

Announcement

1/30/2025 Thursday

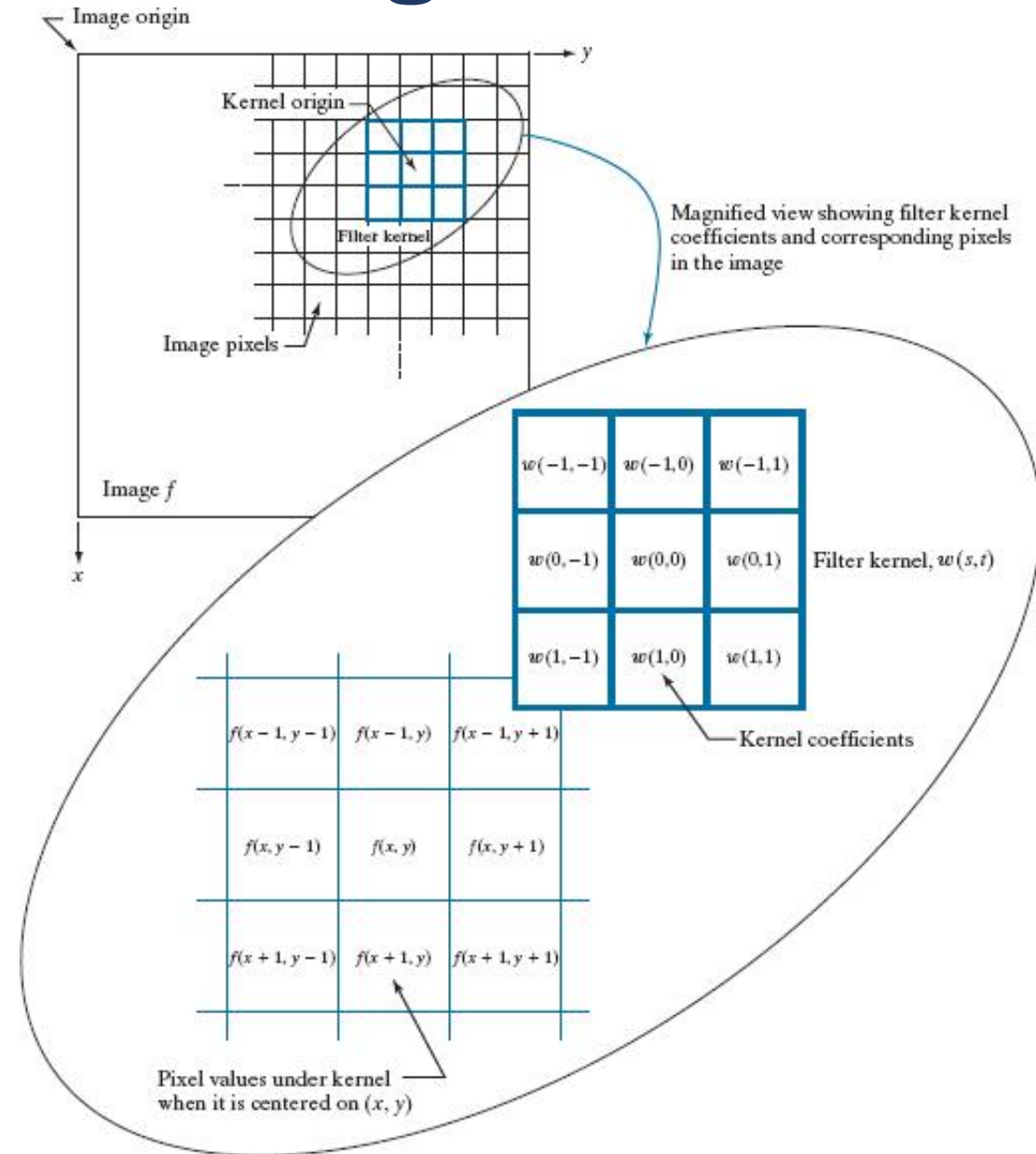
- Zoom class
- Patch-level processing (filtering)
 - Same filter applied to sub-regions/patches

Fundamentals of Spatial Filtering

- Modifying the pixels in an image based on some function of a local neighborhood of the pixels
 - A neighborhood
 - An operator with the same size: linear/nonlinear
 - Each element in the kernel w will visit every pixel in the image just once

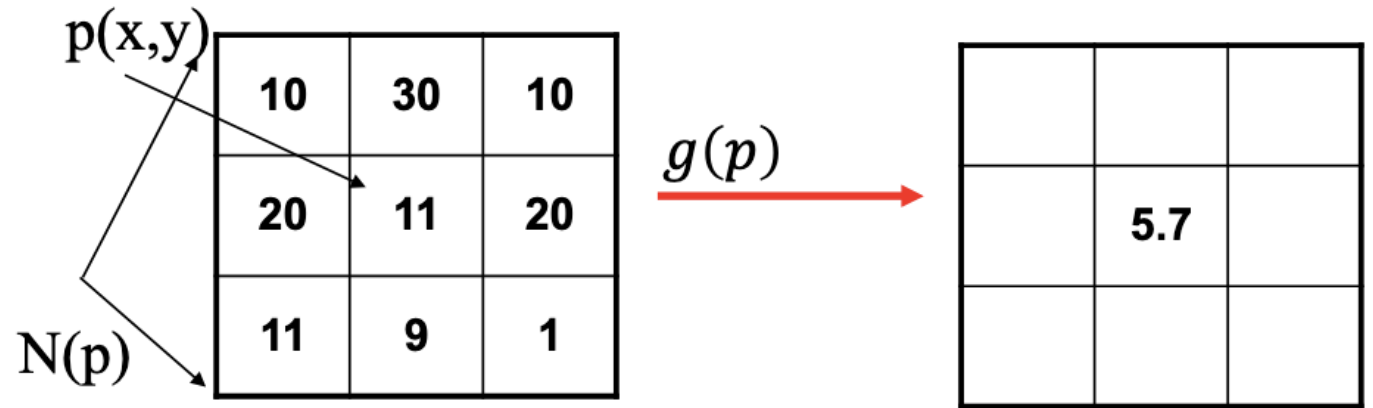
- Example: linear spatial filtering

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



Fundamentals of Spatial Filtering

- $g(p)$
 - Linear function
 - Correlation
 - Convolution
 - Non-linear function
 - Order statistics (e.g., median)



Linear filtering

- Linearity

- $g(\alpha I_1 + \beta I_2) = \alpha g(I_1) + \beta g(I_2)$

Easy to combine things

- Analytic form

If the patch size is 1
(pixel-level function)

$$g(x) = ax + b = (x, 1) \cdot (a, b)$$

If (k+1) pixels in the patch
(patch-level function)

$$\begin{aligned} g(x_0, \dots, x_k) &= \sum_{i=0}^k a_i x_i + b \\ &= (x_0, \dots, x_k, 1) \cdot (a_0, \dots, a_k, b) \end{aligned}$$

Linear filtering

- Implementation with dot product

Pixel-level

```
for i in range(rows):  
    for j in range(cols):  
        I[i,j] = I[i,j] * a + b
```

Patch-level

```
for i in range(rows):  
    for j in range(cols):  
        patch = I[i-half_h:i+half_h+1, \  
                  j-half_w:j+half_w+1]  
        I[i,j] = np.dot(patch.ravel(), A.ravel()) + b
```

Note:

- For 2D matrix, `np.dot(mat1, mat2)` is doing **matrix multiplication**
- Thus, we need to **reshape** input to 1D (`ravel`)

Spatial correlation: 1D signal

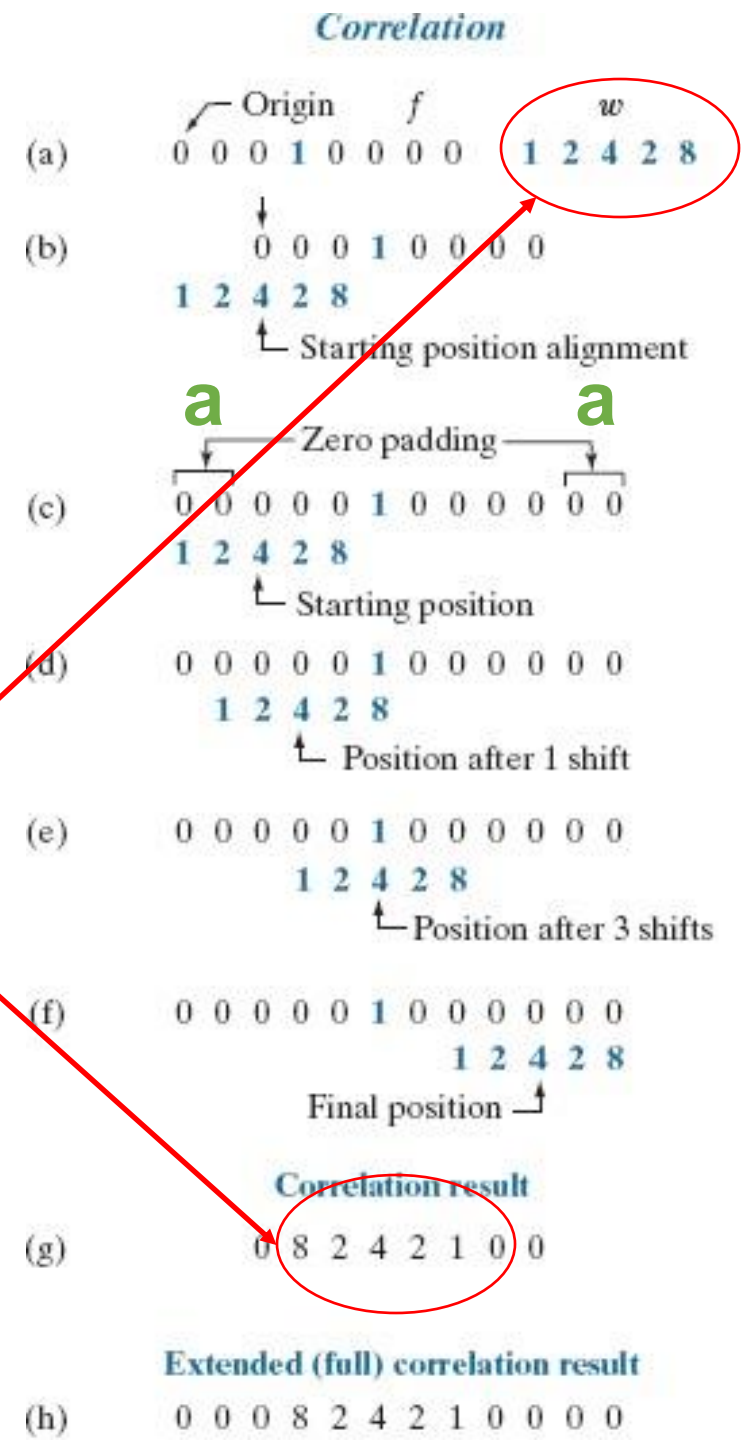
- 1D correlation
$$\sum_{s=-a}^a w(s)f(x \oplus s)$$

Zero-padding: add zeros on the left and right margin, respectively

The impulse response is a **rotation** of the filter by 180 degree

Cropped result has the size of M (i.e., the same as original signal)

Full correlation result has the size of $M + 2a$



Spatial **convolution**: 1D signal

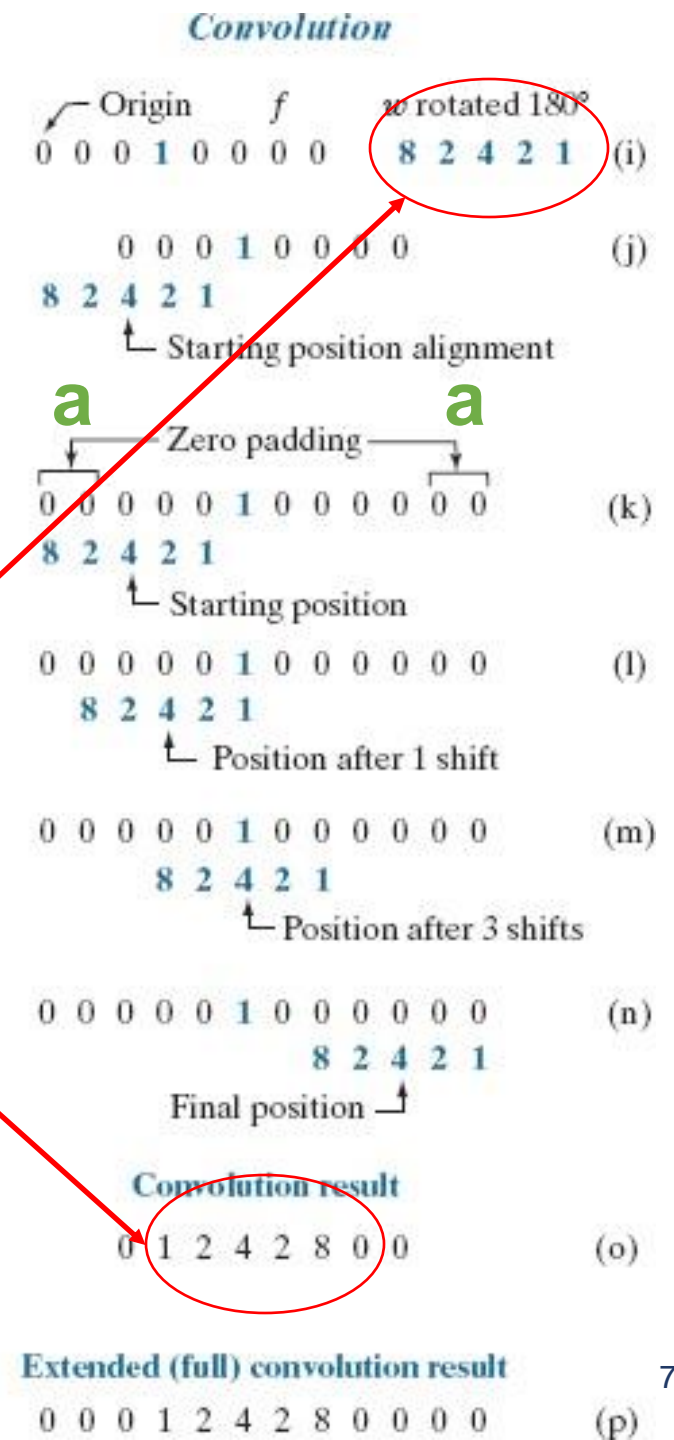
- 1D correlation
$$\sum_{s=-a}^a w(s)f(x - s)$$

Zero-padding: add zeros on the left and right margin, respectively

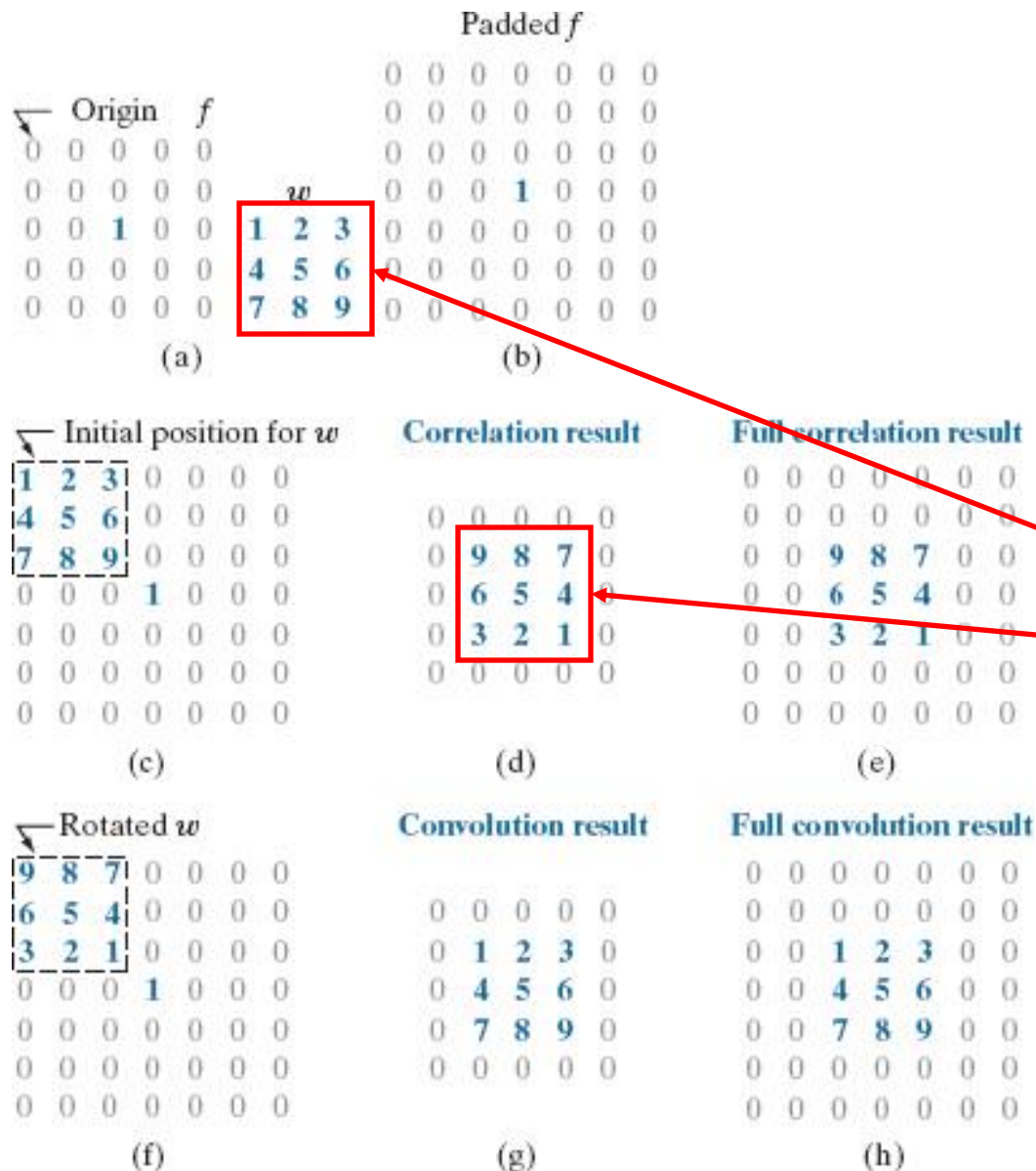
The impulse response is the **same** as the filter

Cropped result has the size of M (i.e., the same as original signal)

Full correlation result has the size of $M + 2a$



Extend to 2D image: 2D image correlation



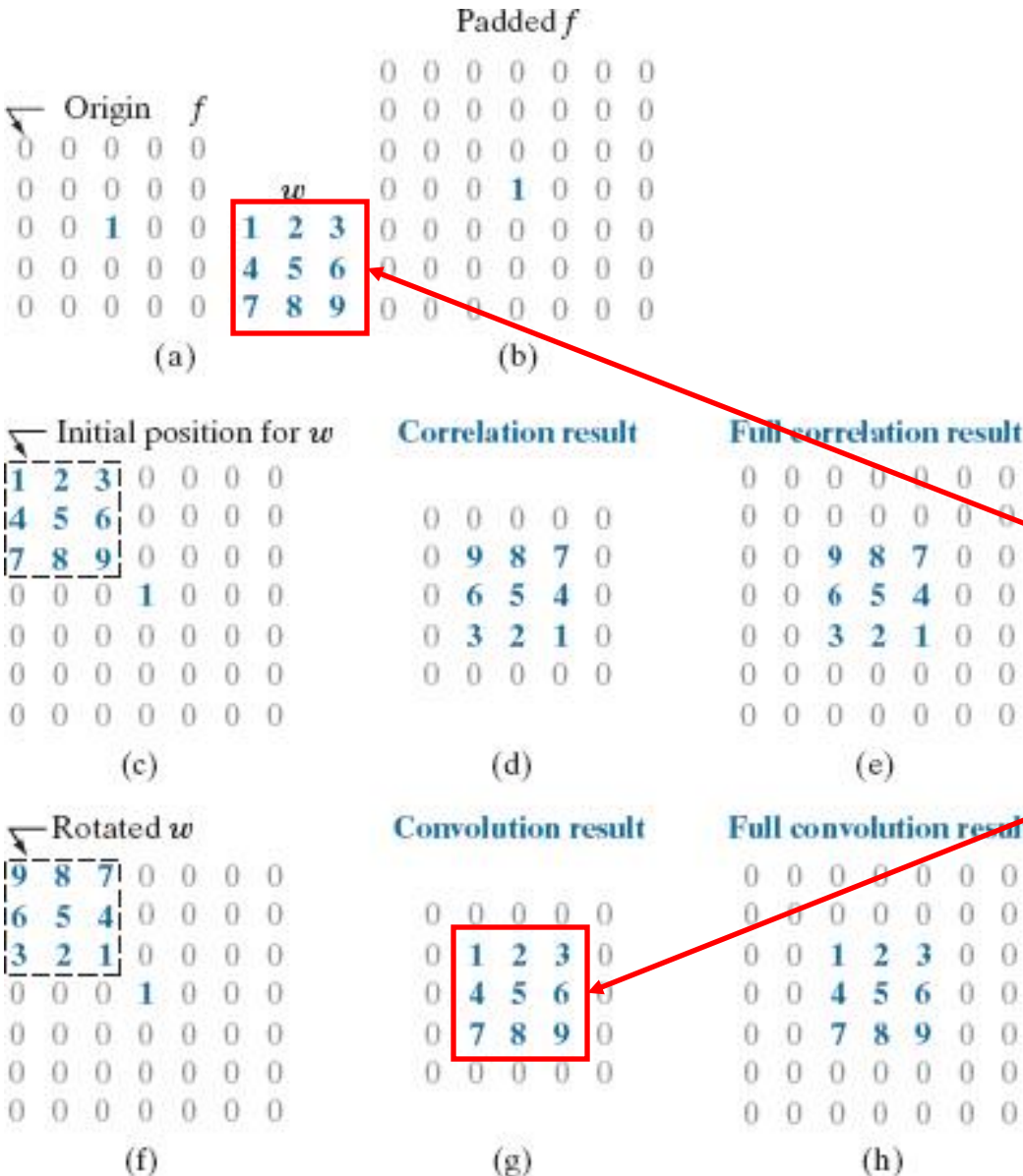
$$\sum_{s=-a}^a w(s) f(x \oplus s)$$



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

The 2D impulse response of image correlation is a rotation of the filter by 180 degree.

Extend to 2D image: 2D image **convolution**



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

↓

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

The 2D impulse response of image convolution is the same as the filter.

Properties of convolution and correlation

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

Let $m = x - s$, then $s = x - m$

$$\sum_{m=-\infty}^{\infty} g(x-m)f(m)$$

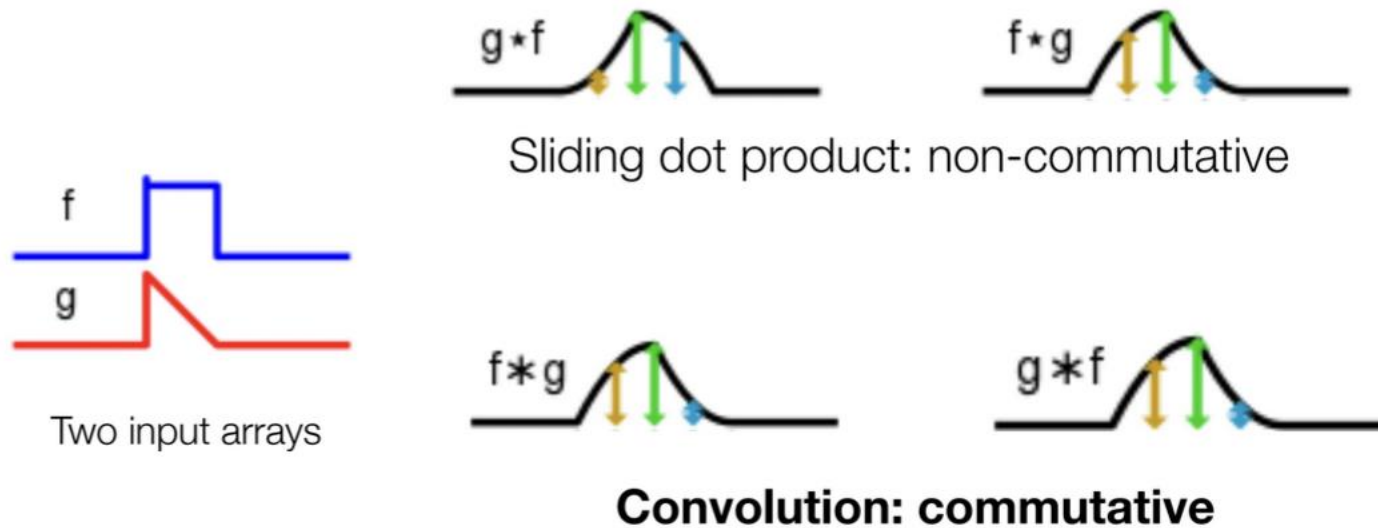
$$\sum_{s=-\infty}^{\infty} g(s)f(x+s)$$



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

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

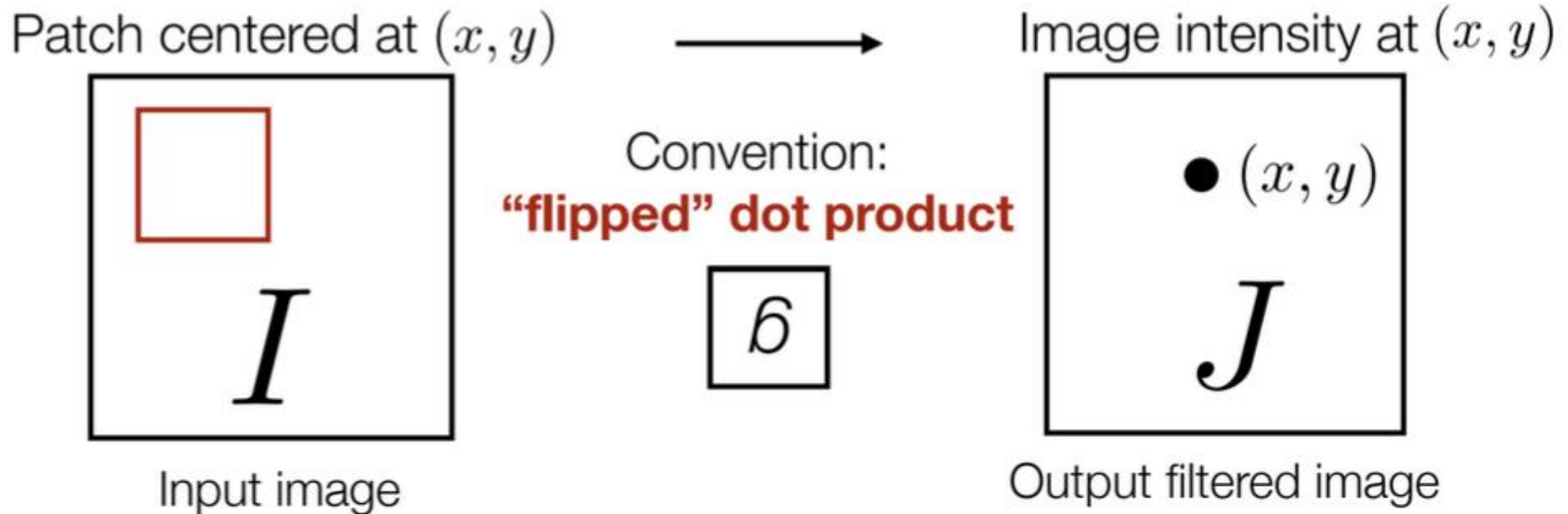
Properties of convolution and correlation



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

Implementation with convolution

- Let I be the input image and g be the filter (convolution kernel)



Convolution by hand

1	2	3
4	5	6
7	8	9

Convolution kernel

$$* \begin{array}{|c|c|c|} \hline 1 & 0 & -1 \\ \hline \end{array}$$

g

= ???

Convolution by hand

Step 1: pad the image

1	1	2	3	3
4	4	5	6	6
7	7	8	9	9

Step 2: flip the kernel

Convolution kernel

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

g

Step 3: sliding dot product

1	2	1
1	2	1
1	2	1

Flipped kernel

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

Box filter

$I[x, y]$: Image intensity at pixel (x,y)

$I[0, 0]$	$I[0, 1]$	$I[0, 2]$
$I[1, 0]$	$I[1, 1]$	$I[1, 2]$
$I[2, 0]$	$I[2, 1]$	$I[2, 2]$

Input

Filter g
Average function

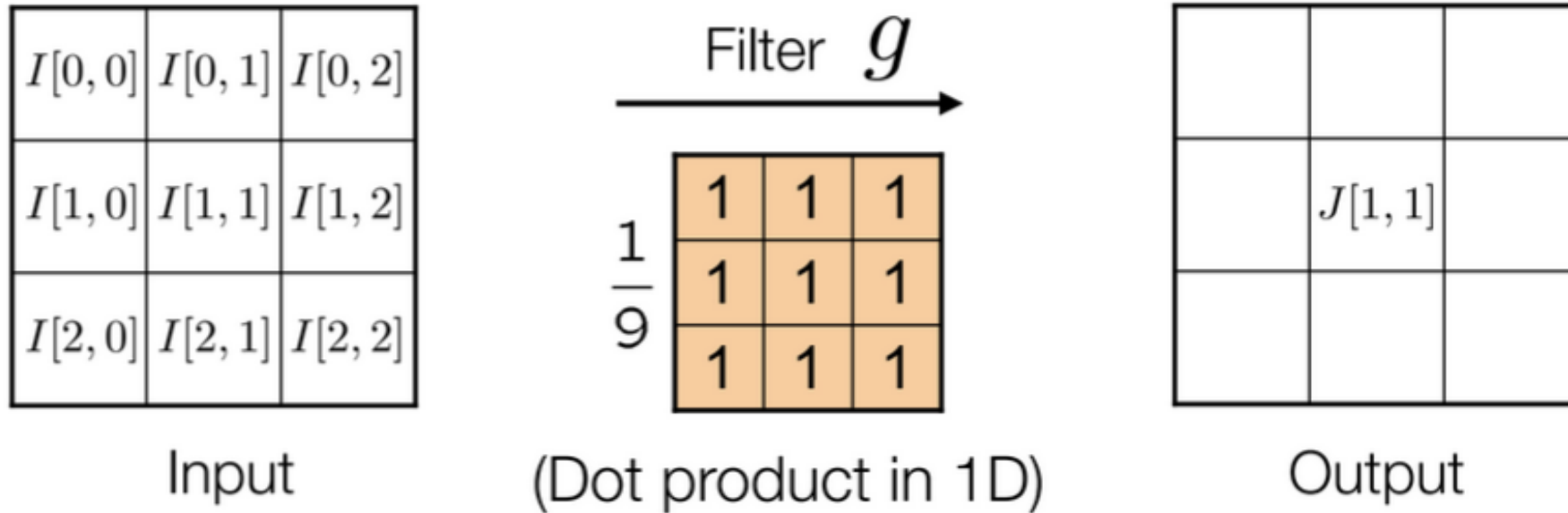
	$J[1, 1]$	

Output

$$J[1, 1] = \frac{1}{N} \sum_{x,y} I[x, y]$$

Box filter

$I[x, y]$: Image intensity at pixel (x,y)



$$J[1, 1] = \frac{1}{N} \sum_{x,y} I[x, y] = I \cdot \frac{1}{N} (1, \dots, 1)$$

Box filter: denoising



Input Image
(Gaussian noise)



Output: Filtered image
(noise averaged out)

Box filter variants: Impulse filter

Box size = [1,1]

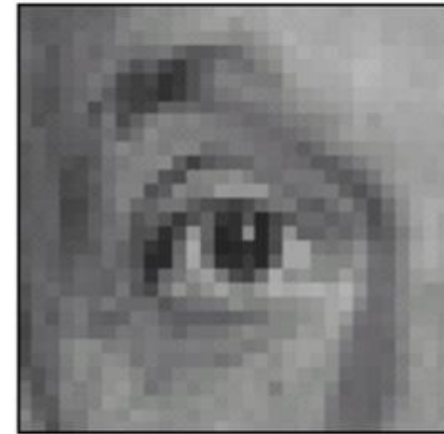


Original

*

0	0	0
0	1	0
0	0	0

=



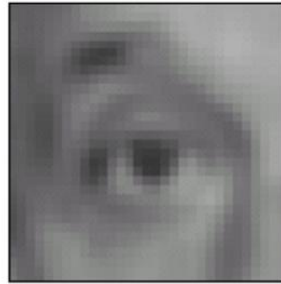
Filtered
(no change)

Box filter variants: Sharpen filter



Original

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

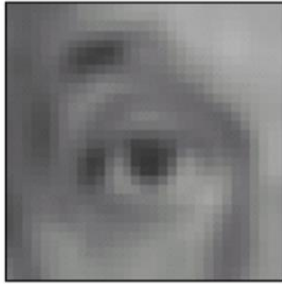


Blur (with a box filter)



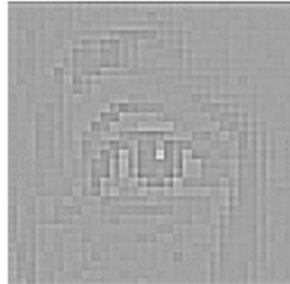
Original

-



Blur (with a box filter)

=

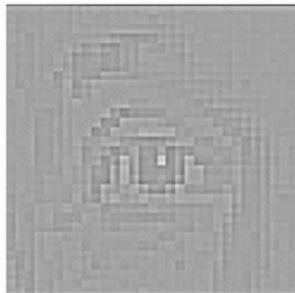


Details



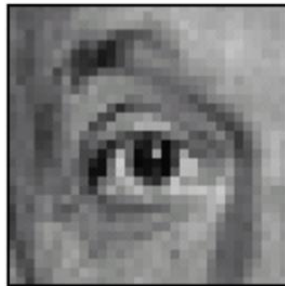
Original

+

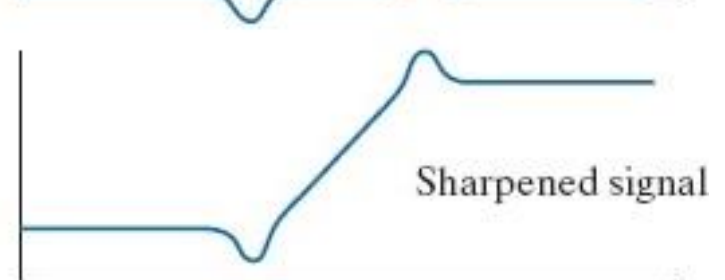
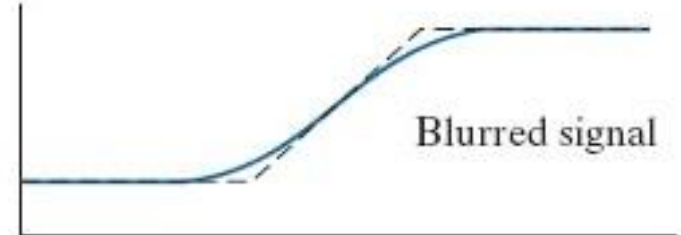


Details


=

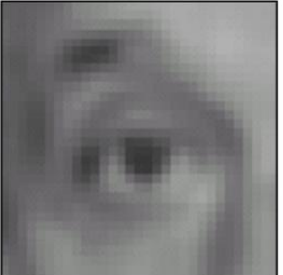


Sharpened




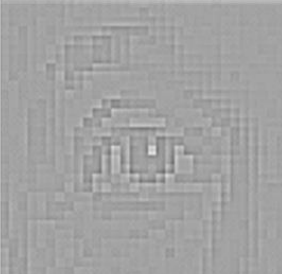
Box filter variants: Sharpen filter




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



Original Blur (with a box filter)




$$- \text{Blur (with a box filter)} =$$


Original Blur (with a box filter) Details



$$+ \text{Details} =$$


Original Details **Sharpened**





$$* \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} +$$


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

Original Details

Convolution filters are linear!



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


Original Sharpened

Sharpen filter $\frac{1}{9} \begin{bmatrix} -1 & -1 & -1 \\ -1 & 7 & -1 \\ -1 & -1 & -1 \end{bmatrix}$

Gaussian filter

$$h(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

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

1	1	1
1	1	1
1	1	1

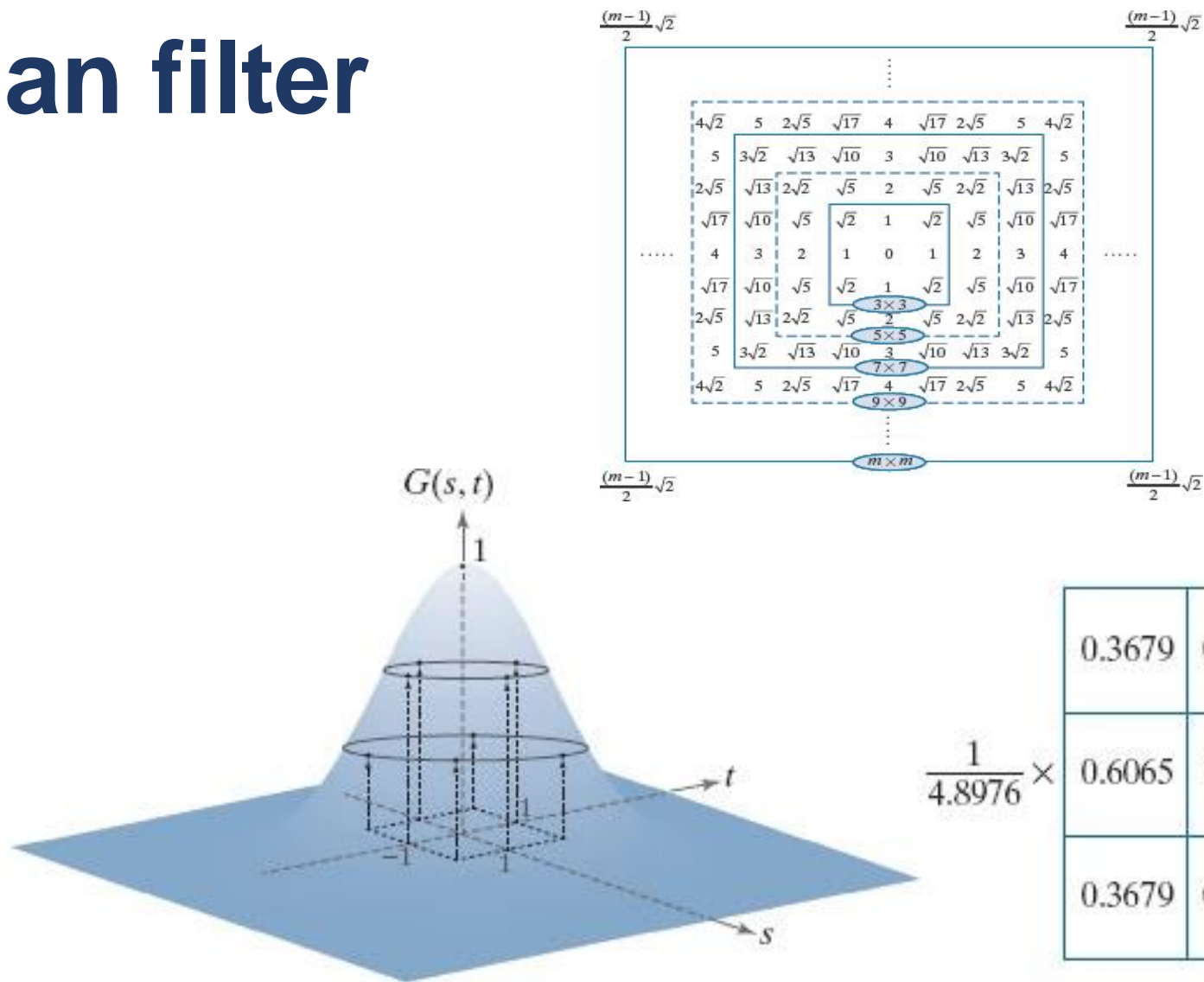
Box filter

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

0.3679	0.6065	0.3679
0.6065	1.0000	0.6065
0.3679	0.6065	0.3679

Gaussian filter

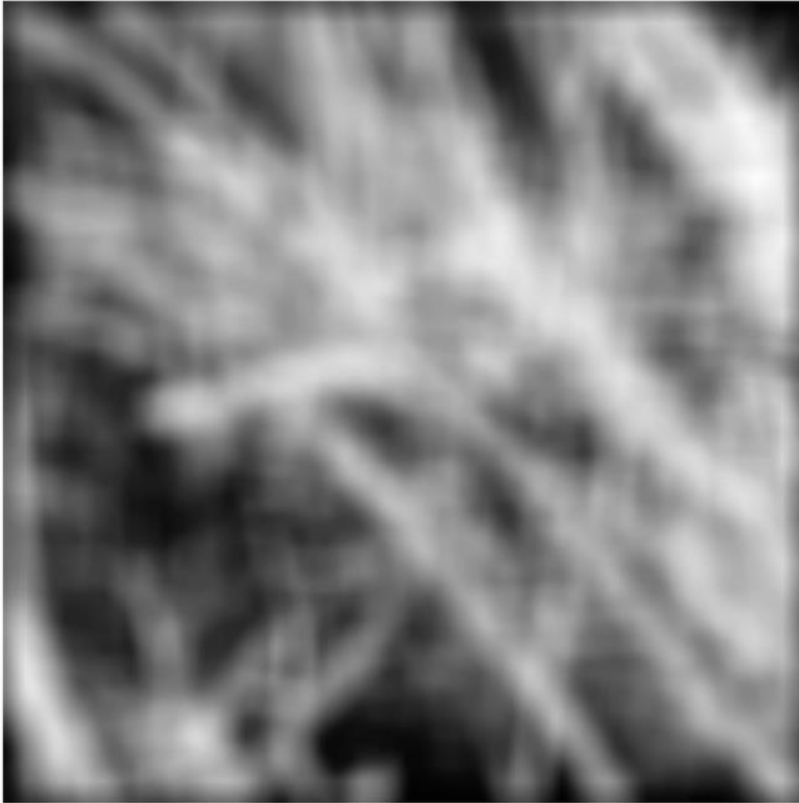
Gaussian filter



Sampling a Gaussian function to obtain a discrete Gaussian kernel. The values shown are for $k = 1, \sigma = 1$

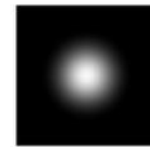
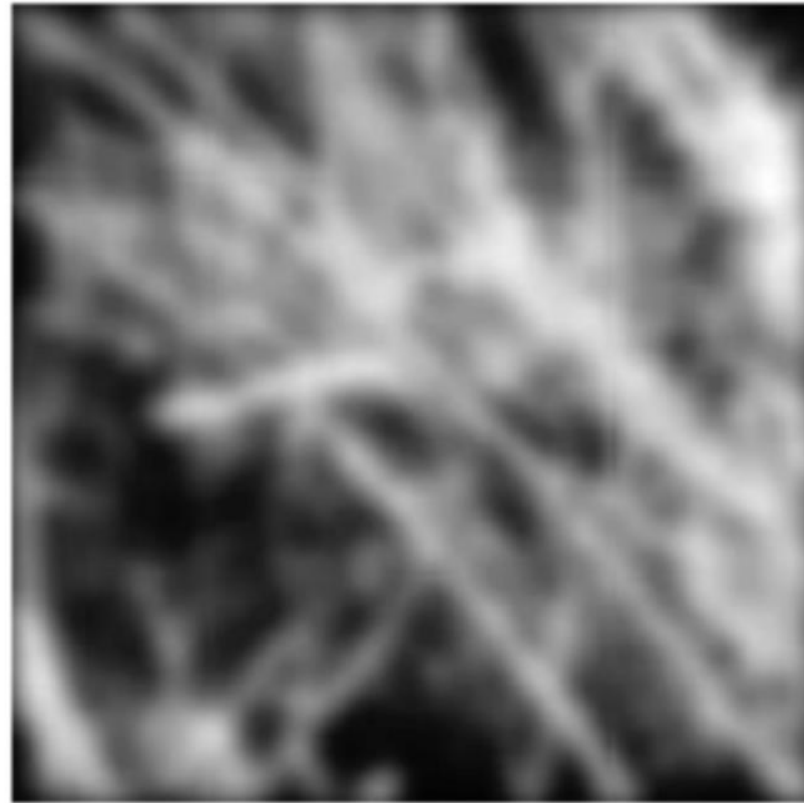
Resulting 3×3 kernel

Result comparison



Box filter

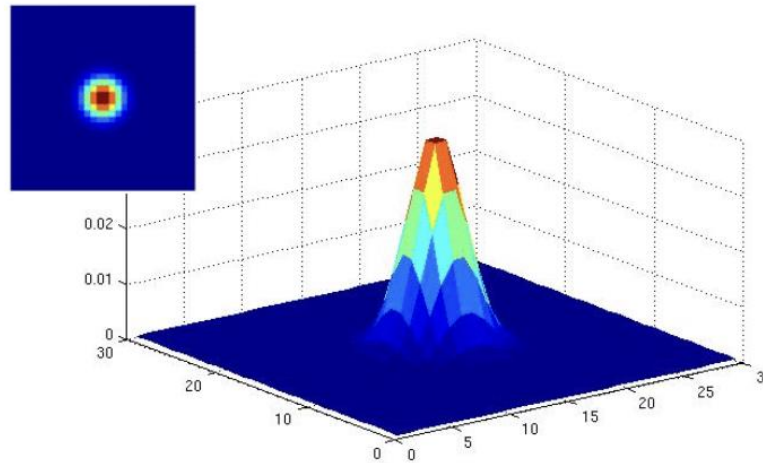
Ghosting grid artifact



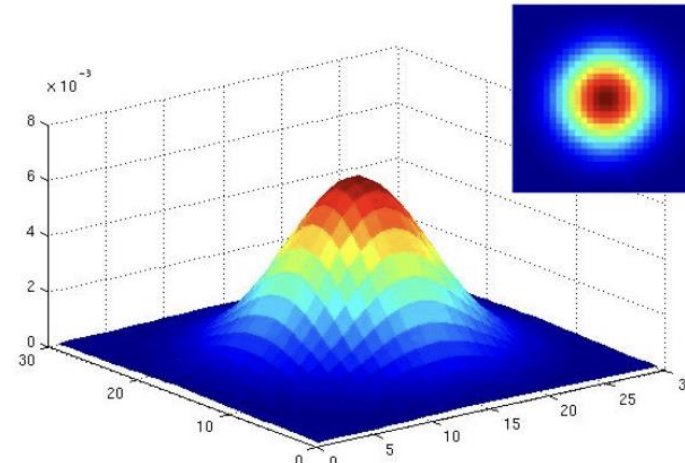
“fuzzy blob”

Gaussian filter parameter: standard deviation

$$G_{\sigma} = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$



kernel: $\sigma = 2$ with 30 x 30

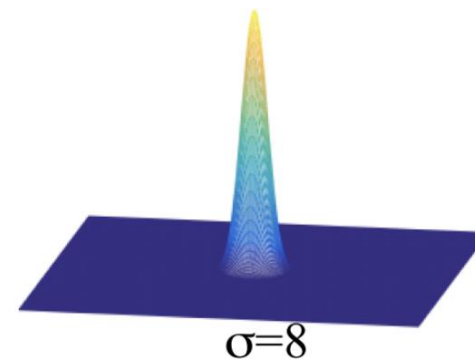
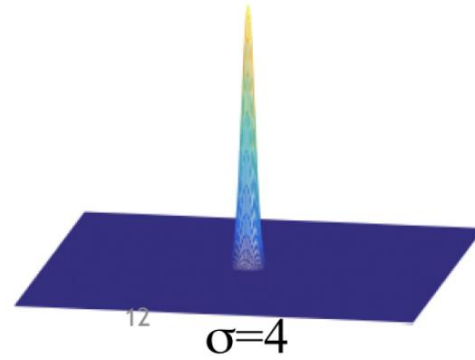
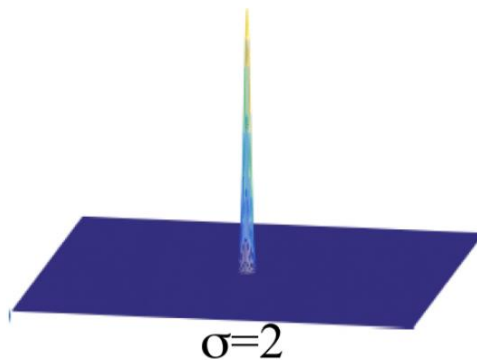


kernel: $\sigma = 5$ with 30 x 30

Standard deviation σ : determines extent of smoothing

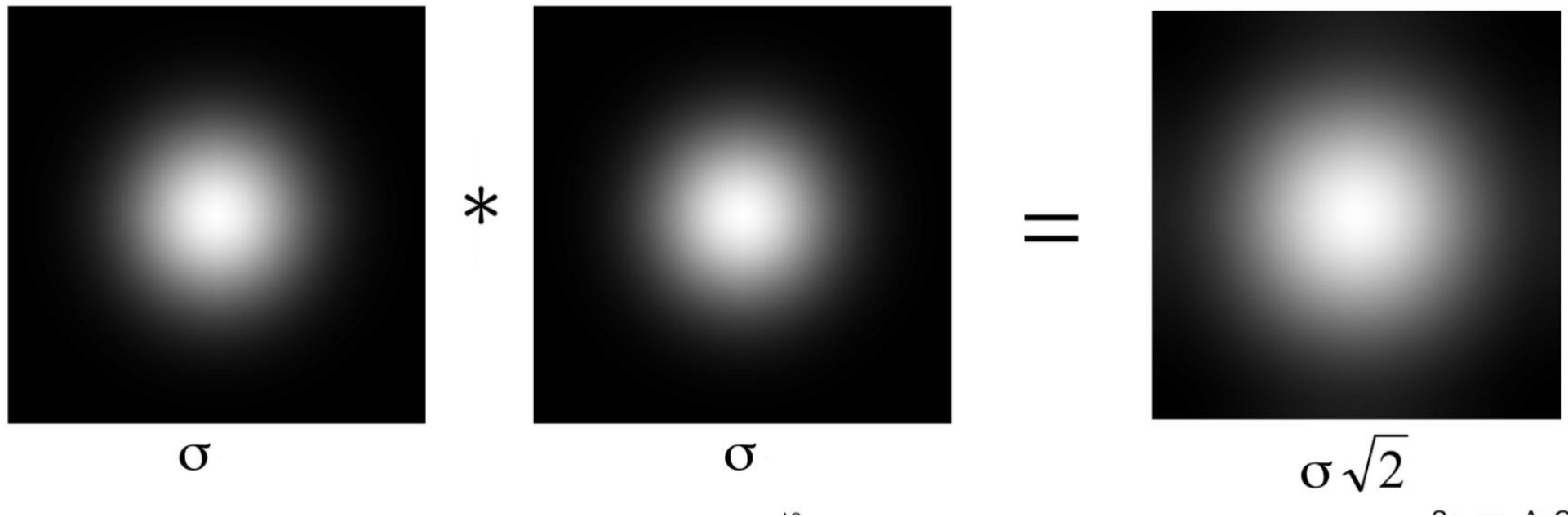
Gaussian filter parameter: standard deviation

- Bigger σ
 - More blurry
 - Bigger effective kernel size



Gaussian filter property: Recursive

- Convolve a Gaussian filter with itself?



The diagram illustrates the recursive property of Gaussian filters. It shows two identical Gaussian filters, each with a standard deviation of σ , being convolved (indicated by the asterisk $*$) to produce a single Gaussian filter with a standard deviation of $\sigma\sqrt{2}$ (indicated by the equals sign $=$).

$$\sigma * \sigma = \sigma\sqrt{2}$$

Gaussian filter property: separability

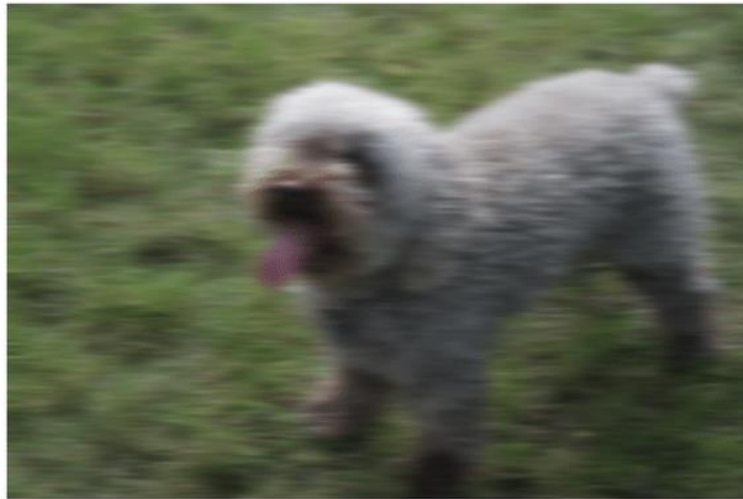
- It is a separable kernel
- Blur with 1D Gaussian in one direction, then the other
- Fast! For an $n \times n$ kernel, $O(n)$ instead of $O(n^2)$

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

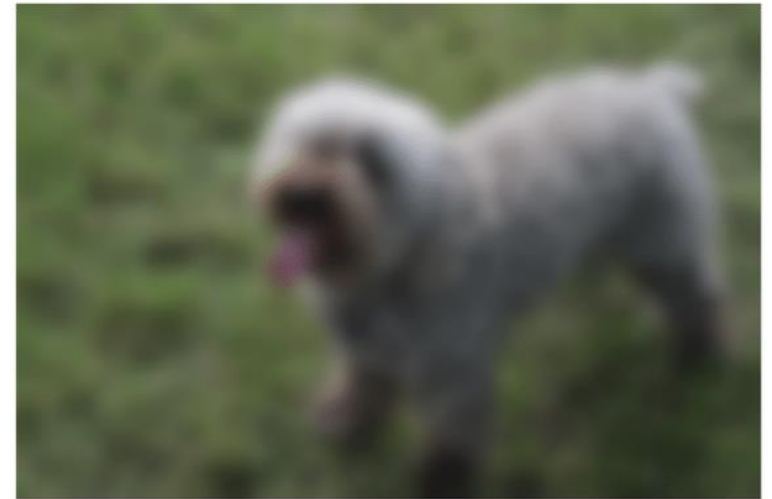
Separable kernel



I



$\text{blur}_x(I)$



$\text{blur}_y(\text{blur}_x(I))$

