* \quad \frac{1}{25} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} =$$

5x5 filter

# Linear Filtering : Mean Filtering

- **Example -1 (c):** Blurring: -

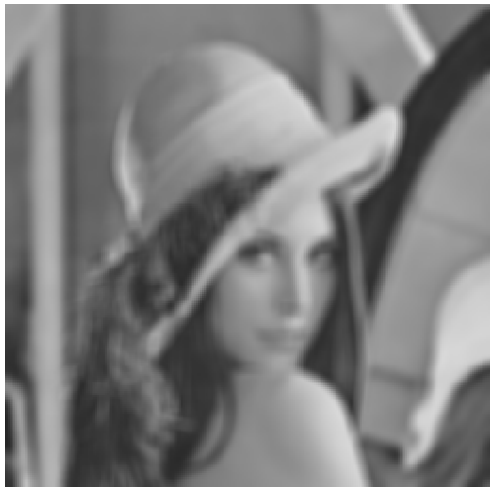



$$* \frac{1}{49}$$

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

7x7 filter ✓

$=$



# Linear Filtering : Mean Filtering

❑ **Example -2 (c):** Denoising -

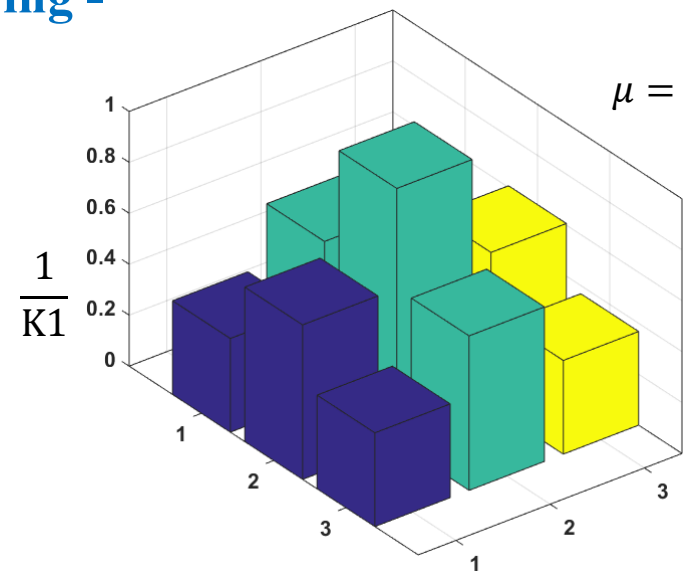

Output : 3 x 3 mean filter

Output : 5 x 5 mean filter

Output : 7 x 7 mean filter

# Linear Filtering : Gaussian Filtering

❑ **Example -3 (a) : Blurring -**




$$* \quad \frac{1}{K1} \begin{bmatrix} 0.3679 & 0.6065 & 0.3679 \\ 0.6065 & 1.0000 & 0.6065 \\ 0.3679 & 0.6065 & 0.3679 \end{bmatrix} =$$

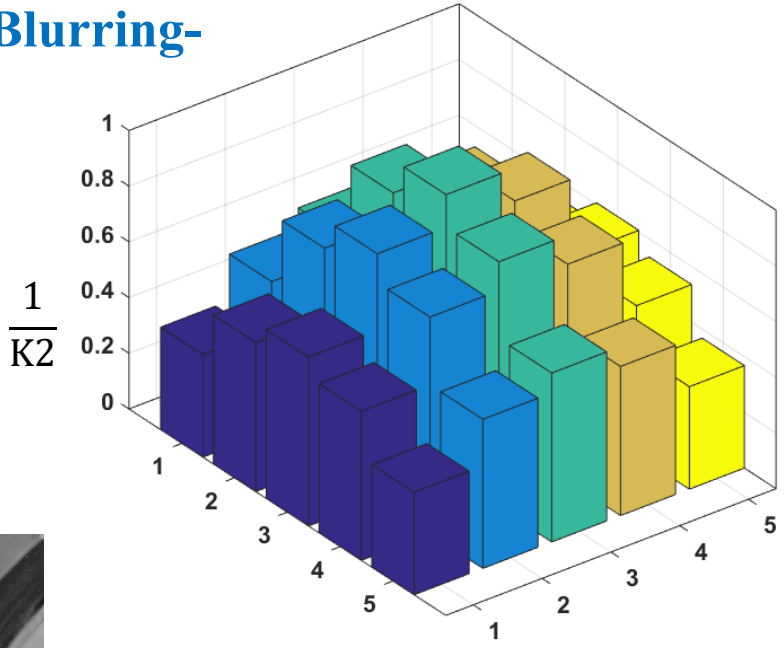
3x3 Gaussian filter

2D-Gaussian Distribution

$$G(x, y, \mu, \Sigma) = \frac{1}{2\pi\Sigma} e^{-(x-\mu)^T \Sigma^{-1} (x-\mu)}$$

# Linear Filtering : Gaussian Filtering

❑ **Example -3 (b) : Blurring-**




| | | | | |
|---|---|---|---|---|
| 0.3679 | 0.5353 | 0.6065 | 0.5353 | 0.3679 |
| 0.5353 | 0.7788 | 0.8825 | 0.7788 | 0.5353 |
| 0.6065 | 0.8825 | 1.0000 | 0.8825 | 0.6065 |
| 0.5353 | 0.7788 | 0.8825 | 0.7788 | 0.5353 |
| 0.3679 | 0.5353 | 0.6065 | 0.5353 | 0.3679 |

$\mu = 0, \sigma^2 = 2$

$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \Sigma^2 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$

$\frac{1}{K2}$ * $\frac{1}{K2}$ **5 x 5 Gaussian Mask** =

5 x 5 Gaussian Mask

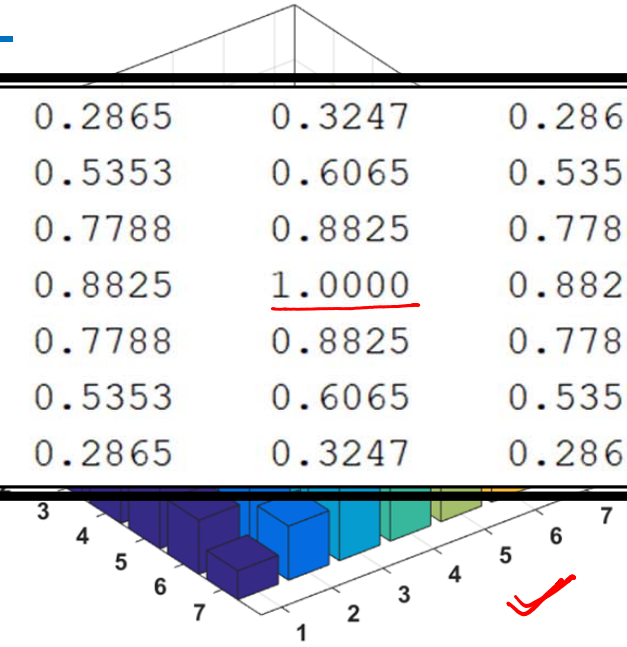
# Linear Filtering : Gaussian Filtering

❑ **Example -3 (c) : Blurring-**

$\mu = 0, \sigma^2 = 2$

$\frac{1}{K3}$

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.1054 | 0.1969 | 0.2865 | 0.3247 | 0.2865 | 0.1969 | 0.1054 |
| 0.1969 | 0.3679 | 0.5353 | 0.6065 | 0.5353 | 0.3679 | 0.1969 |
| 0.2865 | 0.5353 | 0.7788 | 0.8825 | 0.7788 | 0.5353 | 0.2865 |
| 0.3247 | 0.6065 | 0.8825 | 1.0000 | 0.8825 | 0.6065 | 0.3247 |
| 0.2865 | 0.5353 | 0.7788 | 0.8825 | 0.7788 | 0.5353 | 0.2865 |
| 0.1969 | 0.3679 | 0.5353 | 0.6065 | 0.5353 | 0.3679 | 0.1969 |
| 0.1054 | 0.1969 | 0.2865 | 0.3247 | 0.2865 | 0.1969 | 0.1054 |

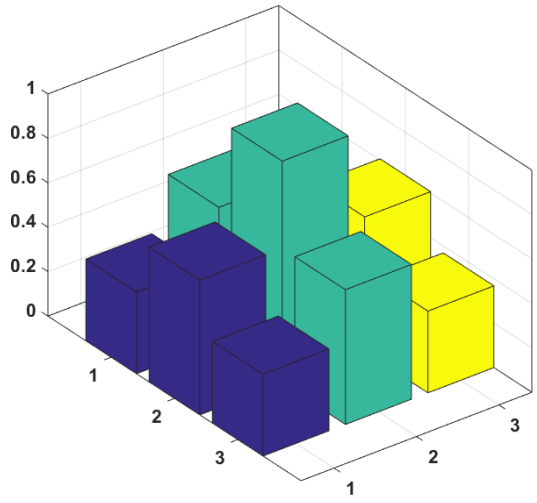



* $\frac{1}{K3}$ **7 x 7 Gaussian Mask** =

7 x 7 Gaussian Mask

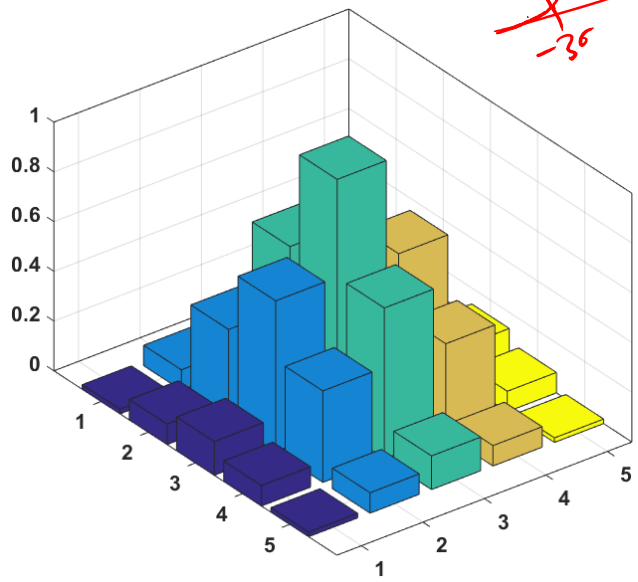
# Linear Filtering : Gaussian Filters

❑ **Example -3(d) :** Size of mask w.r.t $\sigma^2 = 1$:



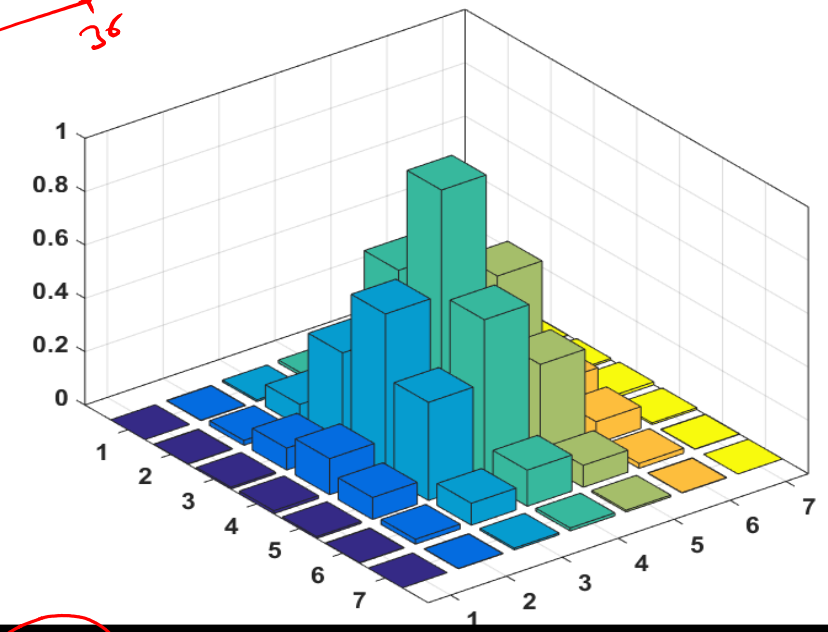

| | | |
|---|---|---|
| 0.3679 | 0.6065 | 0.3679 |
| 0.6065 | 1.0000 | 0.6065 |
| 0.3679 | 0.6065 | 0.3679 |

3 × 3 Gaussian Mask




| | | | | |
|---|---|---|---|---|
| 0.0183 | 0.0821 | 0.1353 | 0.0821 | 0.0183 |
| 0.0821 | 0.3679 | 0.6065 | 0.3679 | 0.0821 |
| 0.1353 | 0.6065 | 1.0000 | 0.6065 | 0.1353 |
| 0.0821 | 0.3679 | 0.6065 | 0.3679 | 0.0821 |
| 0.0183 | 0.0821 | 0.1353 | 0.0821 | 0.0183 |

5 × 5 Gaussian Mask



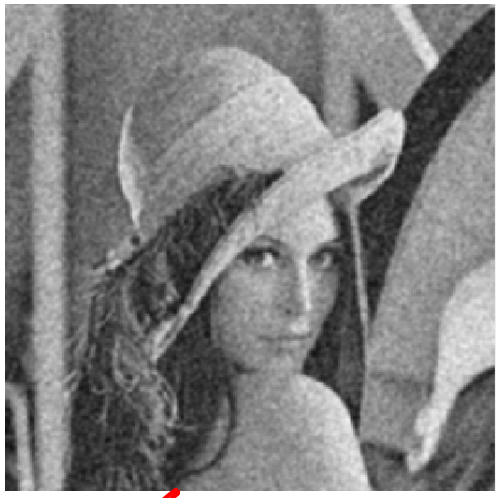

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.0001 | 0.0015 | 0.0067 | 0.0111 | 0.0067 | 0.0015 | 0.0001 |
| 0.0015 | 0.0183 | 0.0821 | 0.1353 | 0.0821 | 0.0183 | 0.0015 |
| 0.0067 | 0.0821 | 0.3679 | 0.6065 | 0.3679 | 0.0821 | 0.0067 |
| 0.0111 | 0.1353 | 0.6065 | 1.0000 | 0.6065 | 0.1353 | 0.0111 |
| 0.0067 | 0.0821 | 0.3679 | 0.6065 | 0.3679 | 0.0821 | 0.0067 |
| 0.0015 | 0.0183 | 0.0821 | 0.1353 | 0.0821 | 0.0183 | 0.0015 |
| 0.0001 | 0.0015 | 0.0067 | 0.0111 | 0.0067 | 0.0015 | 0.0001 |

7 × 7 Gaussian Mask

# Linear Filtering : Gaussian Filtering

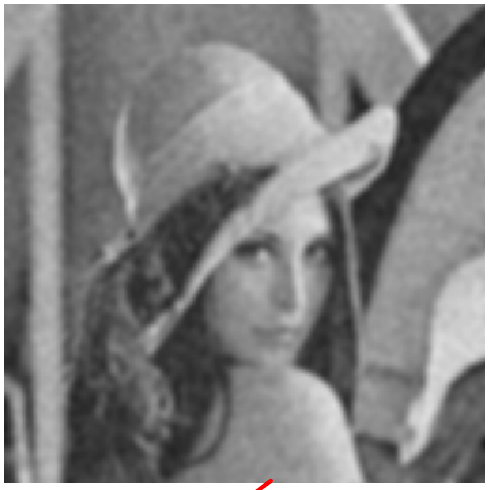
❑ **Example -3 (e):** Denoising -

Input image

Output : 3 x 3 Gaussian filter (var =1)

Output : 5 x 5 Gaussian filter (var =2)

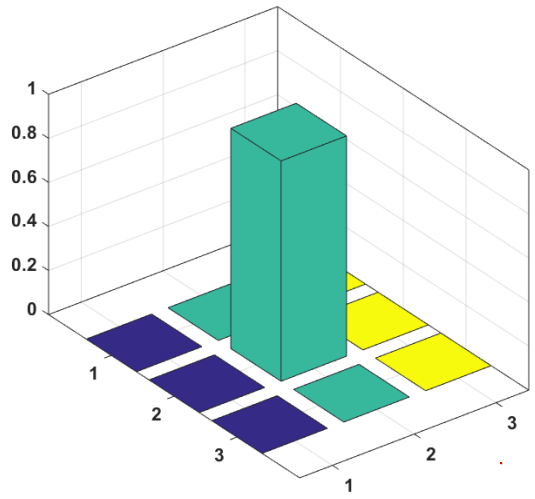


Output : 7 x 7 Gaussian filter (var =2)

# Linear Filtering – Other examples

❑ **Example -4 (a) :**

$x(n) * \delta(n) = x(n)$




$h(m,n) = \delta(m,n)$

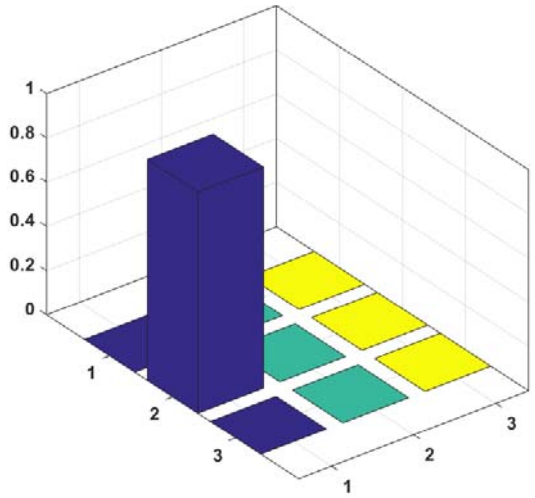
$*$

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

$=$

# Linear Filtering – Other examples

- **Example -4 (b) :**



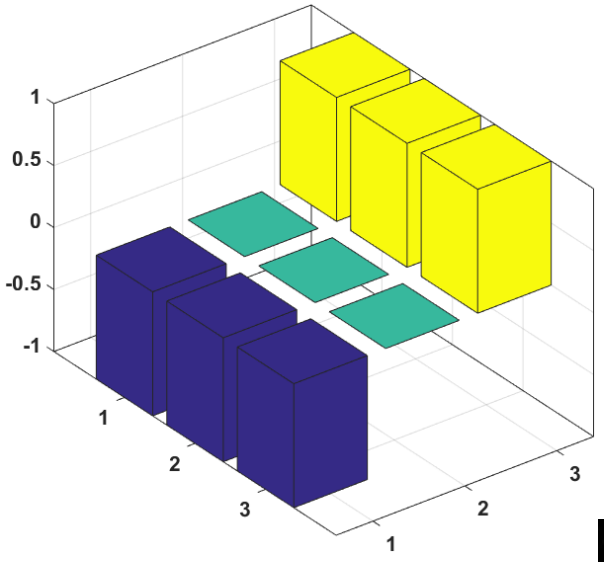

$*$

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

$=$

# Linear Filtering – Other examples

- **Example -4 (c) :**




$*$

| | | |
|---|---|---|
| −1 | 0 | 1 |
| −1 | 0 | 1 |
| −1 | 0 | 1 |

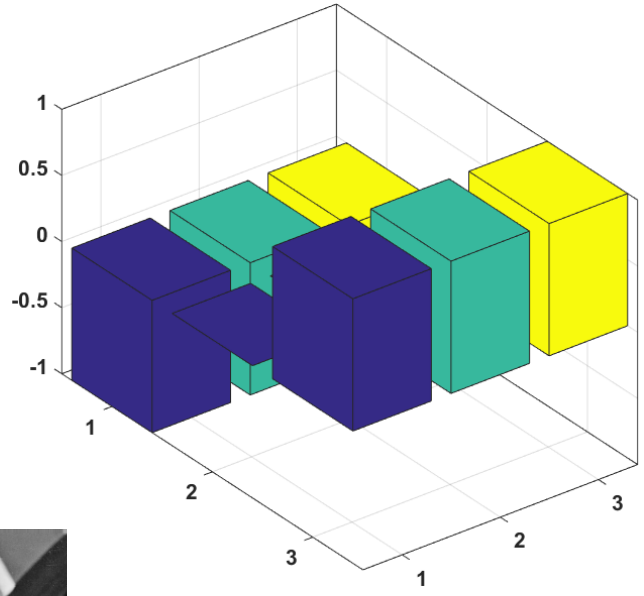
Prewitt Mask

$=$

Vertical edges are highlighted

# Linear Filtering – Other examples

- **Example -4 (d) :**




$*$

| | | |
|---|---|---|
| −1 | −1 | −1 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |

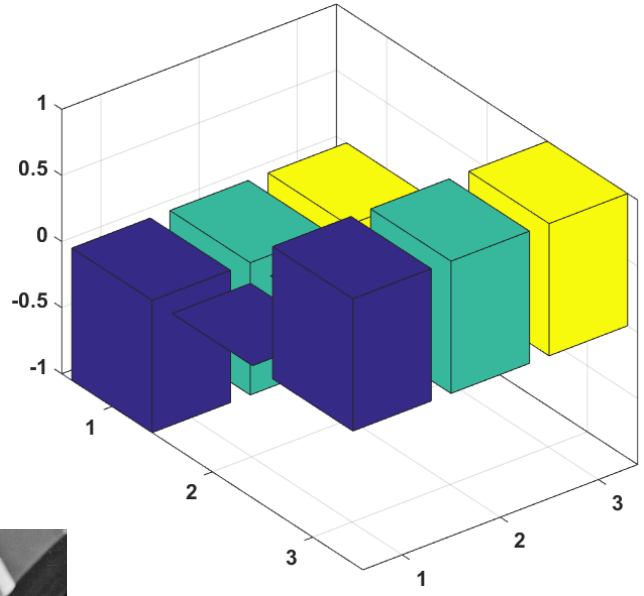
Prewitt Mask

$=$

Horizontal edges are highlighted

# Linear Filtering – Other examples

❑ **Example -4 (e) :**




$*$

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

Sobel Mask

$=$

Horizontal edges are highlighted

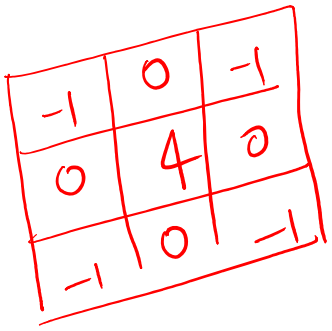
# Linear Filtering – Other examples

- **Example -4 (f) :**

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

Laplacian Mask

Edges are highlighted



# Non-linear Filtering : Median Filtering

- Compute some mathematical operations over the neighbourhood of pixel (x,y) in order to modify it.
- 1D-example
  - Median (3, 1, 4, 1, 0, 5, 67) = 3 ; However, mean is high because of "67"
- 2D-example
  - Median ( | 3 | 1 | 4 |<br>| 1 | 0 | 5 |<br>| 2 | 67 | 1 | ) = | | | |<br>| | 2 | |<br>| | | |

Input image Output image

- Median ( | 3 | 1 | 4 | 0 | 5 |<br>| 1 | 0 | 5 | 67 | 1 |<br>| 2 | 67 | 1 | 3 | 1 |<br>| 3 | 1 | 4 | 1 | 0 |<br>| 3 | 1 | 4 | 0 | 5 | ) =

| | | | | |
|---|---|---|---|---|
| | 2 | 3 | 3 | |
| | 2 | 3 | 1 | |
| | 3 | 1 | 1 | |

Input image Output image

# Non-linear Filtering : Median Filtering

Input image

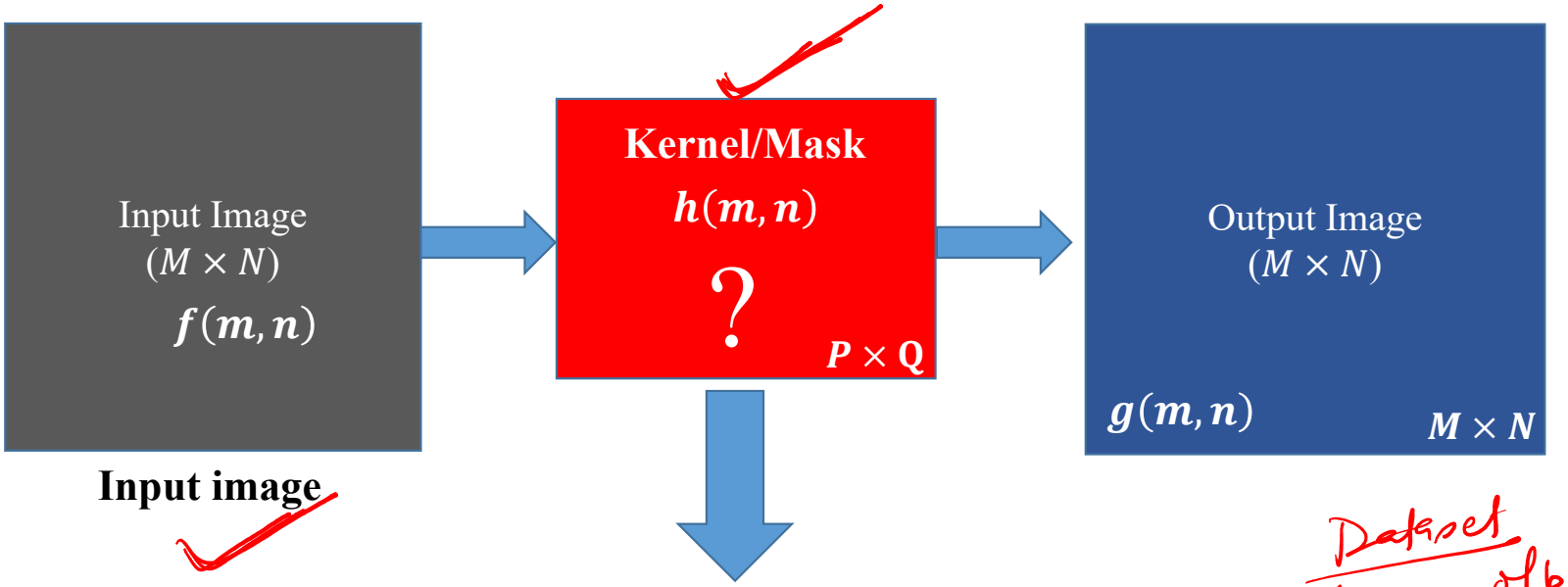
salt & pepper

Mean filter output

Median filter output

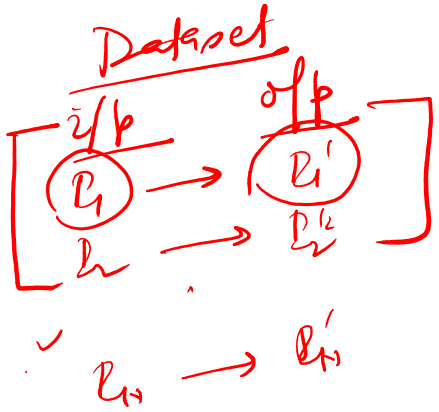
Gaussian filter output

# Filtering in Research




**Input image**

Needs to be learned



# Reference


- Richard Szeliski, Computer Vision: Algorithms and Applications, Springer, 2010 (online draft),
- Mubarak Shah, "Fundamentals of Computer Vision" (Online available)
- Ian Goodfellow, Yoshua Bengio and Aaron Courville, "Deep Learning" (Online available)

# Acknowledgement!

Sources for this lecture include materials from works by Szeliski, Abhijit Mahalanobis, Sedat Ozer, Ulas Bagci, Mubarak Shah, Antonio Torralba, and others. References are given for the source image contents.

# Queries!