Image Filters

modify image pixels based on some function of a local neighbourhood of each pixel

Box filter

$$\begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix}$$

Gaussian filter

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

edge filter

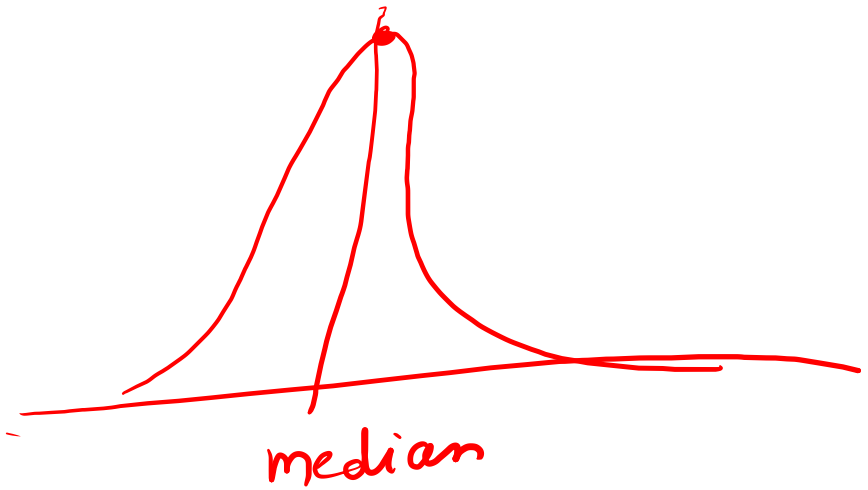
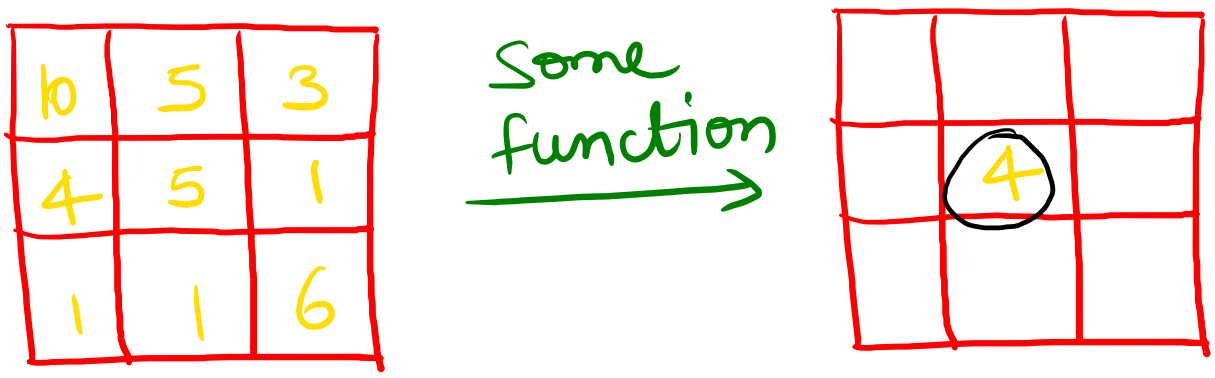


Image Filters

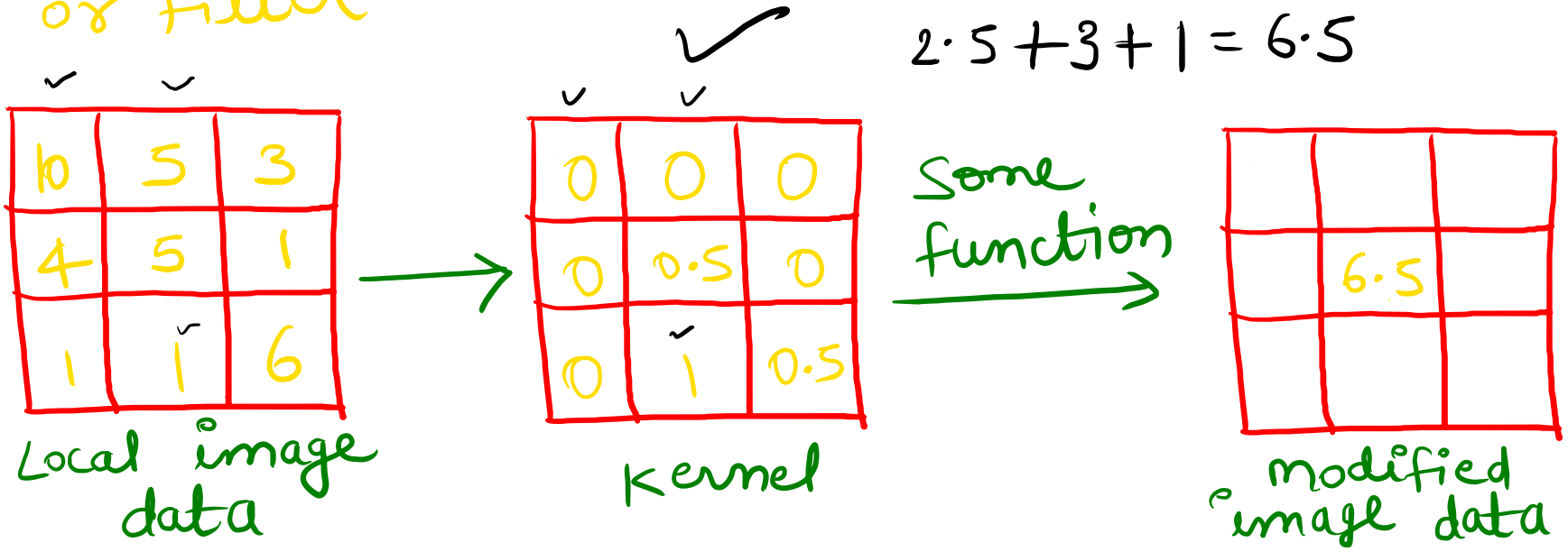
modify image pixels based on some function of a local neighbourhood of each pixel

$36/9 = 4$



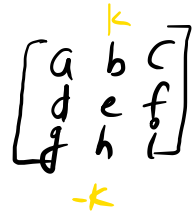
Linear Filters

- Replace each pixel by linear combinational (a weighted sum) of neighbours
- Linear combinations called kernel, mask or filter



Linear Filter: Cross-Correlation

Given a kernel of size $(2K+1) \times (2K+1)$



* Correlation defined as

$$Cr(i,j) = \underbrace{\frac{1}{(2K+1)^2}}_{\text{uniform weight to each pixel}} \underbrace{\sum_{u=-K}^{K} \sum_{v=-K}^{K} I(i+u, j+v)}_{\text{Loop over pixels is considered neighbourhood around } I(i,j)}$$

* Cross correlation is defined as

$$G_1(i,j) = \sum_{u=-K}^{K} \sum_{v=-K}^{K} H(u,v)\, I(i+u, j+v)$$

$H(u,v)$ → Non uniform weights

* Cross-correlation is denoted as

$$G_1 = H \otimes I$$

$$\begin{bmatrix} 1/8 & 1/16 \\ 0 & 1/8 \end{bmatrix}$$

$$\frac{1}{16}\begin{bmatrix} 2 & 1 \\ 0 & 2 \end{bmatrix}$$

$$\frac{1}{24}\begin{bmatrix} 1 & 0 & 1 \\ 2 & 4 & 1 \end{bmatrix}$$

$\frac{1}{4}$ $\frac{1}{2}$

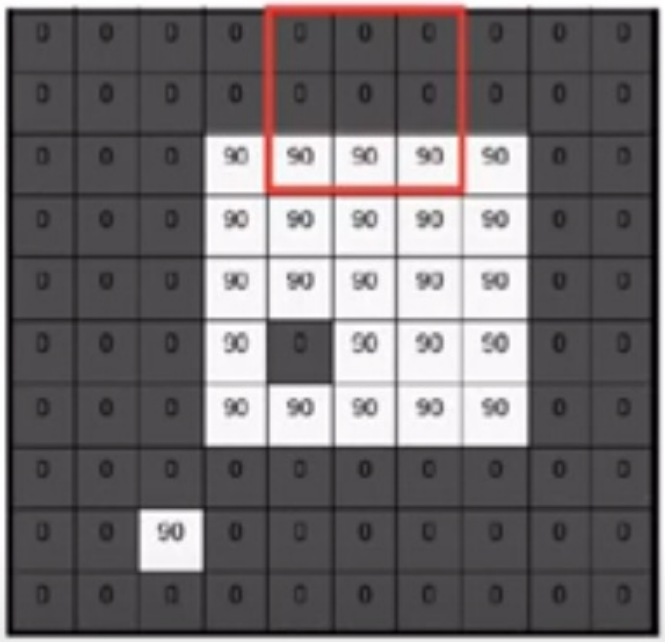
$\frac{1}{9}$

$\frac{1}{18}$

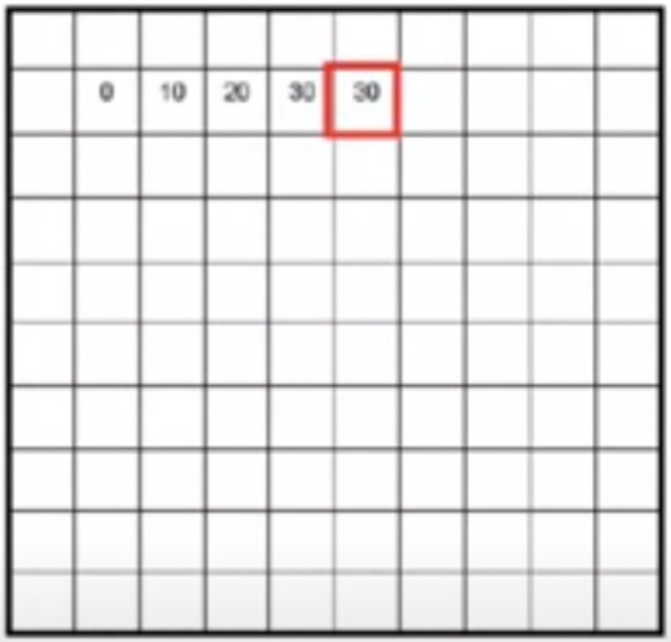
* Can be viewed as dot product between local neighbourhood and kernel for each pixel
* Entries of kernel or mask $H(u, v)$ called filter coefficients

Local operations (moving average linear filter)

$I(i,j)$



$G(i,j)$

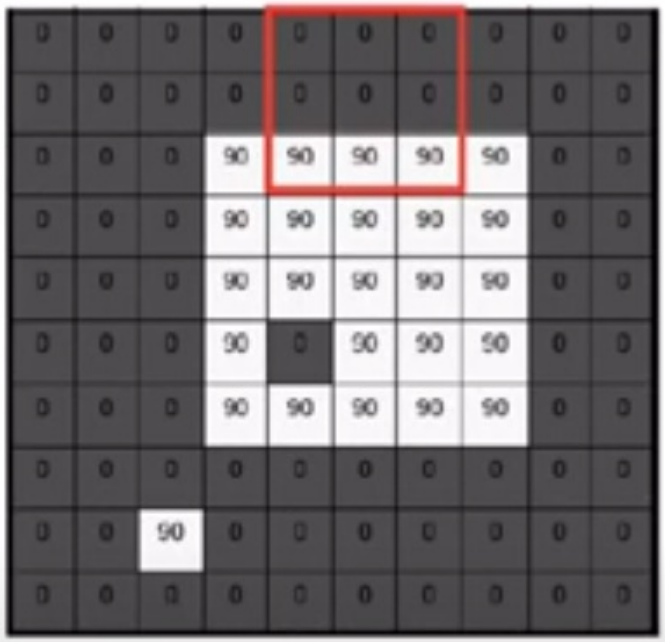


$\otimes H(u,v)$

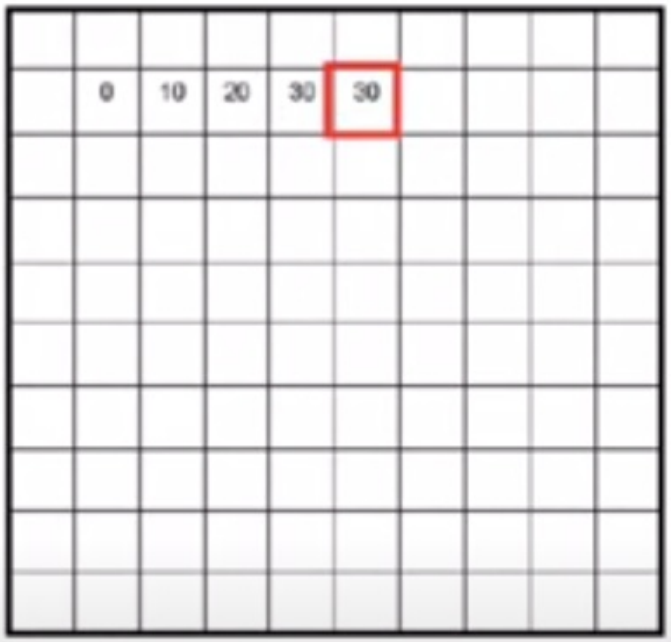
$\frac{1}{9}$ [1 1 1] 3x3

Local operations (moving average linear filter)

$I(i,j)$

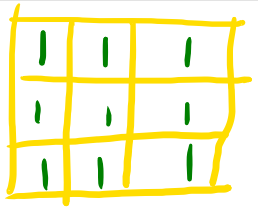


$G(i,j)$



$\otimes H(u,v)$

$\frac{1}{9}$

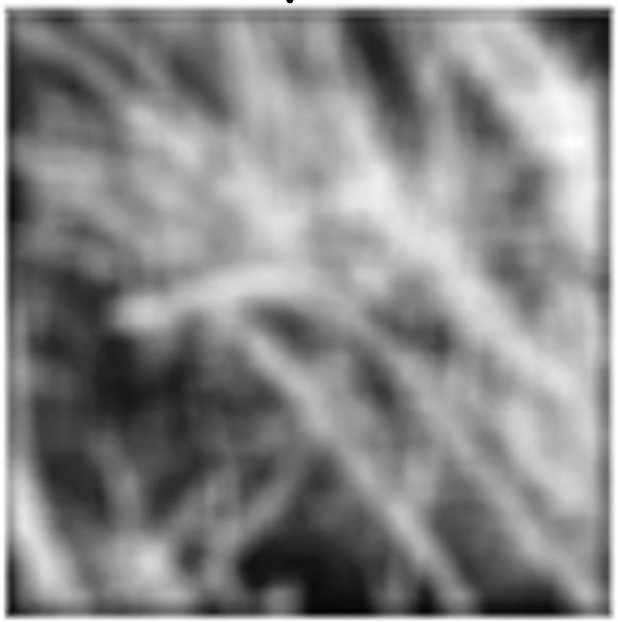


- Box filter

Effect of Moving average filter

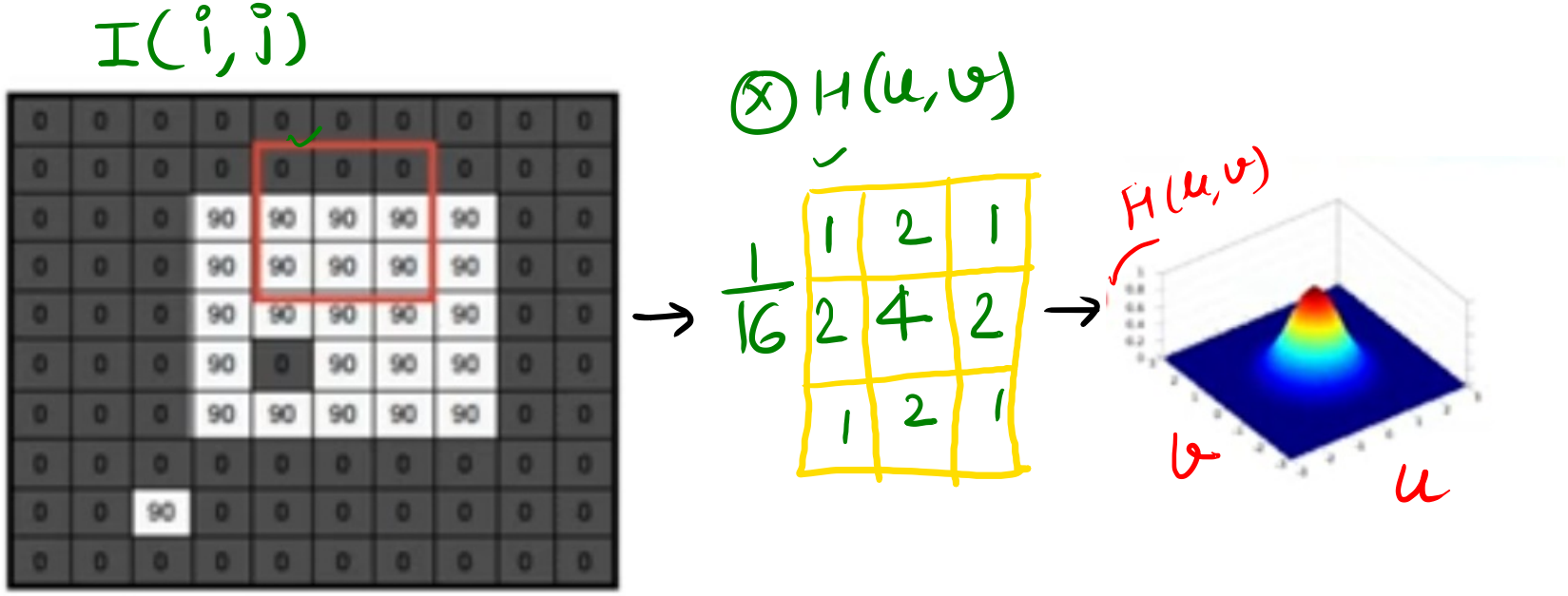
Input

output



Gaussian Average filter

what if want nearest neighbouring pixels to have the most influence on the output



This kernel is an approximation of a 2D Gaussian function

$$h(u,v) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{u^2+v^2}{2\sigma^2}}$$

$u = 0$

$$e^{-\frac{u^2+v^2}{2\sigma^2}} = \frac{1}{e^{\frac{u^2+v^2}{2\sigma^2}}}$$

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \text{- - - -}$$

e power x = 1+x+sqr x/fact (2)+ cube (x)/fact (3) + - - -

$e^1 =$

e power 1 = 1 + 1/1! + 1/2! + 1/3! + 1/4! + 1/5! + - - - e power 1 = 2.718

e^1= 2.718281828459045

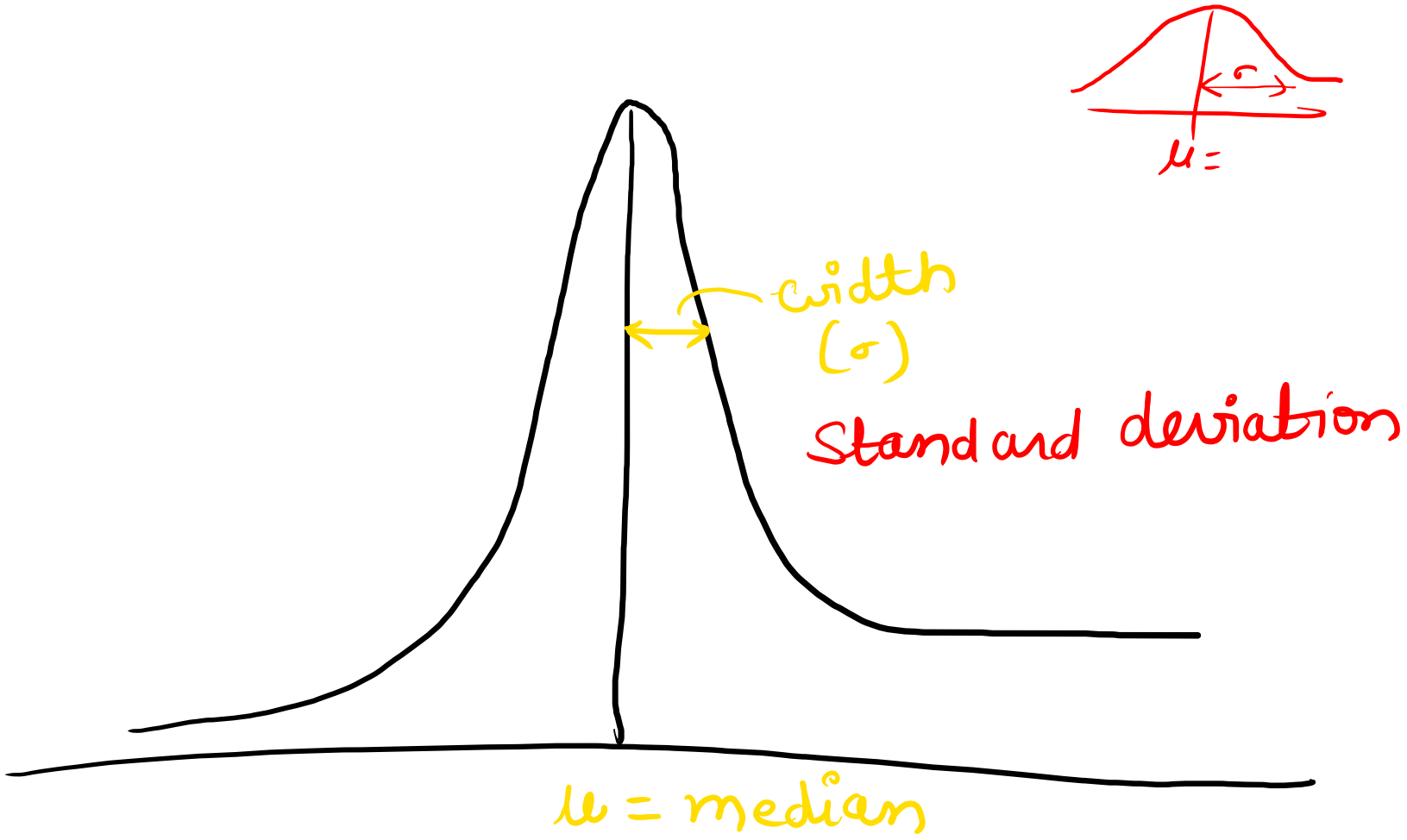
$e^{-1} =$ 1/2.718281828459045 = 0.36787944117 ✓

1/e

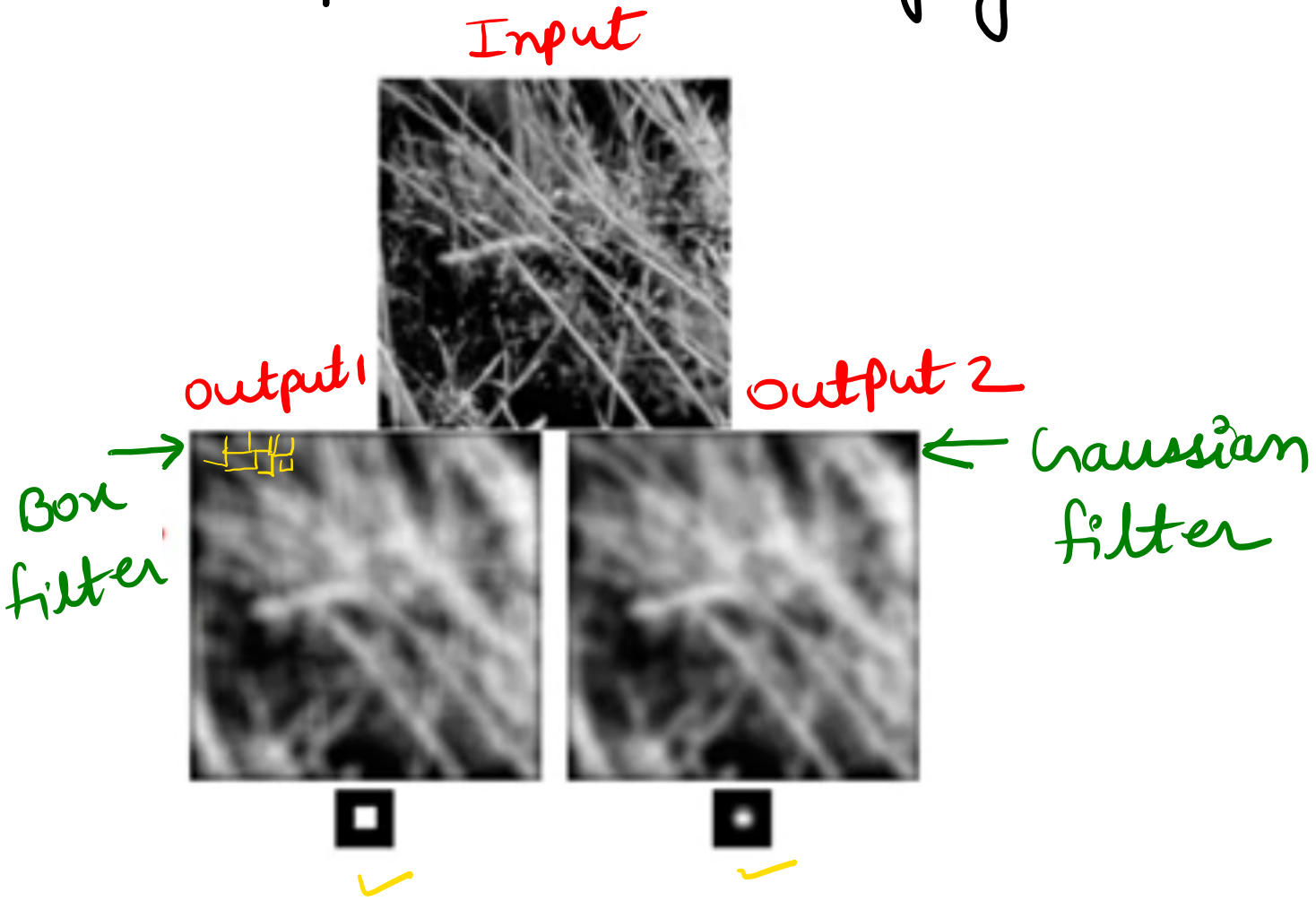
1D Gaussian function

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \qquad \mu = 0$$

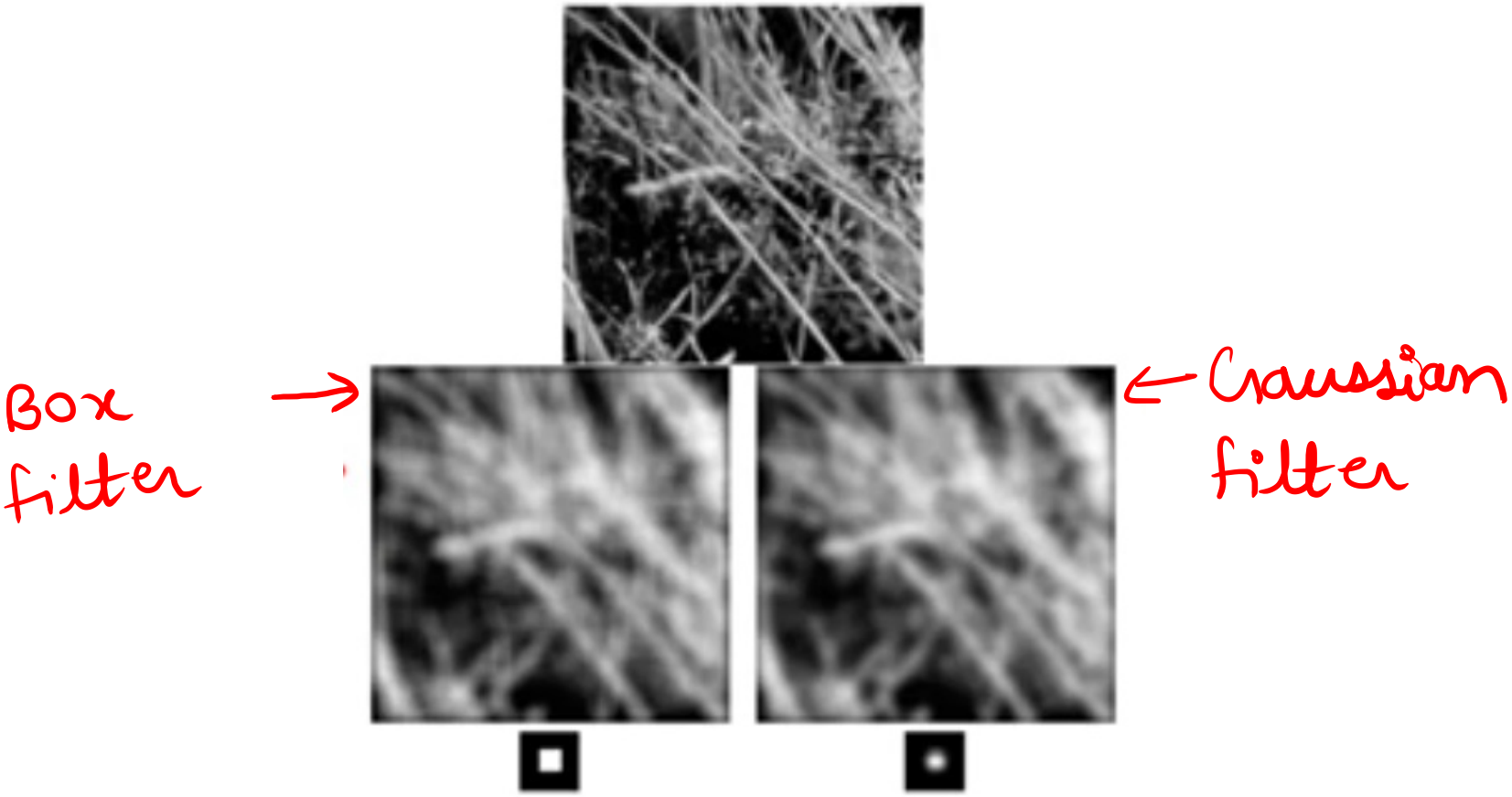
$$= \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$


σ
μ =
width
(σ)
Standard deviation
μ = median

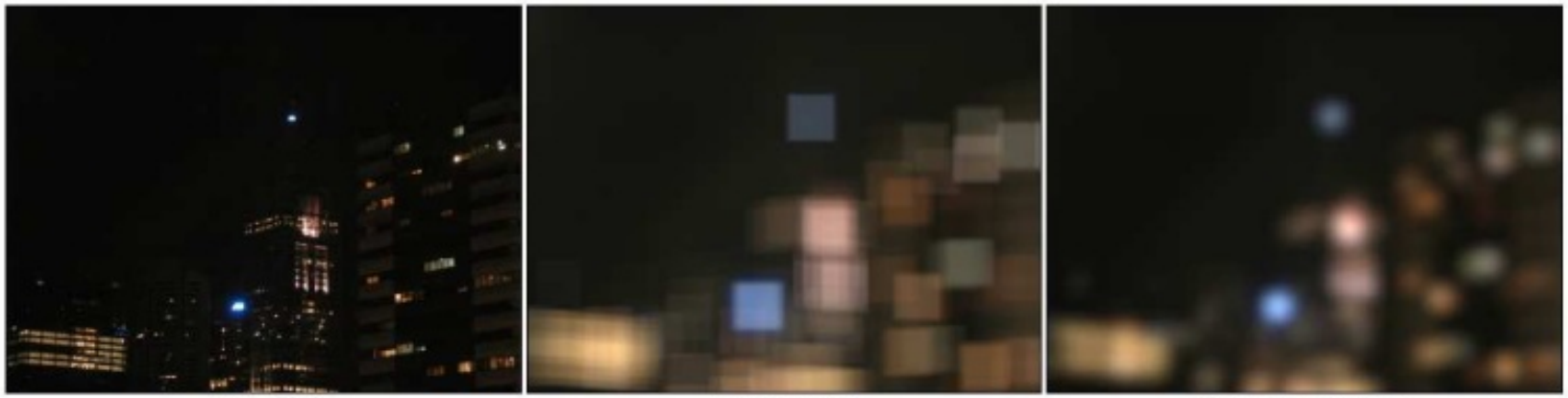
A Comparison in Averaging filter



A Comparison in Averaging filter



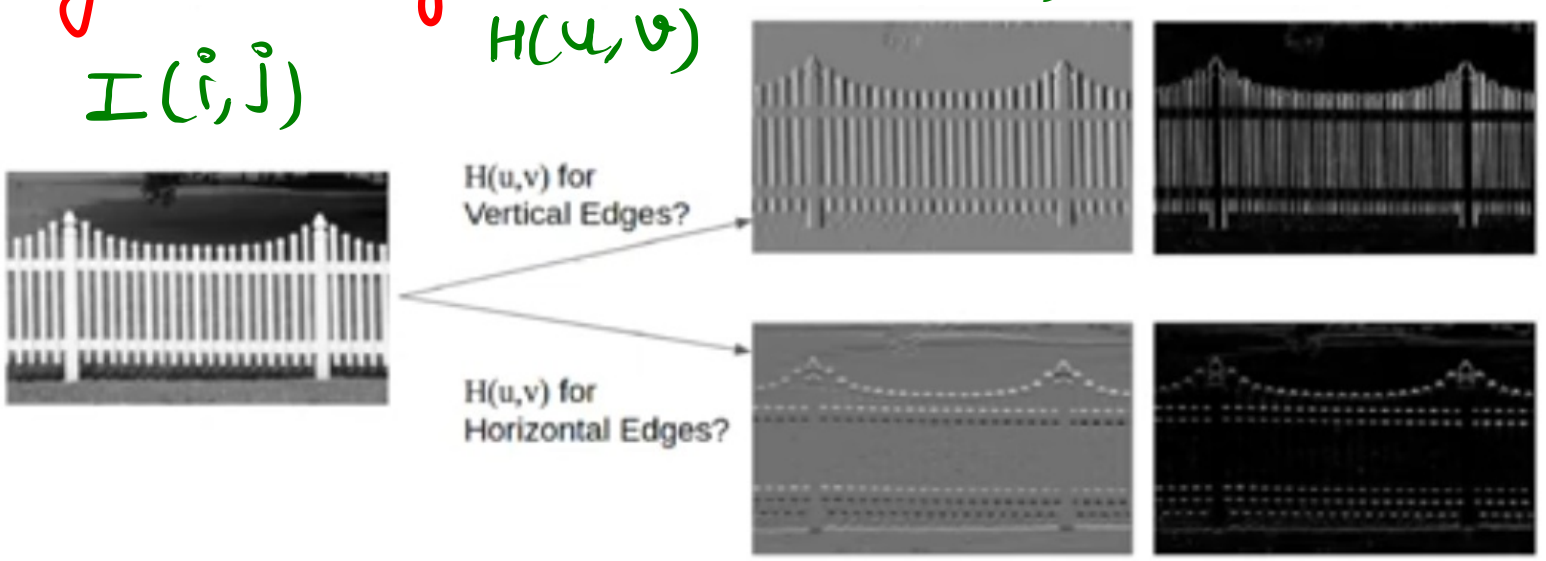
Box filter vs Gaussian Filter

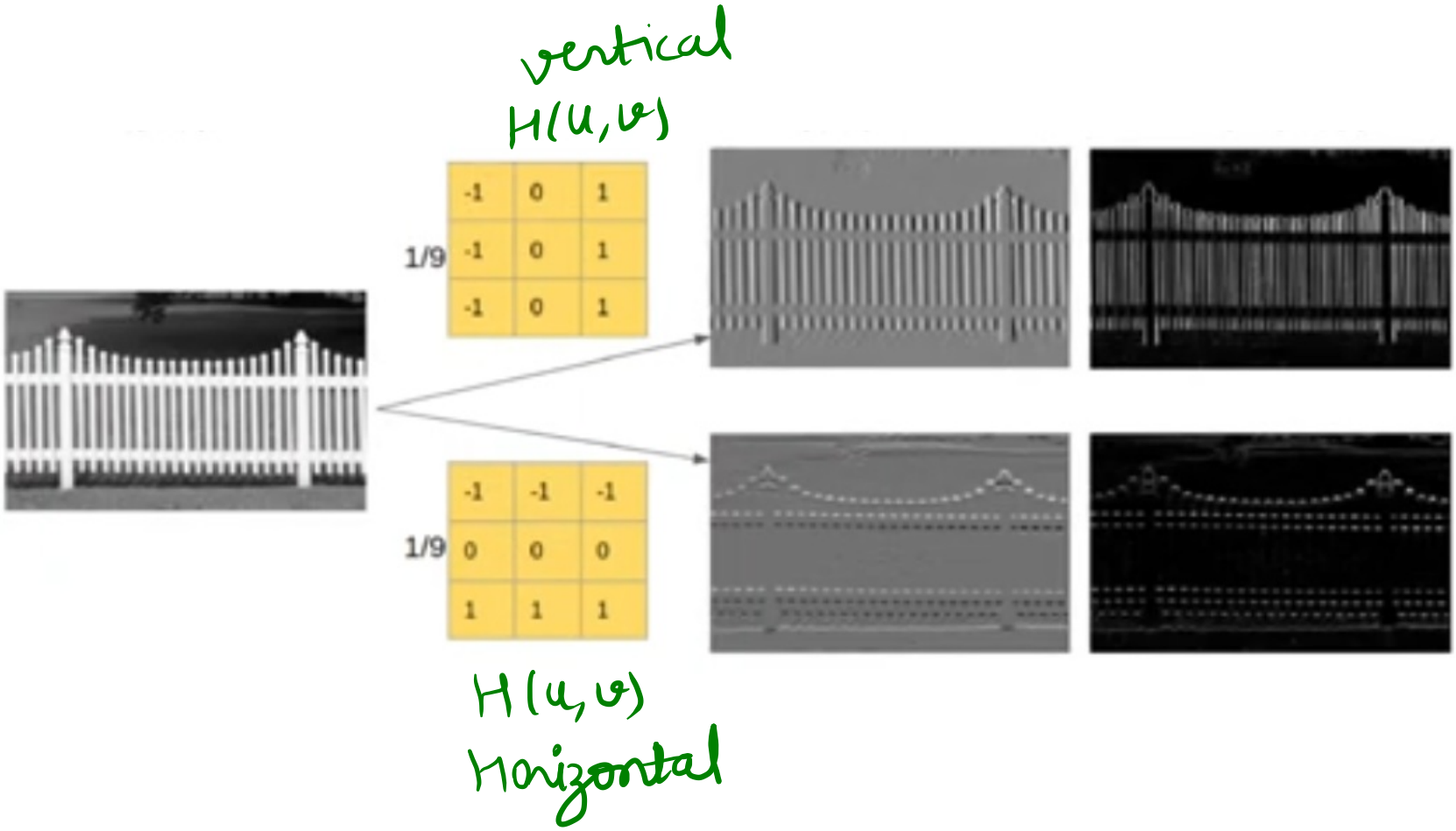


Original | Box filter | Gaussian filter

Edge filter

What should H look like to find edges in a given image?




vertical
H(u,v)
1/9
-1 0 1
-1 0 1
-1 0 1
1/9
-1 -1 -1
0 0 0
1 1 1
H(u,v)
Horizontal

Edge filter

- A vertical filter would have -1, -1, -1 on one side and 1, 1, 1 on the other side.
- So effectively filter looks for the places in the image where the significant difference between the left of the pixel and right of the pixel vertically.
- You try to exaggerate those pixels in the output image.

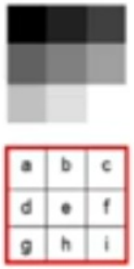
- A horizontal edge filter will do the same thing, but along the horizontal directions.

- The normalizing factor is 1/9, but you typically do not need that in an edge filter because we are not interested in absolute edge value, but we are interested in high intensity where there is an edge.

Beyond Correlation (limitation of cross-correlation)

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

$I(i,j)$ $\otimes H(u,v)$ $G(i,j)$

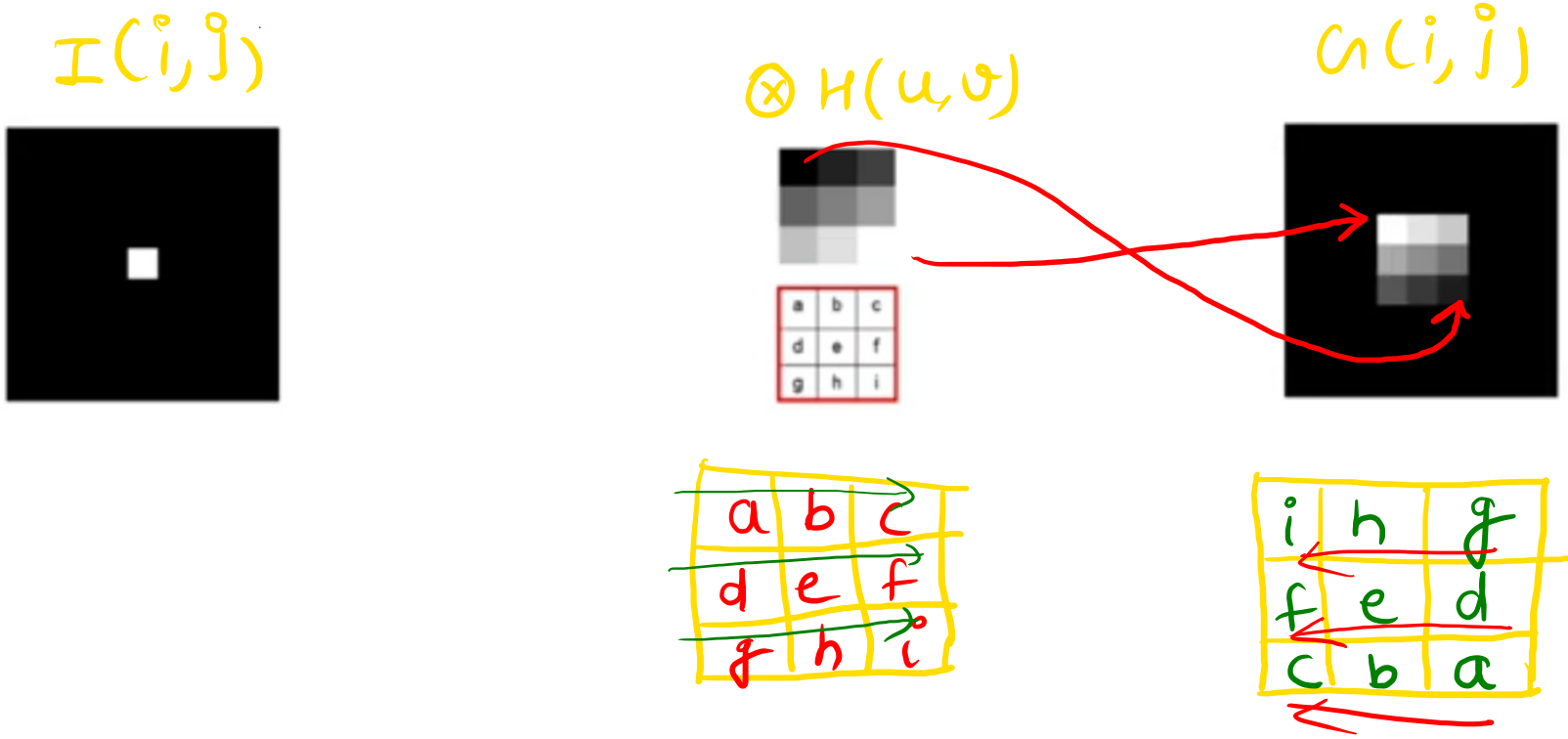


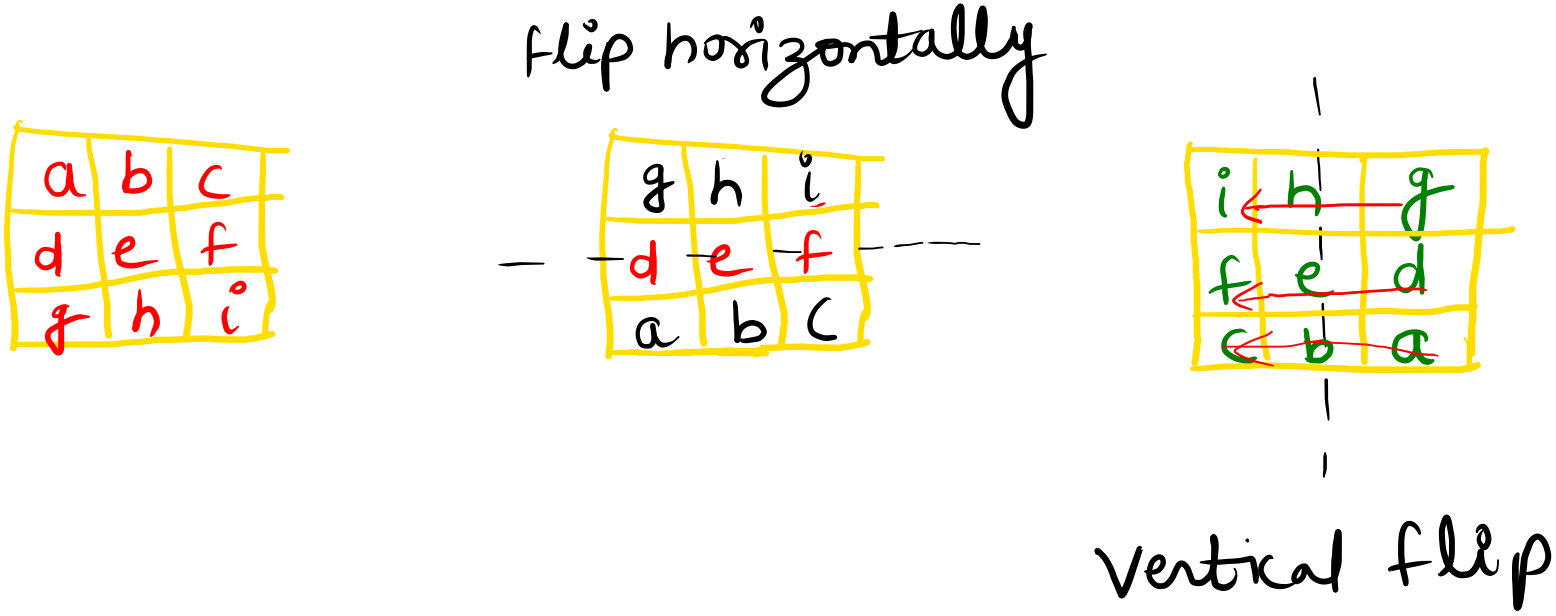
?

↓

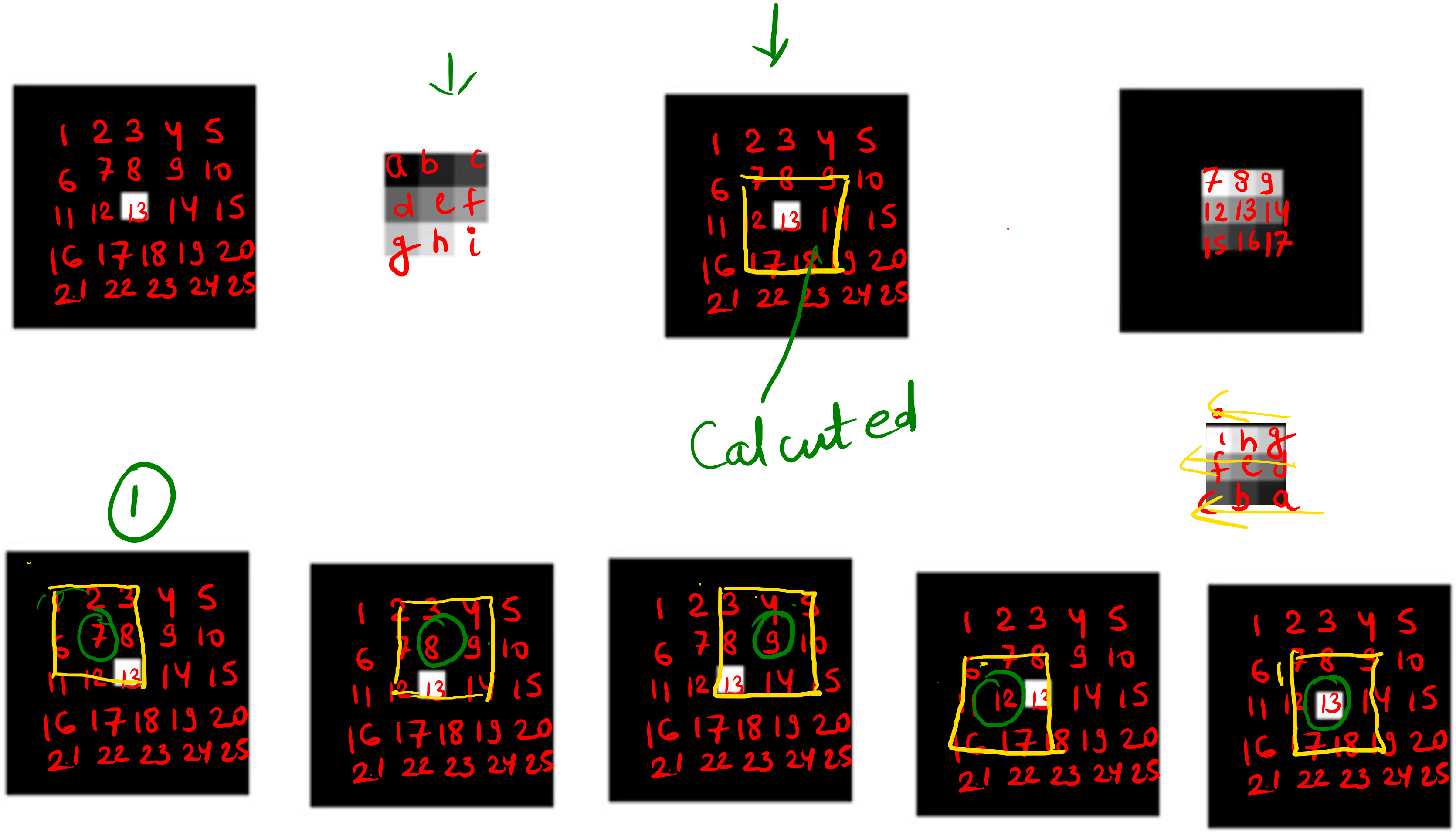
neither box filter nor Gaussian filter

Detailed Explanation

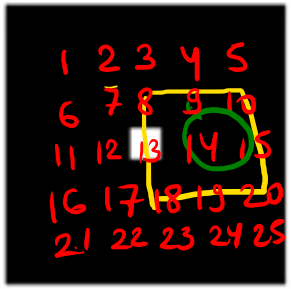


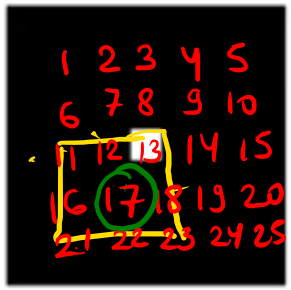


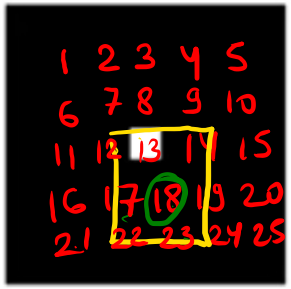
double flipping

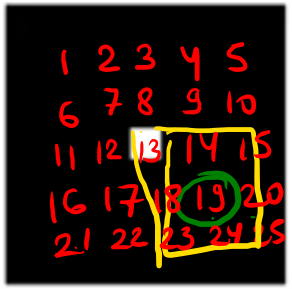


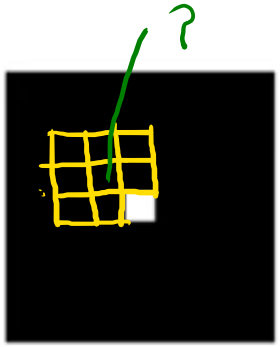
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25
a b c
d e f
g h i
7 8 9
12 13 14
15 16 17
i h g
f e d
c b a
Calcuted
1

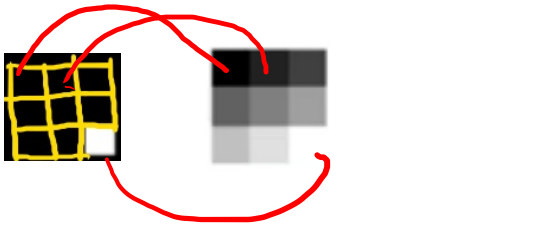

1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25

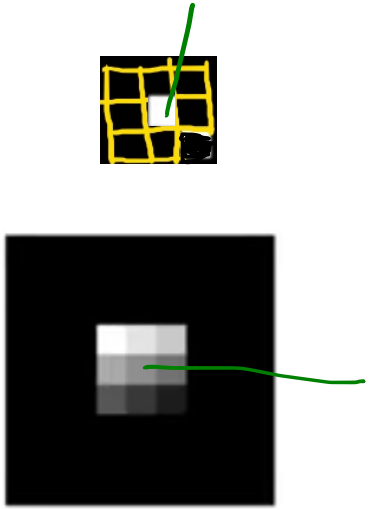

1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25

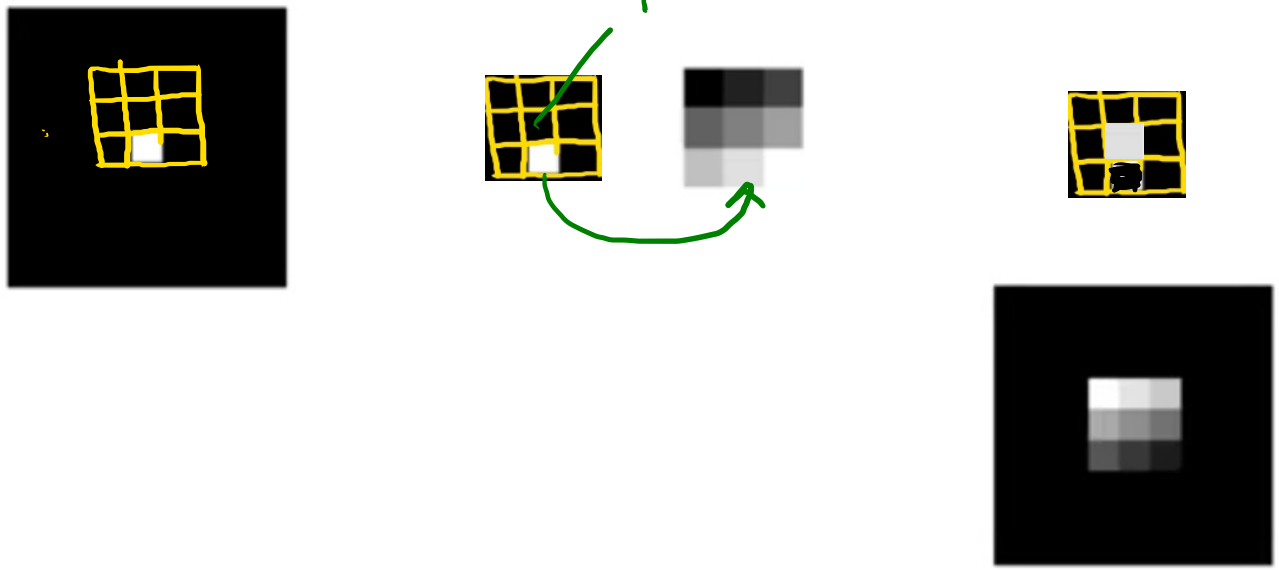

1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25


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


?

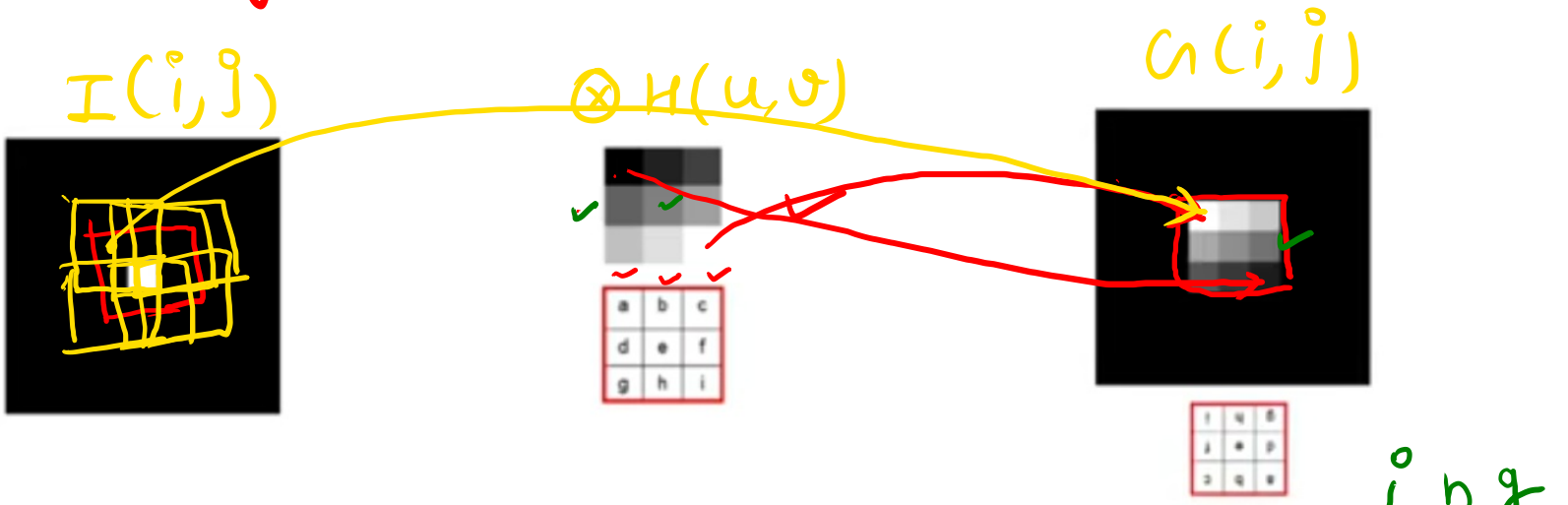

filter





Beyond Correlation

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



- The output image is completely flipped
- The bottom right swapped with top left.
- your output is the double flipped version of your input
- white value is the only value that will get multiplied by 1 and other kernel value get multiplied with by corresponding value 0

- which means the output at this particular location is going to be white and we will keep doing this over and over again at each pixel

Convolution

Given a kernel of size $(2k+1) \times (2k+1)$

* convolution defined as

$$G_I(i,j) = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} H(u,v)\, I(i-u, j-v)$$

* Equivalent to filp the filter in both directions (bottom to top, right to left) and apply correlation

* Denoted by $G = H * I$

↑ Convolution

Cross-correlation vs convolution

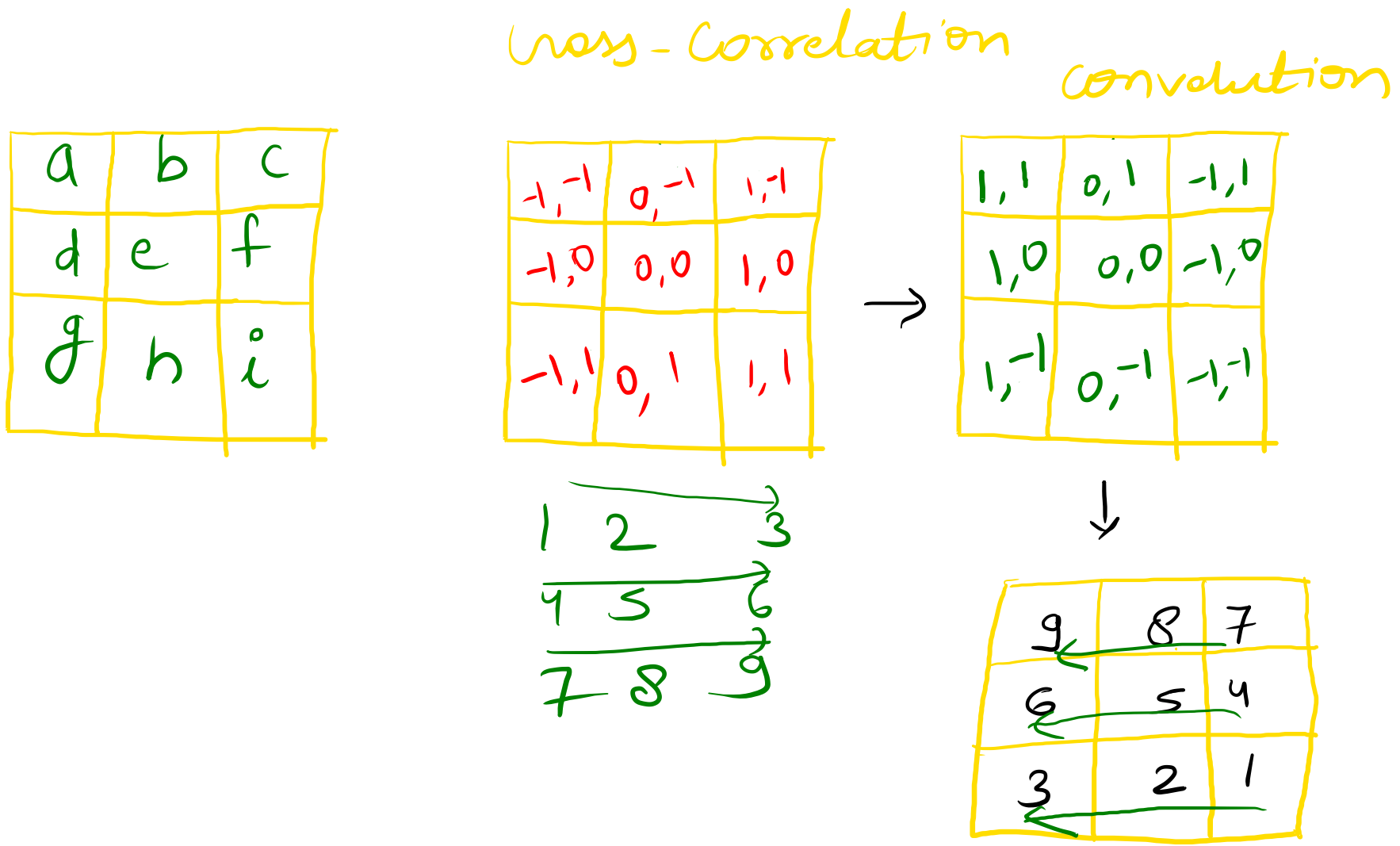
$G = H \otimes I$ (cross-correlation)

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H(u,v)\, I(i+u, j+v)$$

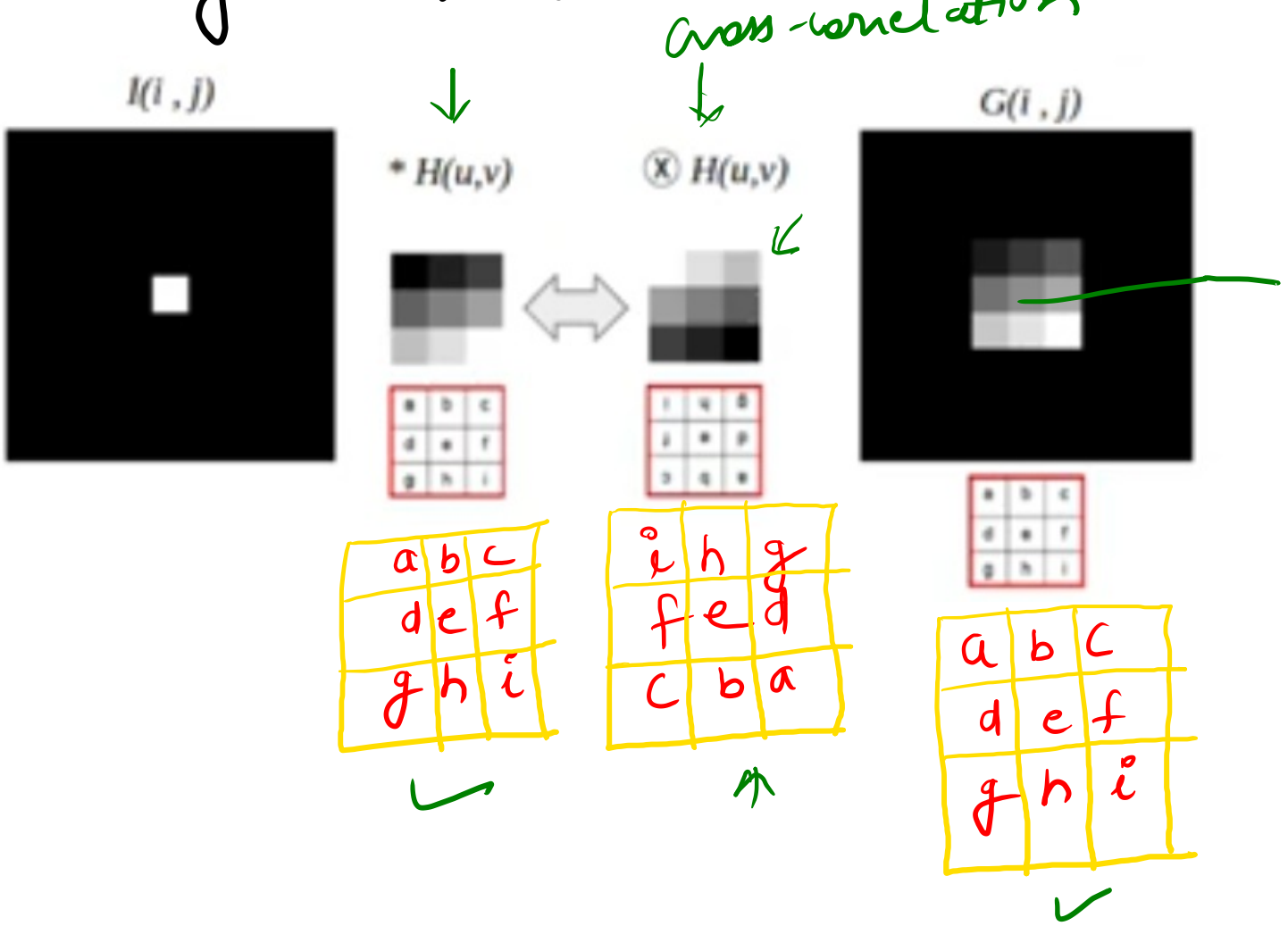
$G = H * I$ (convolution)

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H(u,v)\, I(i-u, j-v)$$

(-1)

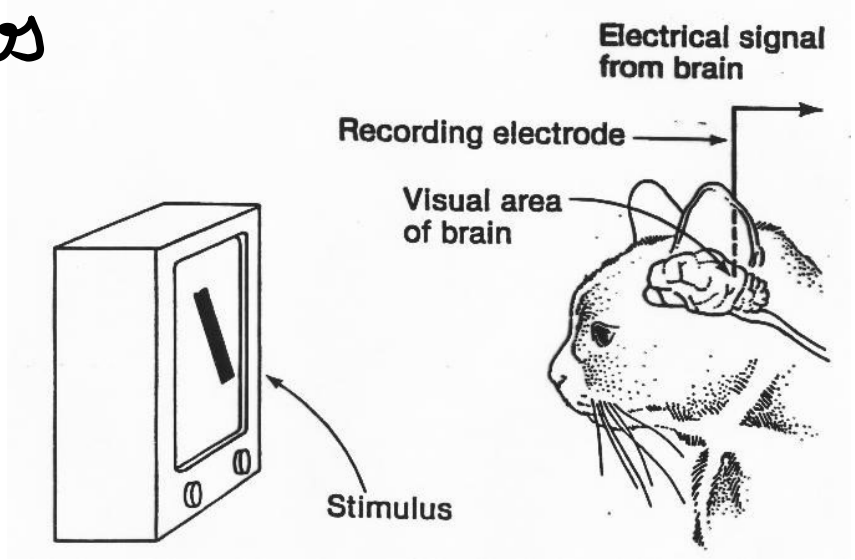

cross-correlation
convolution
a b c
d e f
g h i
-1,-1 0,-1 1,-1
-1,0 0,0 1,0
-1,1 0,1 1,1
1,1 0,1 -1,1
1,0 0,0 -1,0
1,-1 0,-1 -1,-1
1 2 3
4 5 6
7 8 9
9 8 7
6 5 4
3 2 1

Introducing Convolution



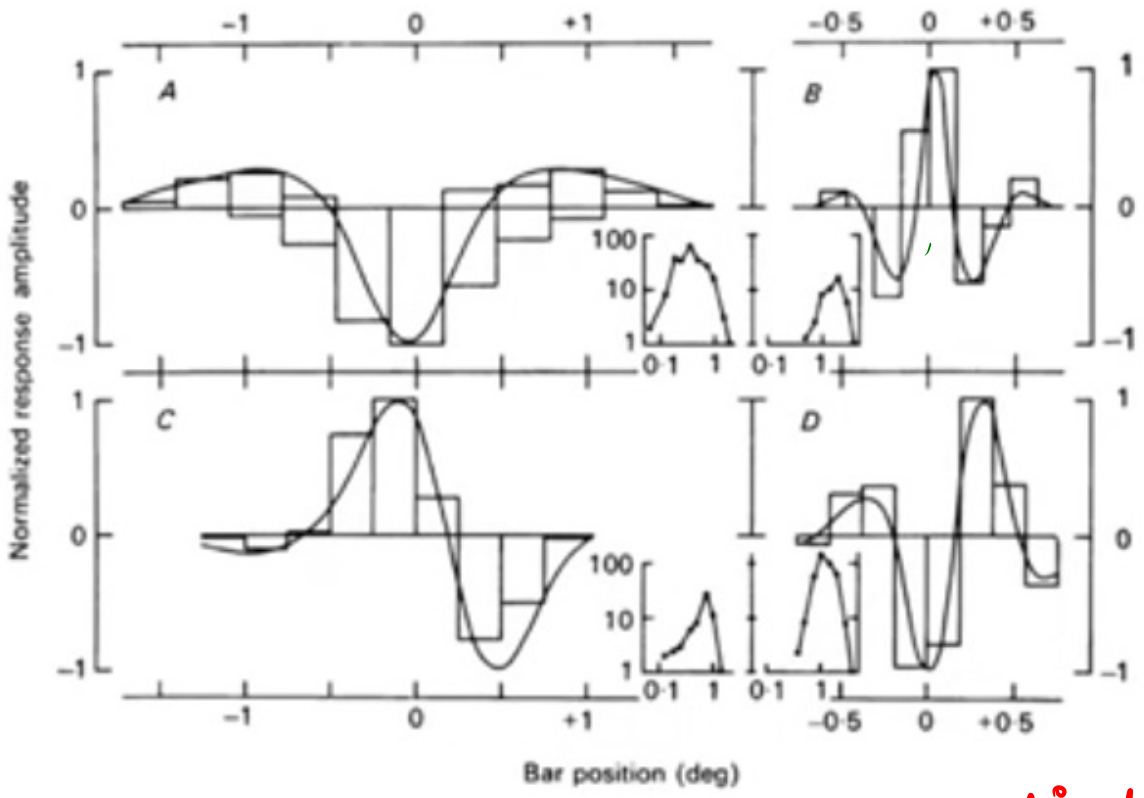
Recall early history

This experiment shows that simple and complex neurons exist.



The visual system starts with simple structures like oriented edges.

Linear Summation in the visual Cortex



simple cells performs linear spatial summation over their receptive fields

Linear Shift-Invariant Operators

* Both correlation and convolution are Linear Shift-Invariant operators, which obey

(1) Linearity (or Superposition principle)

$$I \circ (h_0 + h_1) = I \circ h_0 + I \circ h_1$$

(2) Shift-invariance: shifting (or translating) a signal commutes with applying the operator

$$g(i,j) = h(i+k, j+l)$$

$$(f \circ g)(i,j) = (f \circ h)(i+k, j+l)$$

* Equivalent to saying that effect of operator is the same everywhere. Why do we need this in computer vision?

Properties of Convolution

(1) Commutative: $a \times b = b \times a$

- Conceptually no difference between filter and signal

(2) Associative: $(a \times b) \times c = a \times (b \times c)$

- We often apply filters one after the other $(((a \times b1) \times b2) \times b3)$
- This is equivalent to applying one filter: $a \times (b1 \times b2 \times b3)$

(3) Distributive over addition:

$$a \times (b+c) = (a \times b) + (a \times c)$$

- We can combine the responses of a signal over two or more filters by combining the filters

(4) Scalars factor out: (k: constant) $k\,a \times b = a \times k\,b = k(a \times b)$

(5) Identity: Unit impulse, $e = \{\cdots, 0, 0, 1, 0, 0, \cdots\}$

$$a * e = a$$

(6) One more important property of convolution is separability

Separability

- Convolution operator requires K^2 operations per pixel, where K is the width (and height) of a convolution kernel.
- This is costly operation. How can we reduce cost?

- For certain filters, can be speed up by performing a 1D horizontal convolution followed by a 1D vertical convolution, requiring 2K operations

⇓

Convolution kernel is separable

$$K = vh^T$$

where v, h are 1D kernel, and K is 2D kernel

For Example 1.

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \Rightarrow u = v = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

$$v = h = \frac{1}{4}\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

2D kernel

$$h^T = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

$$K = v h^T$$

$A = A^T$

$A =$

1	2	1
2	4	2
1	2	1

$A^T =$

1	2	1
2	4	2
1	2	1

$A \cdot A^T = U$

$A^T \cdot A = U$

$$U = \begin{bmatrix} 1+4+1 & 2+8+2 & \\ & & \\ & & \end{bmatrix}$$

$$\frac{1}{4}\underset{3\times1}{\begin{bmatrix}1\\2\\1\end{bmatrix}} \times \frac{1}{4}\underset{1\times3}{\begin{bmatrix}1, 2, 1\end{bmatrix}}$$

$$= \frac{1}{16}\underset{3\times3}{\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}}$$

Characteristics equation

$$(A - \lambda I)\, X = 0$$

- A → original matrix
- λ → eigen values
- X → eigen vector
- I → Identity matrix

assume $(A - \lambda I)$

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$1-\lambda$	2	1
2	$4-\lambda$	2
1	2	$1-\lambda$

$A - \lambda I$

eigen value $1-\lambda(4-\lambda)(1-\lambda) - 2 \times 2 +$

$\ldots\ldots = 0$

$$\lambda^3 - S_1\lambda^2 + S_2\lambda - S_3 = 0$$

S_1 ↓ det(A) = 0

S_2 ↓ Submatrix = 0

S_3 ↓ addition of diagonal values

1	2	1
2	4	2
1	2	1

$$S_3 = 6$$

$$S_2 = 0$$

$$\begin{vmatrix} 4 & 2 \\ 2 & 1 \end{vmatrix} + \begin{vmatrix} 2 & 2 \\ 1 & 1 \end{vmatrix} + \begin{vmatrix} 2 & 4 \\ 1 & 2 \end{vmatrix}$$

$$0 \qquad 0 \qquad 0$$

$$\lambda^3 - 6 = 0$$

$$\lambda^3 = 6$$

$$\lambda = (6)^{1/3}$$

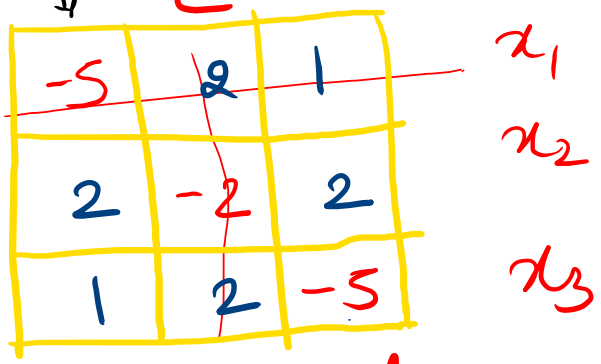
$\lambda = 6$

$(6)^{1/3}$

$(A - \lambda I)X = 0$

$1-6$

$[A - \lambda I]$

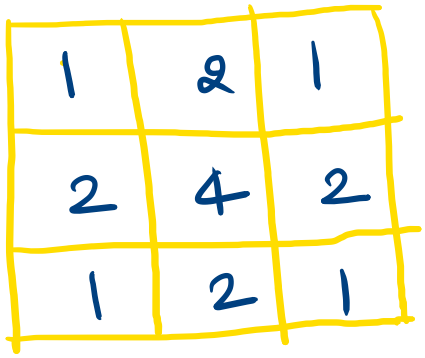


✓ Cramers rulee

$$\frac{x_1}{10-4} = \frac{-x_2}{-10-2} = \frac{x_3}{4+2}$$

$$\frac{x_1}{6} = \frac{-x_2}{-12} = \frac{x_3}{6}$$

$$\frac{1}{6} \quad 1 \quad 2 \quad 1$$



$$(A - \lambda I)X = 0$$

Example 2

$$\frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \Rightarrow \begin{matrix} v = ? \\ h = ? \end{matrix}$$

A

$$A^T = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

$A \cdot A^T = U$

$A^T \cdot A = V$

Example 2

$$\frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \Rightarrow v = \frac{1}{4}\begin{bmatrix} -1 \\ -2 \\ -1 \end{bmatrix} \quad h = \frac{1}{2}\begin{bmatrix} 1 & 0 & 1 \end{bmatrix}$$

Separable Convolution

How can we tell if a given kernel is separable?

Separable Convolution

How can we tell if a given kernel is separable?

- Visual inspection
- Analytically, look at the singular value decomposition (SVD), and if only one singular value is non-zero, then it is separable

$$K = U \Sigma V^T = \sum_i \sigma_i u_i v_i^T$$

where $\Sigma = \text{diag}(\sigma_i)$

$$\Sigma_i = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_K \end{bmatrix}$$

$\sqrt{\sigma_1} u_1$ and $\sqrt{\sigma_1} v_1$ are the vertical and horizontal kernels

Practical Issues

(1) Ideal size for the filter ?

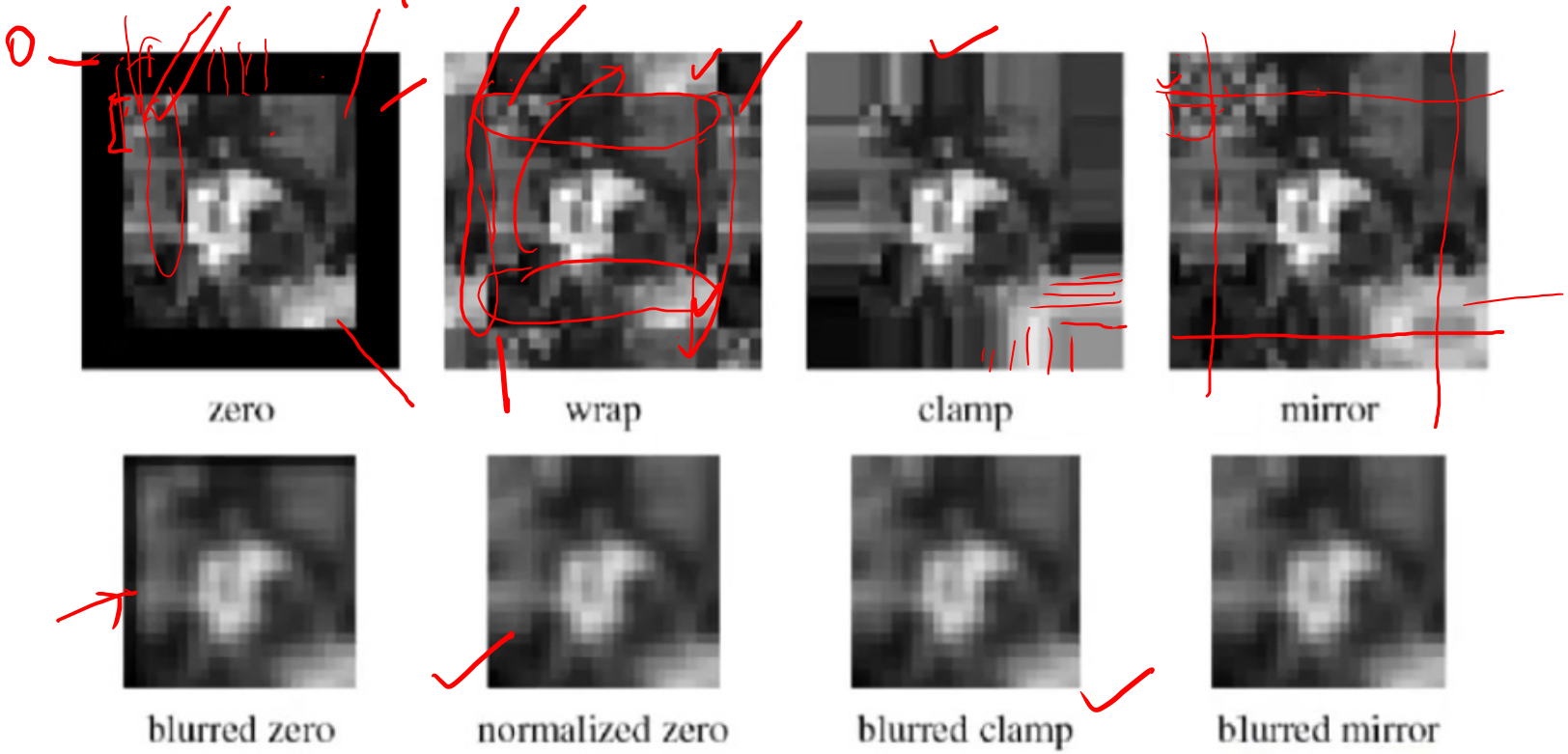
The bigger the mask

- more neighbours contribute
- smaller noise variance of output
- bigger noise spread
- more blurring
- computation intensive

(2) what about the boundaries? Do we lose information?

- without padding, we lose out information at the boundaries.
- we can use a variety of strategies such as zero padding, wrapping around, copy the edge

Different padding strategies

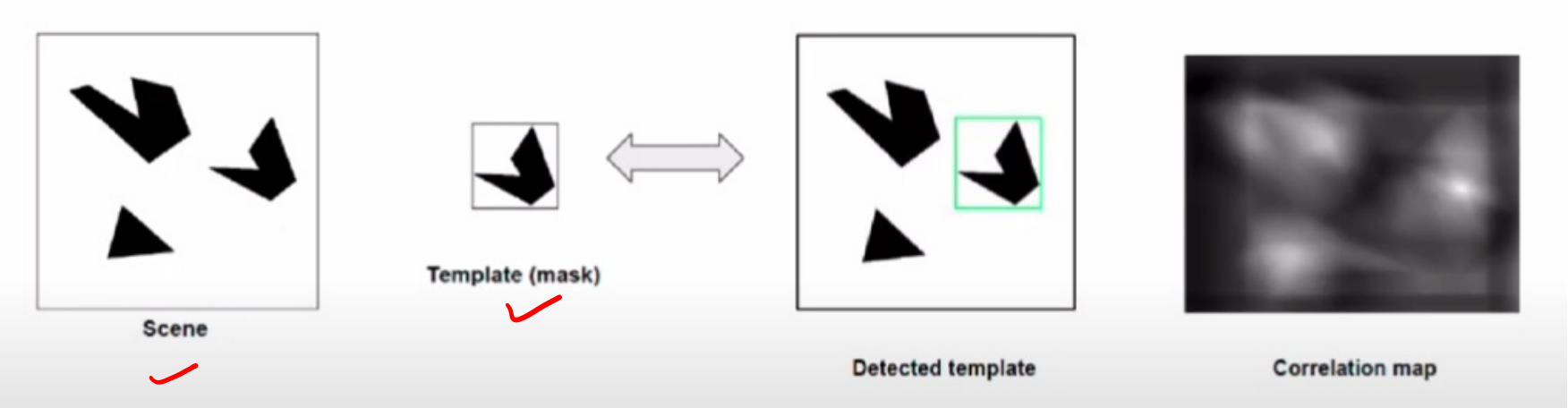


Q1. Do we need cross-correlation at all?

Q2. Are all filters always linear?

what is histogram equalization?

Is correlation still useful?



- It can be used for template matching
- Filters look like object you are intended to find ⇒ use Normalized Cross-Correlation (to control relative brightness) score to find a given pattern in an image

Is correlation still useful?

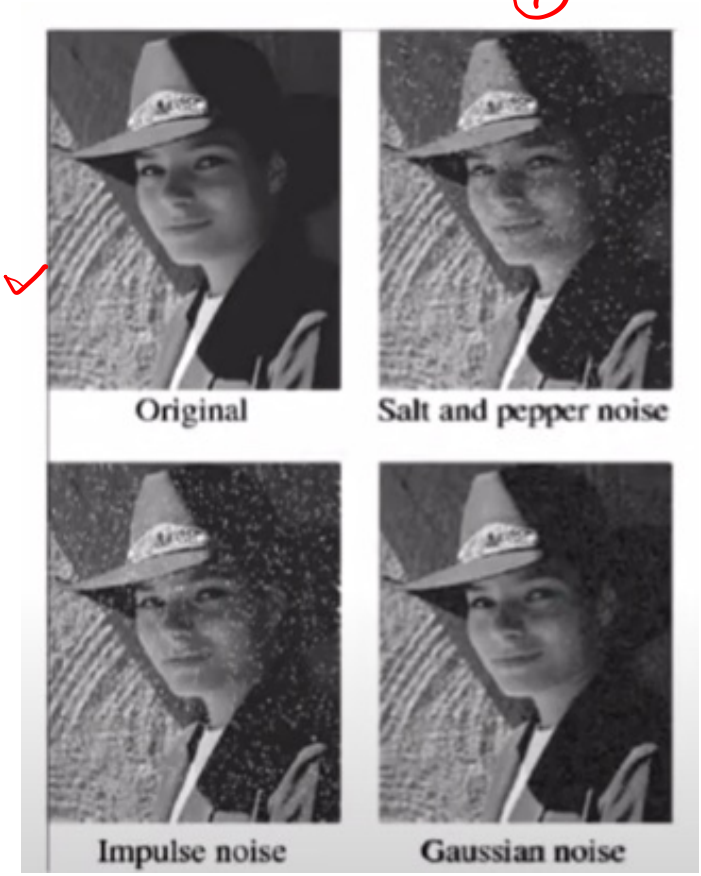
Scene

Template

Detected template

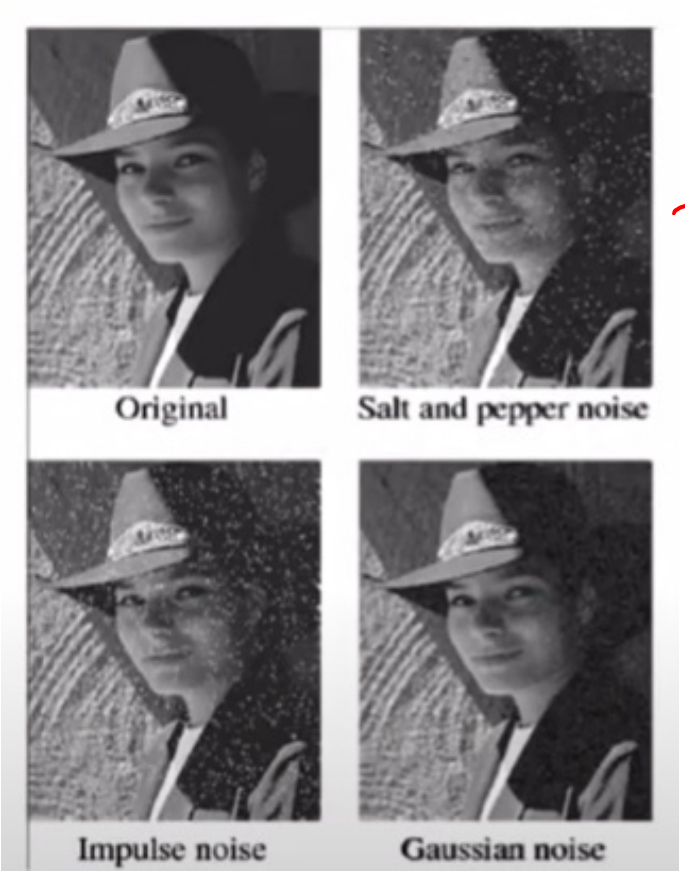
Even if the template is not identical to subimage in the scene, match can be meaningful, if scale, orientation and general appearance is right.

Non-Linear Filters



Different types of noise in images

noise distribution on adding different size Gaussian filters



- Salt and pepper noise has black and white specks
- Impulse noise has only white specks.
- Gaussian noise occurs due to addition of Gaussian to signal/Image at each location which distributes existing noise.

Irrespective of the size of the Gaussian filter, we are smudging around the salt and pepper noise and not really removing the noise.

Non linear filters: Median Filter

- Replace each pixel with Median value of all pixels in neighbourhood
- Properties
  - Non-linear
  - Does not spread noise
  - Can remove spike noise
  - Robust to outliers, but not good for Gaussian noise


10 11 10 0 0 1
9 10 11 1 0 1
10 9 10 0 2 1
11 10 9 10 9 11
9 10 11 9 99 11
10 9 9 11 10 10

I

10	11	10	0	0	1
9	10	11	1	0	1
10	9	10	0	2	1
11	10	9	10	9	11
9	10	11	9	99	11
10	9	9	11	10	10

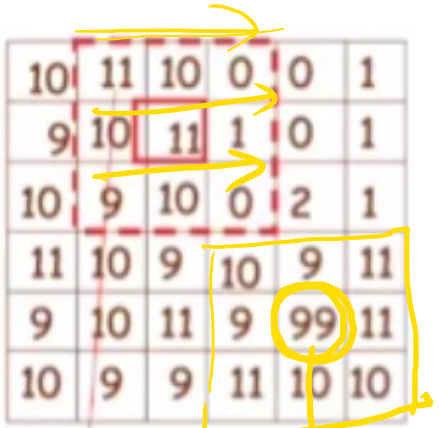
→

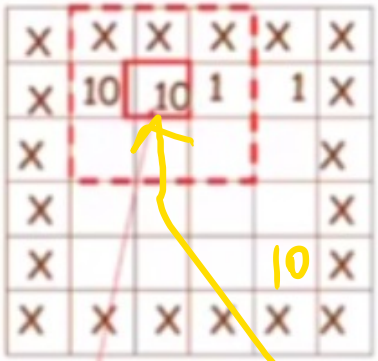
O

X	X	X	X	X	X
X	10				X
X					X
X					X
X					X
X	X	X	X	X	X

10, 11, 10, 9, 10, 11, 10, 9, 10 → Sort → 9, 9, 10, 10, 10, 10, 10, 11, 11


10 11 10 0 0 1
9 10 11 1 0 1
10 9 10 0 2 1
11 10 9 10 9 11
9 10 11 9 99 11
10 9 9 11 10 10





noise

Sort → 0, 0, 1, 9, 9, 10, 10, 10, 11, 11

median

9, 9, 10, 10, 10, 11, 11, 11, 99