Announcement

1/30/2025 Thursday

- Zoom class

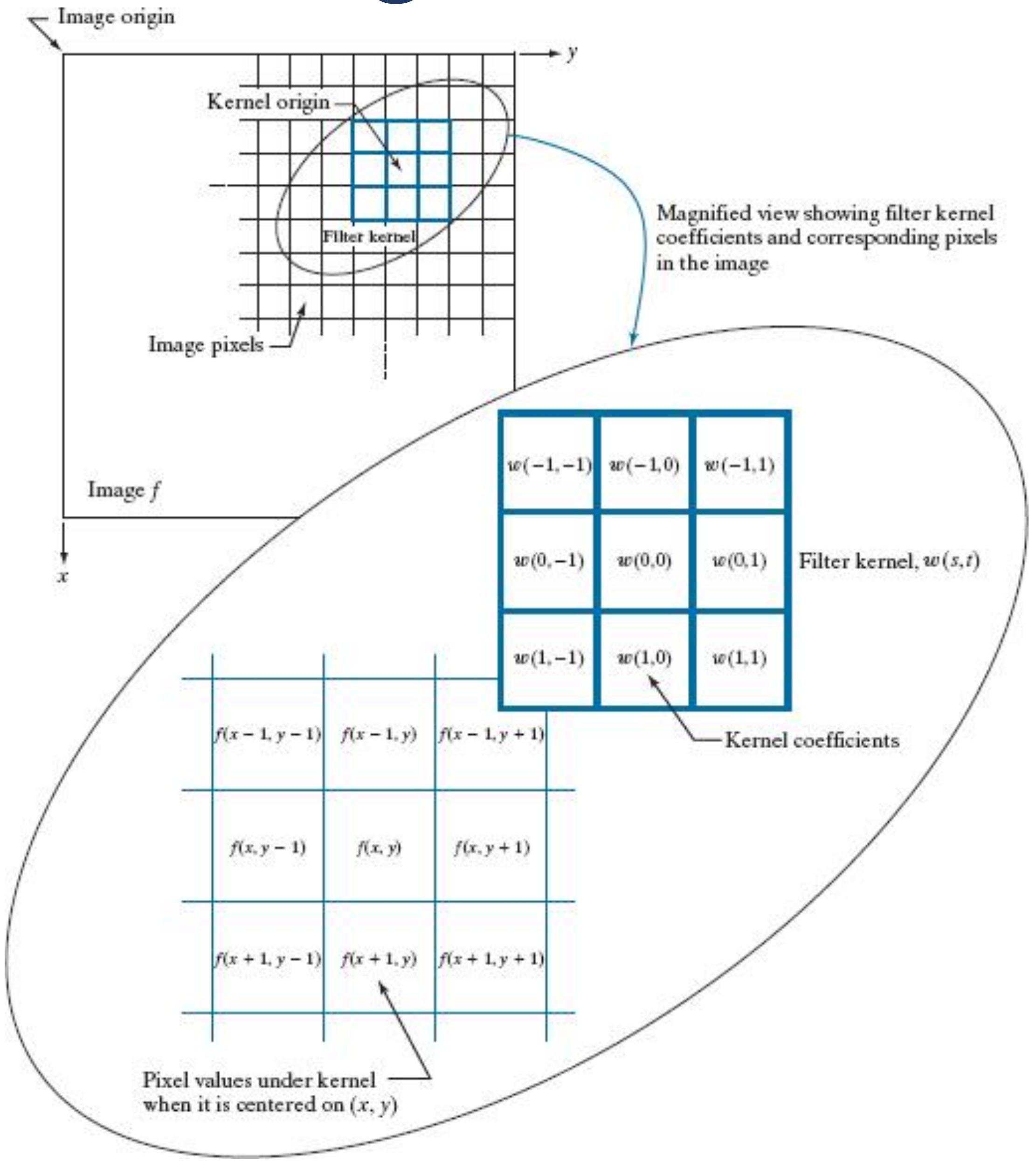
- Patch-level processing (filtering)
  - Same filter applied to sub-regions/patches

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

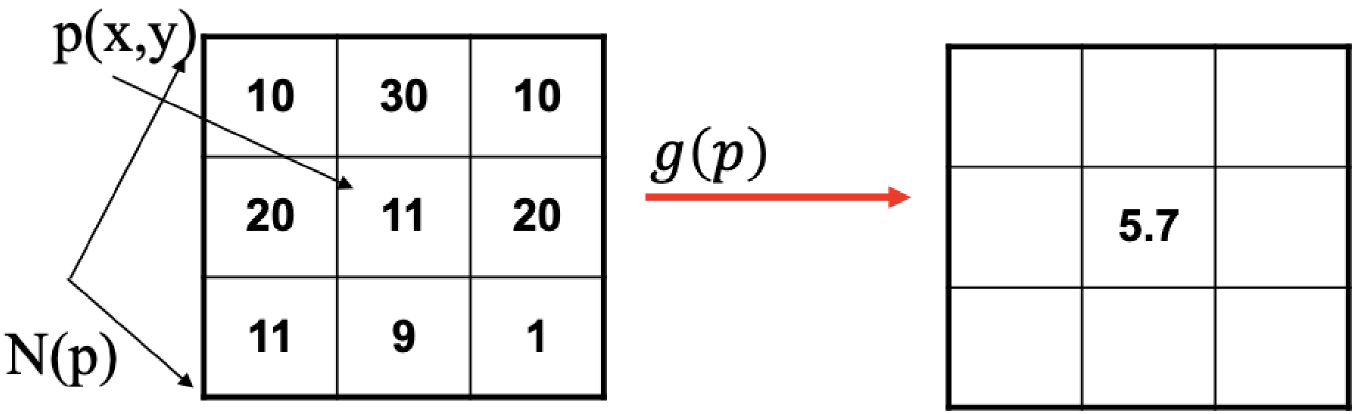
- Example: linear spatial filtering

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



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)

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

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(rows):
        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

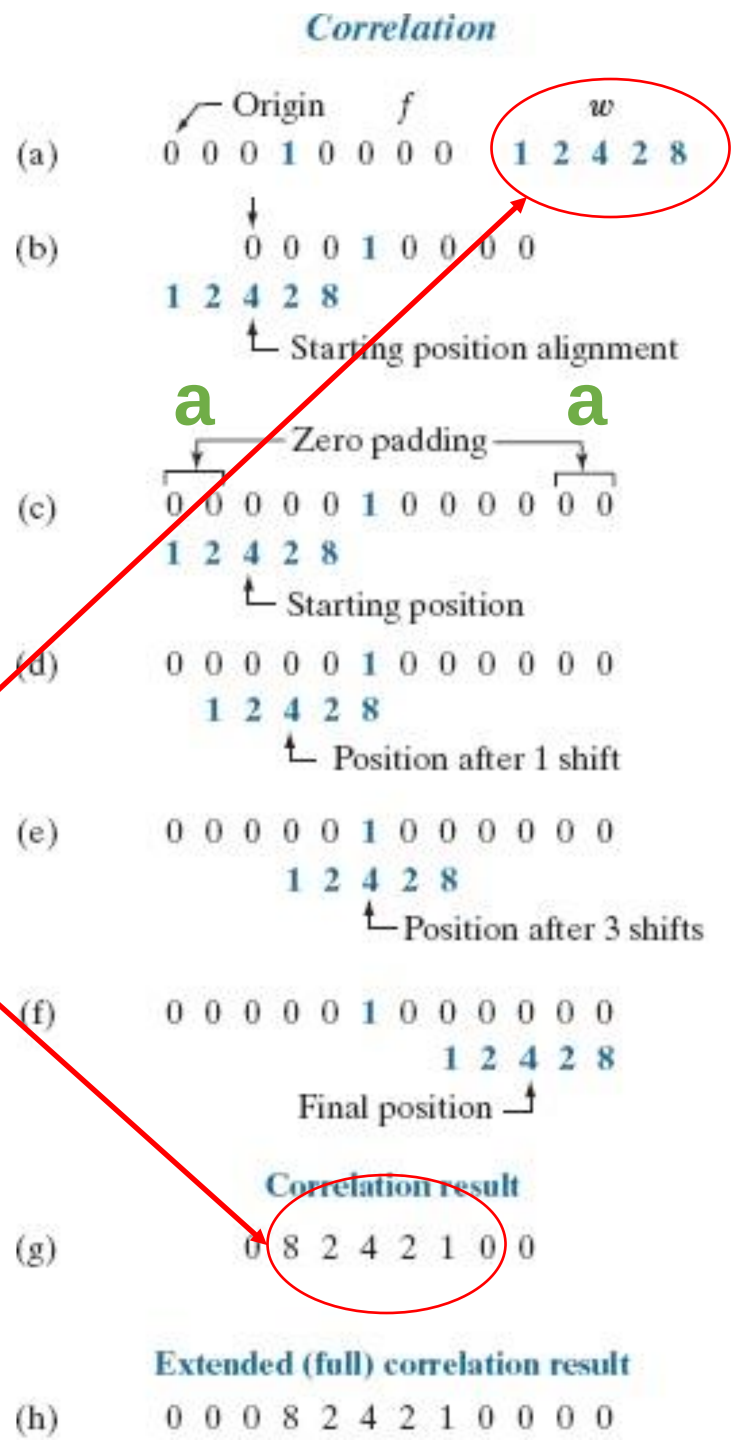
- 1D correlation $\sum_{s=-a}^{a} w(s)f(x+s)$

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

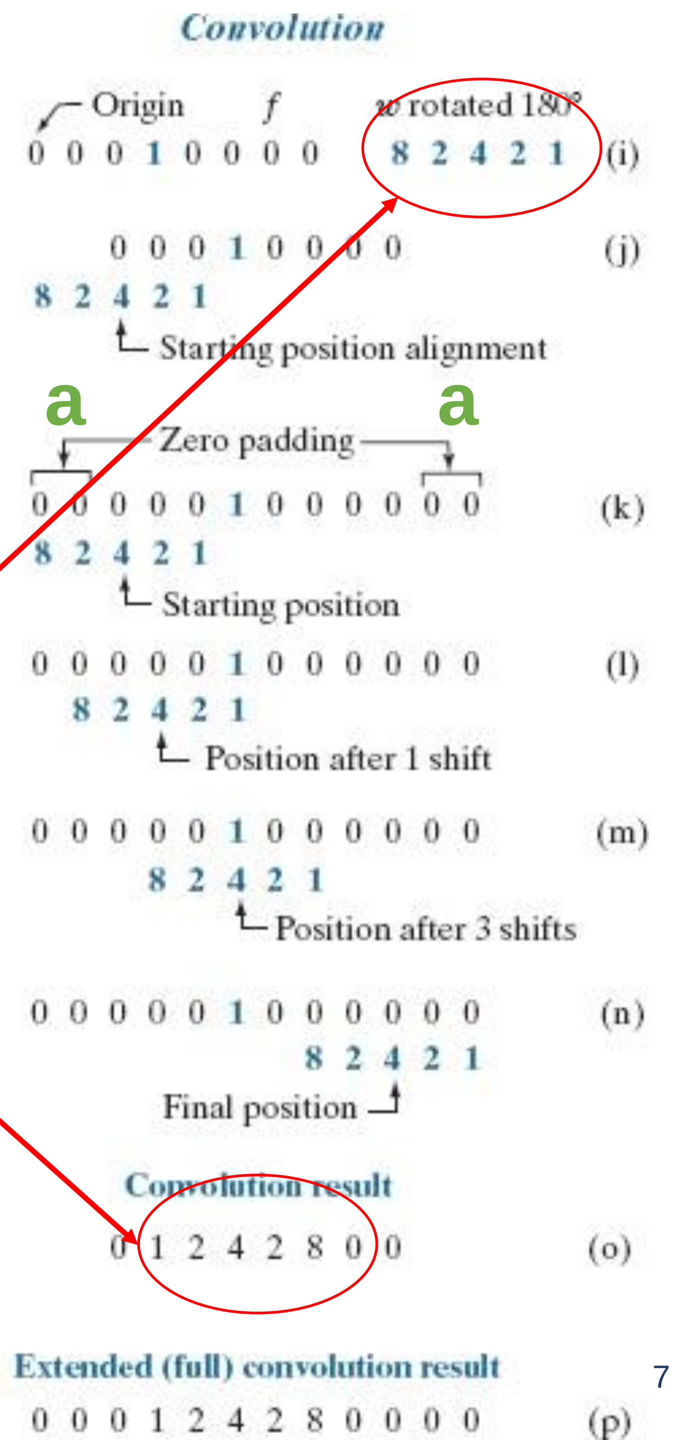
- 1D correlation $\sum_{s=-a}^{a} w(s) f(x - s)$

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

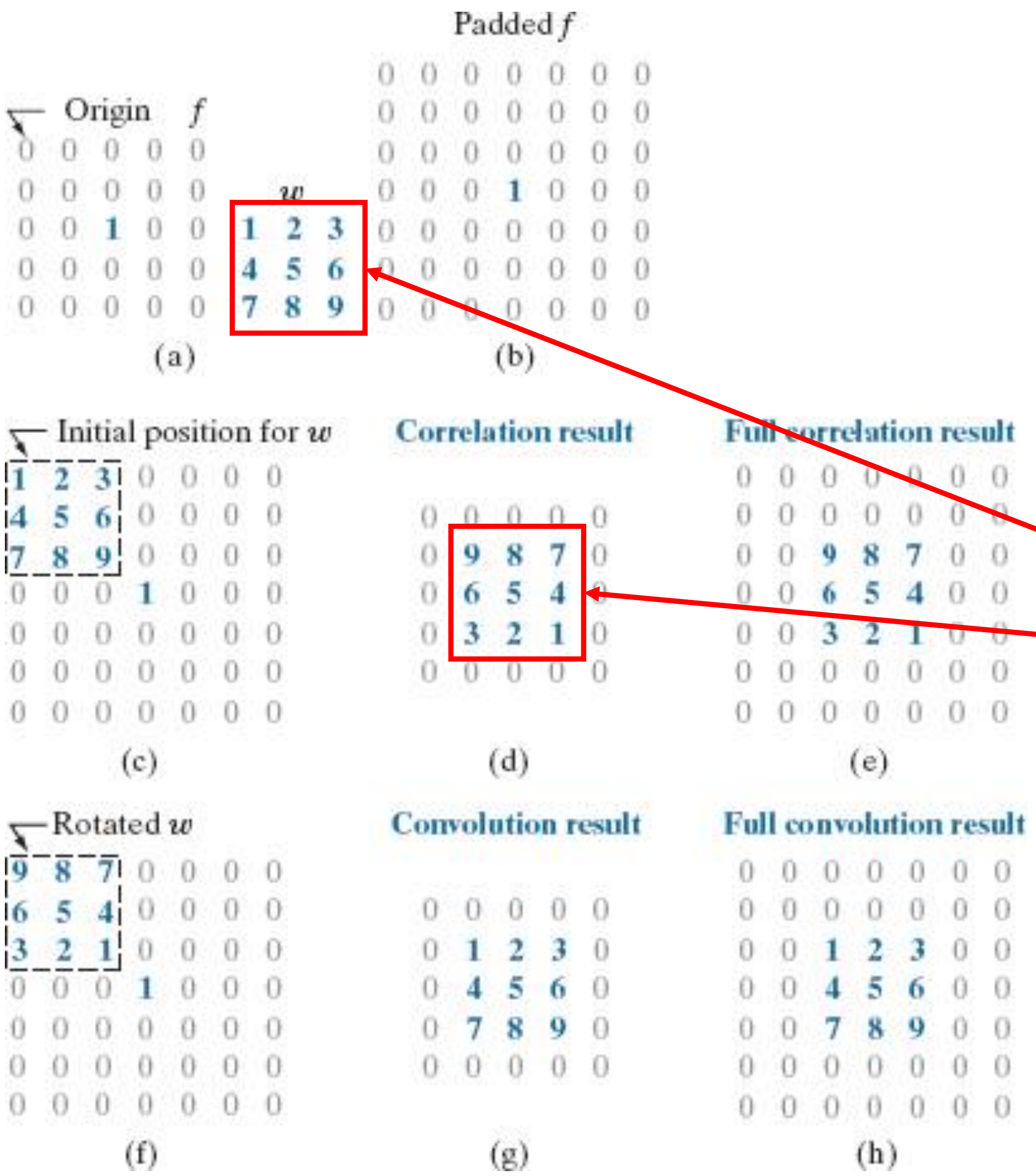
The impulse response is the same as the filter

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

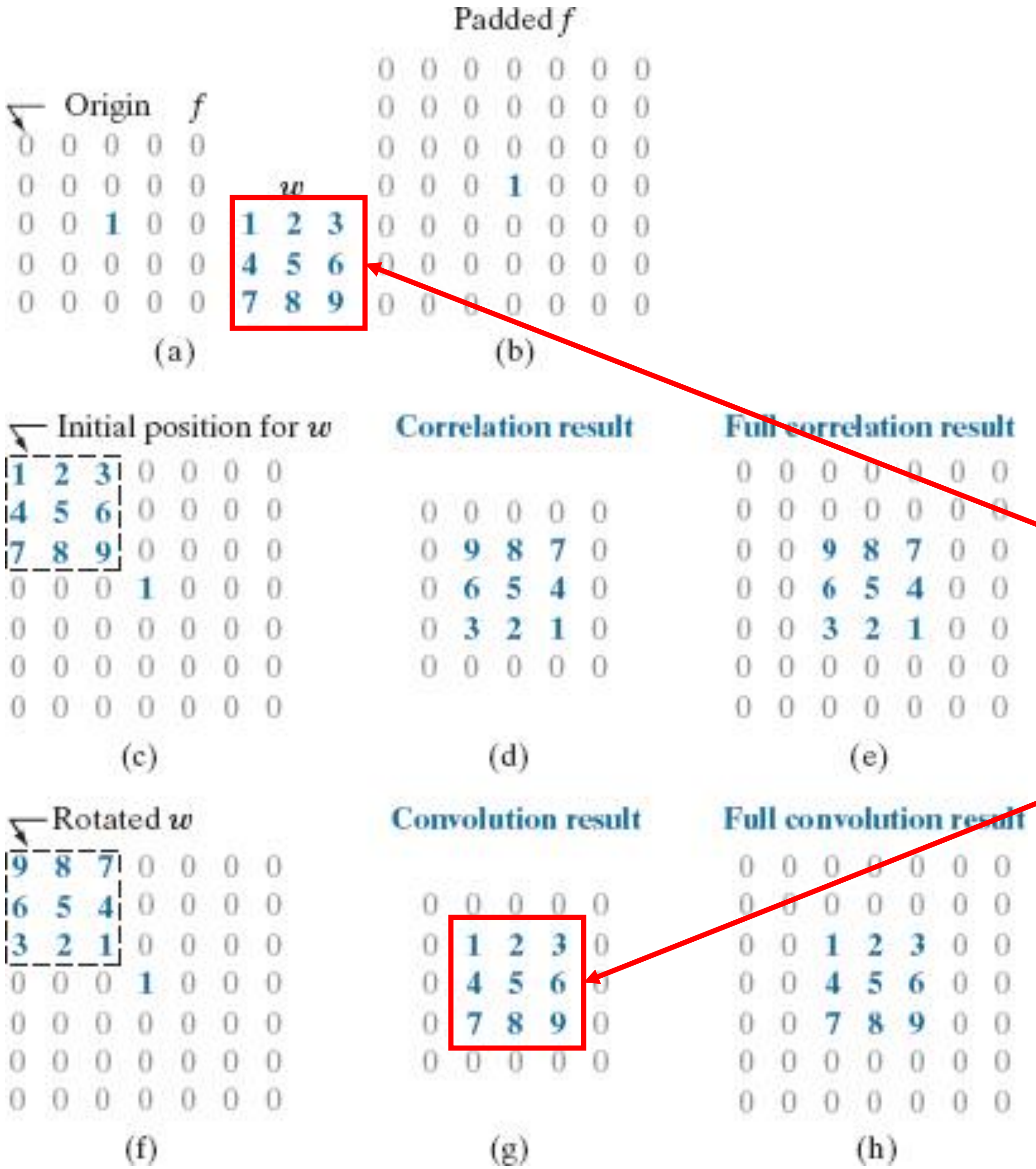


$$\sum_{s=-a}^{a} w(s) f(x + s)$$

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

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

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

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

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

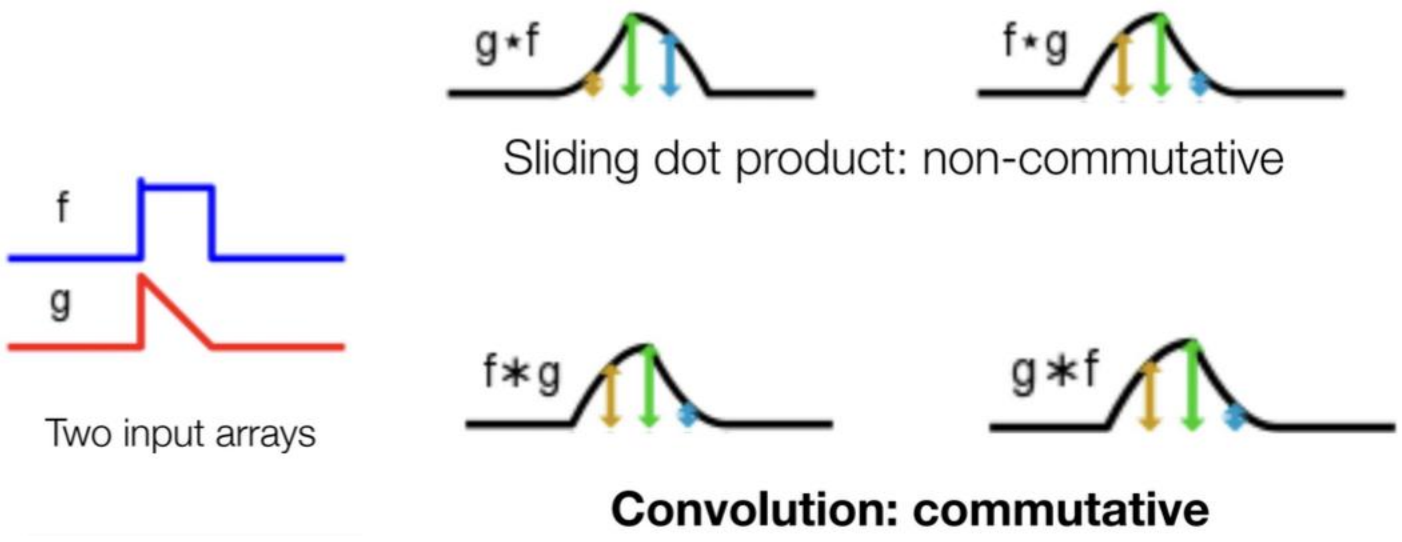
?

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

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

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

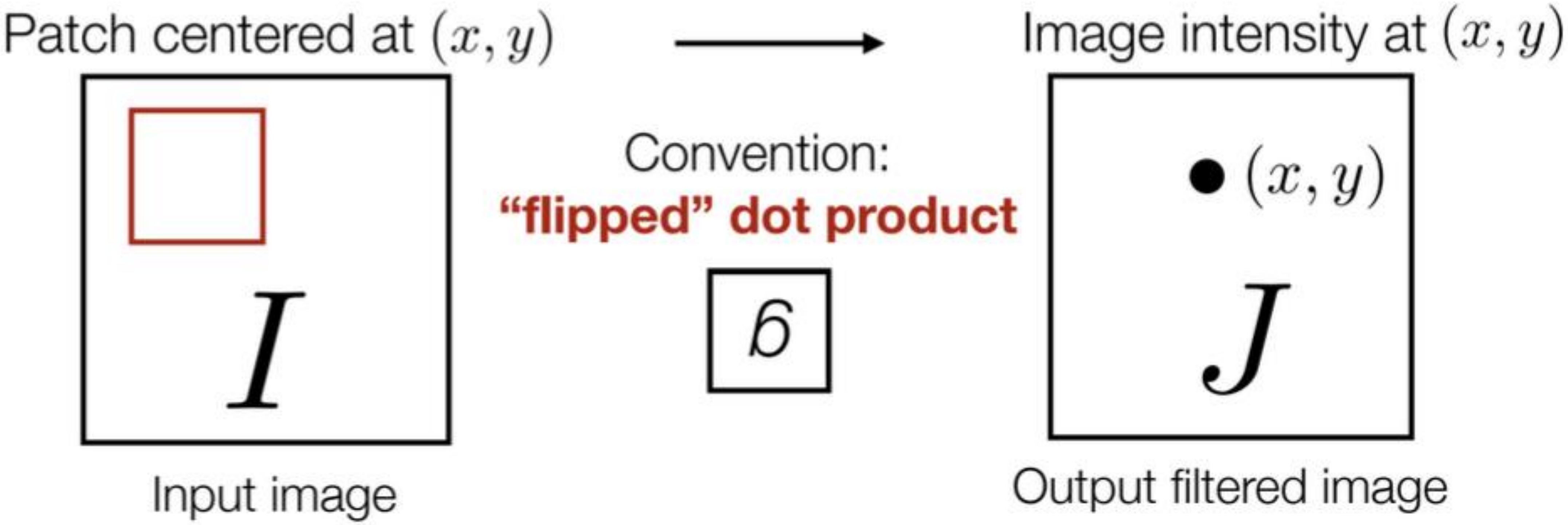
Properties of convolution and correlation



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

Implementation with convolution

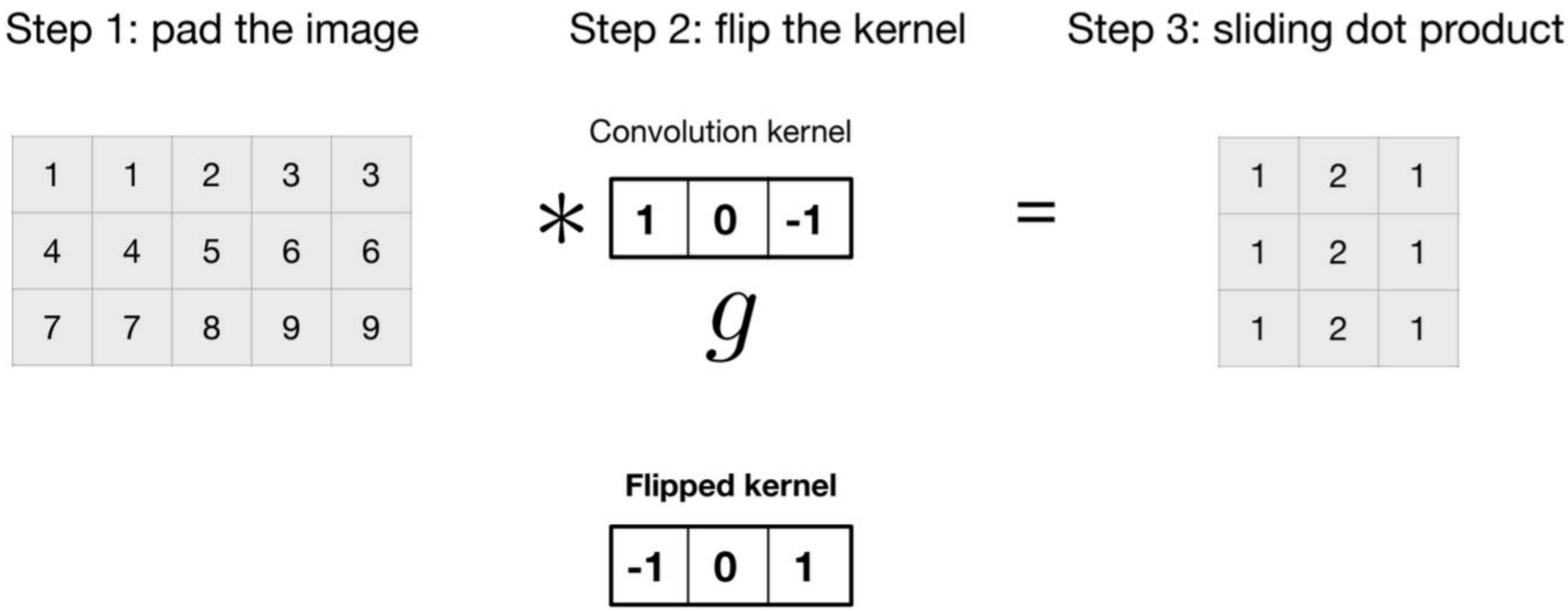
- Let I be the input image and g be the filter (convolution kernel)



Convolution by hand

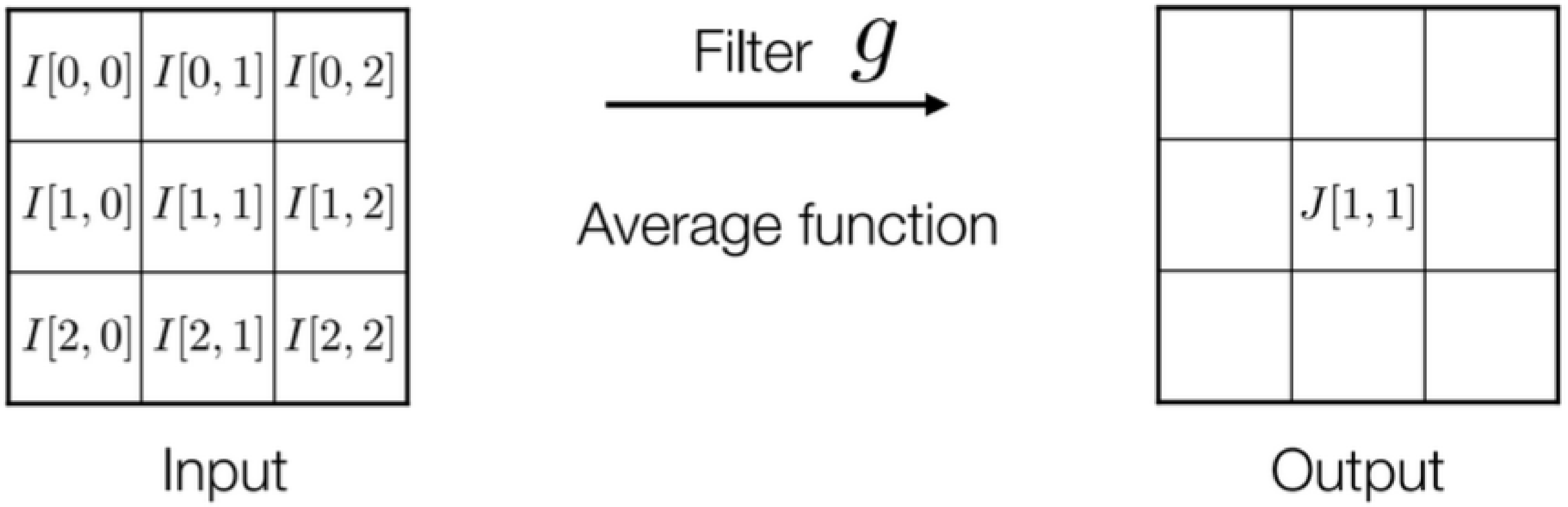
$$\begin{array}{|c|c|c|}\hline 1 & 2 & 3 \\ \hline 4 & 5 & 6 \\ \hline 7 & 8 & 9 \\ \hline \end{array} * \underset{g}{\overset{\text{Convolution kernel}}{\begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline \end{array}}} = ???$$

Convolution by hand



Box filter

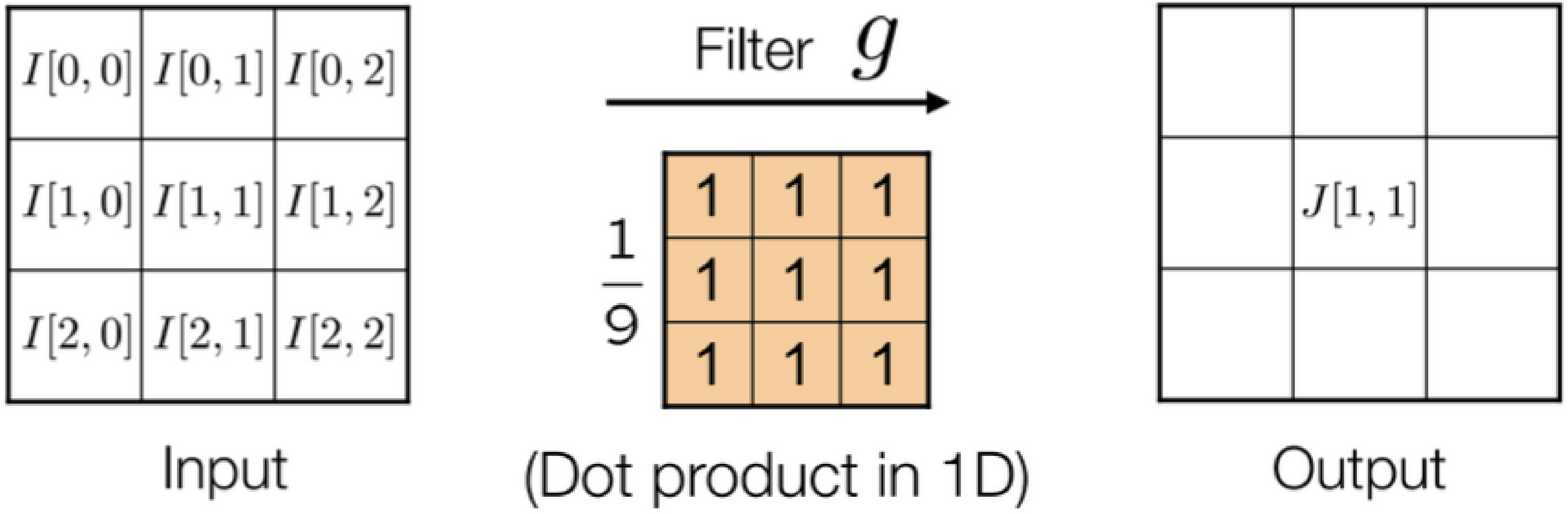
$I[x, y]$: Image intensity at pixel (x,y)



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

Box filter

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



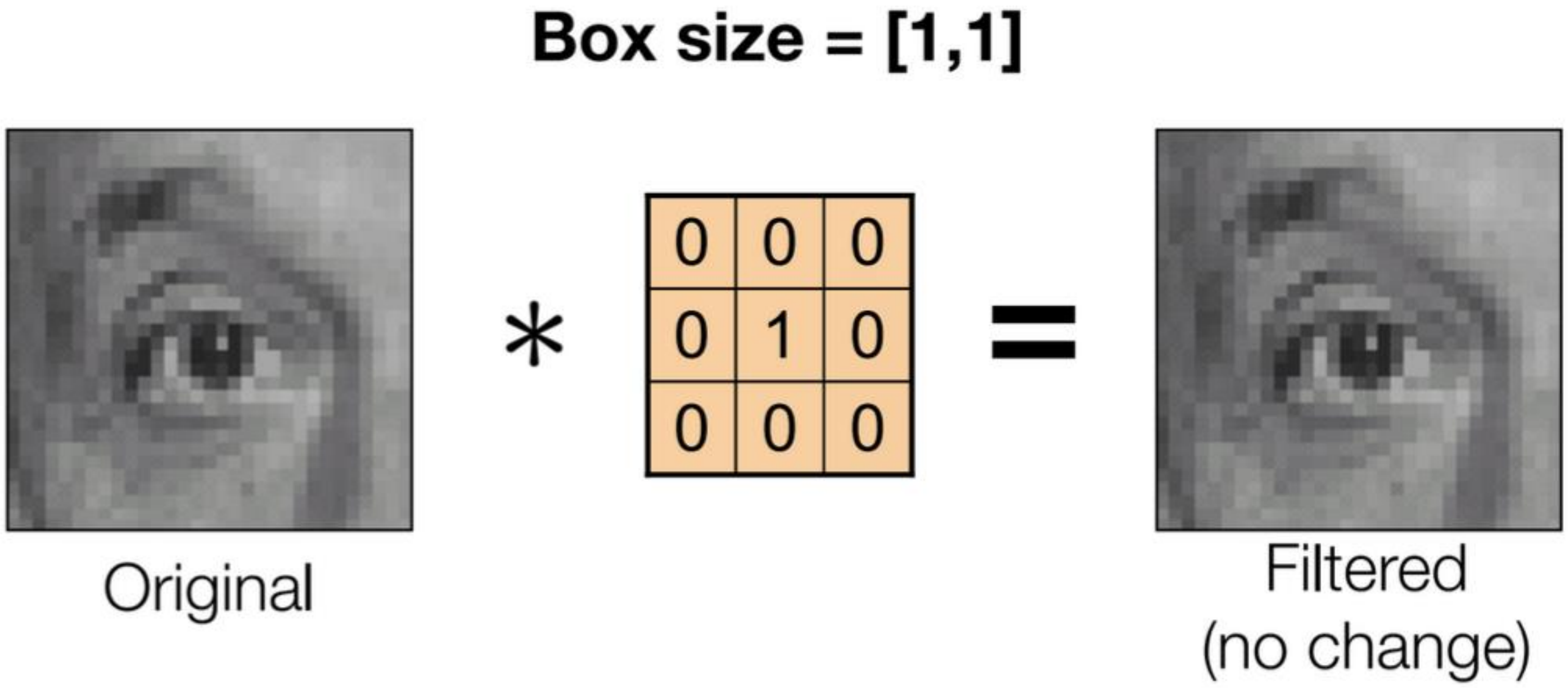
$$J[1,1] = \frac{1}{N} \sum_{x,y} I[x,y] = I \cdot \frac{1}{N}(1, \ldots, 1)$$

Box filter: denoising

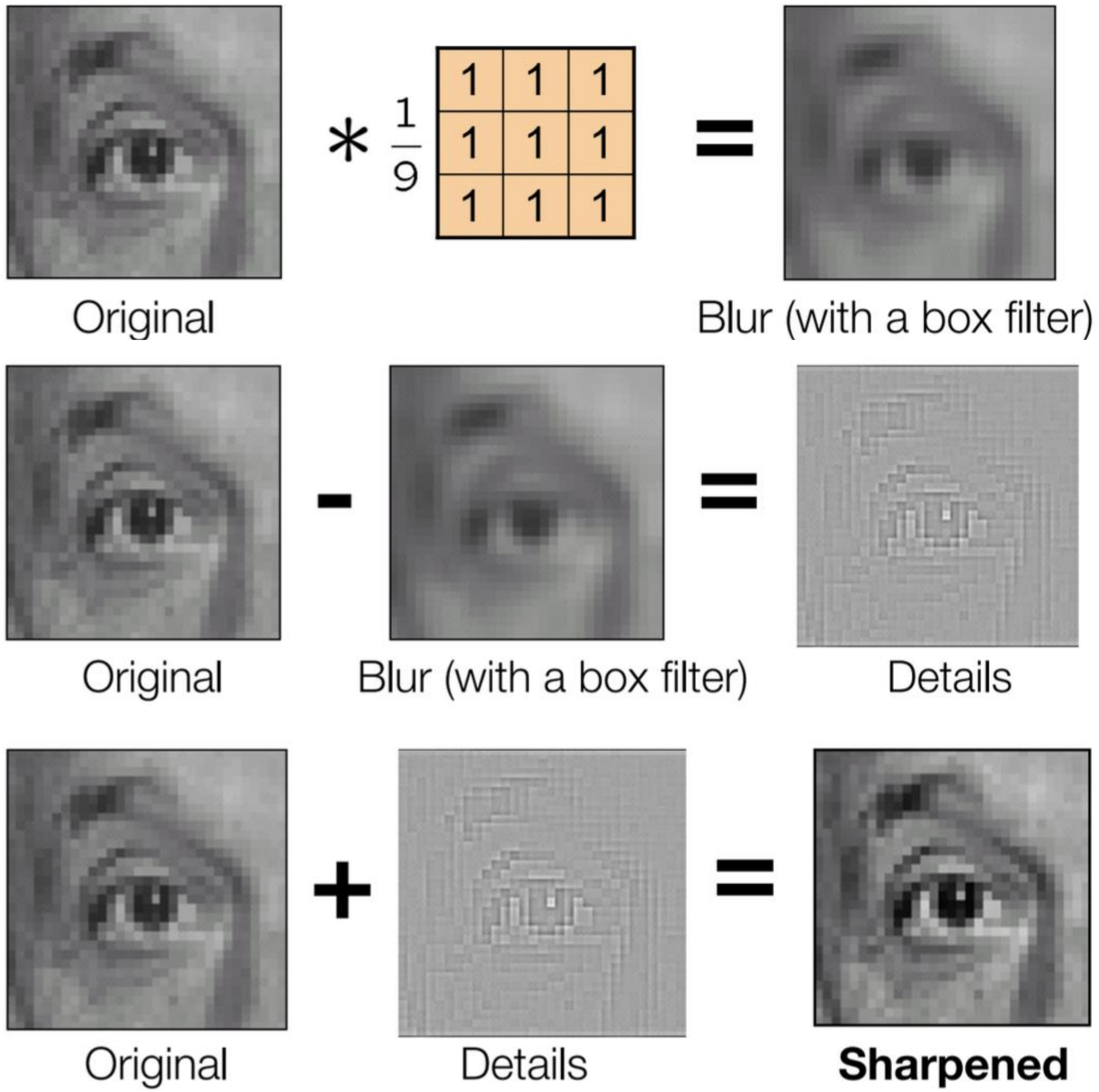
Input Image
(Gaussian noise)

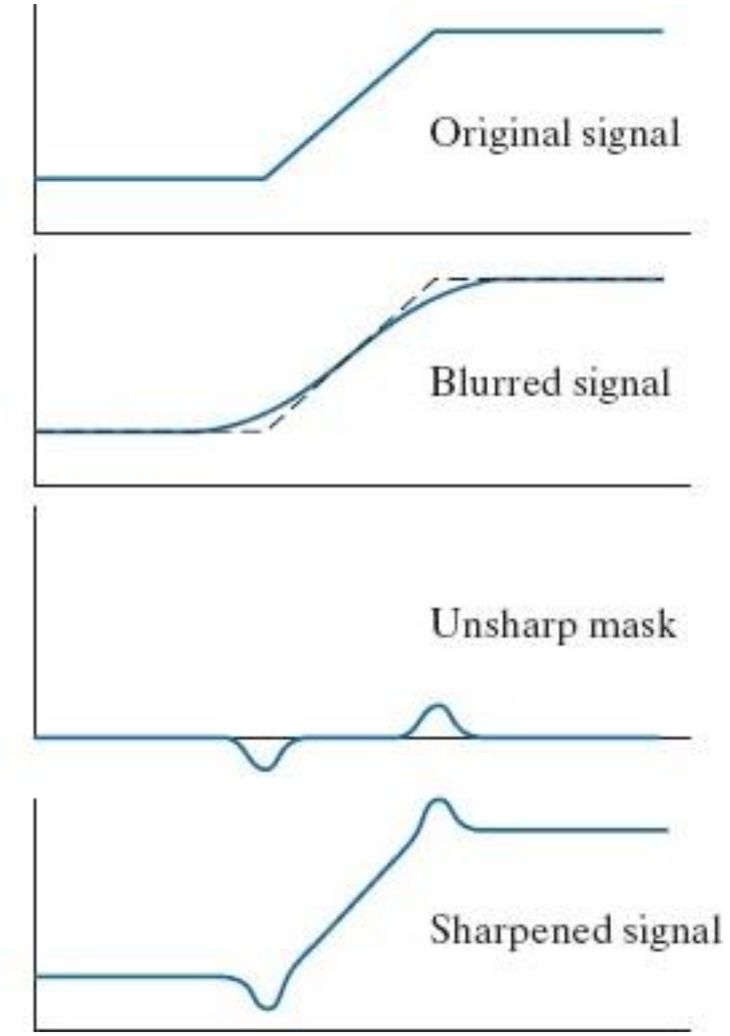
Output: Filtered image
(noise averaged out)

Box filter variants: Impulse filter

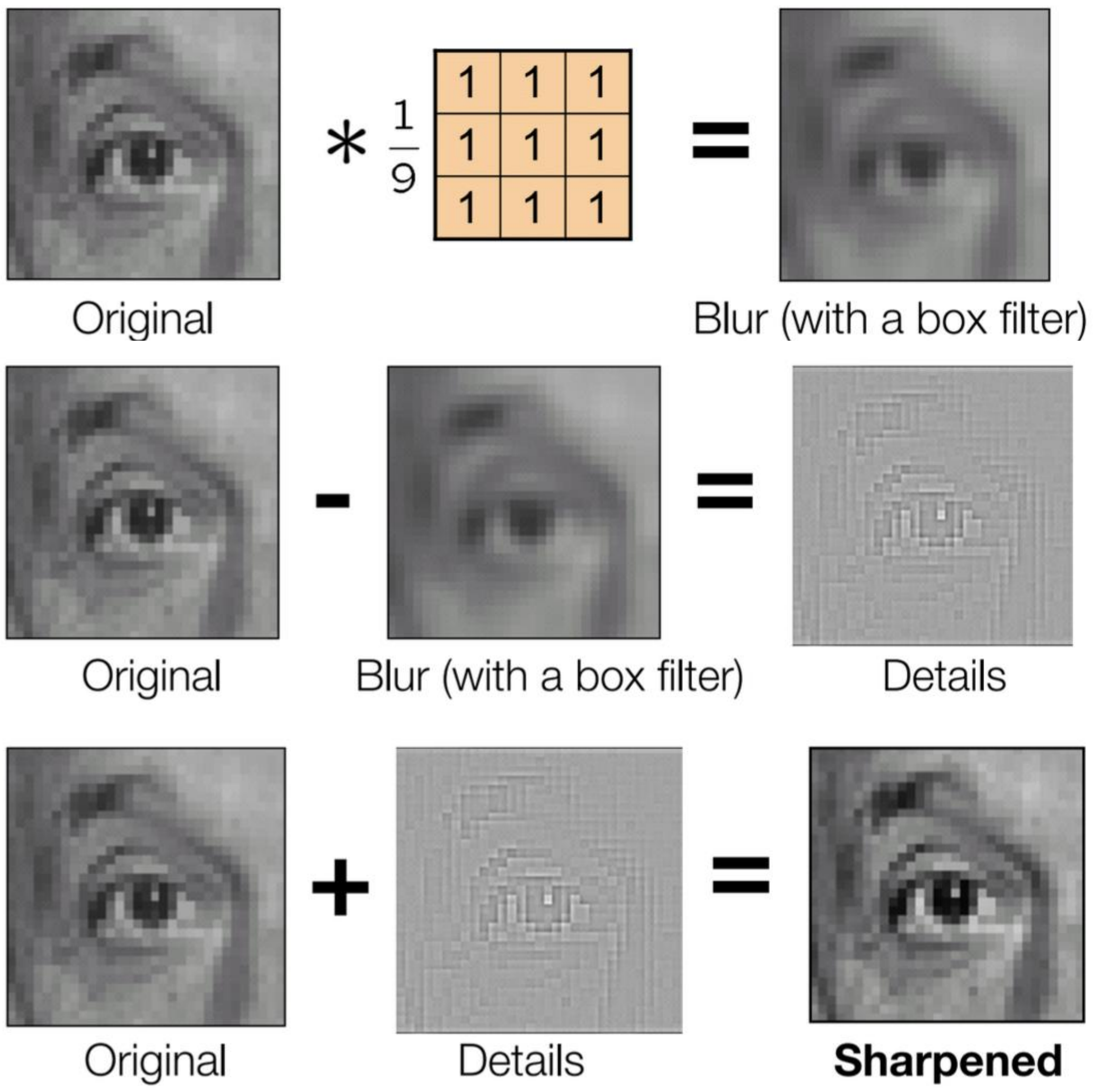


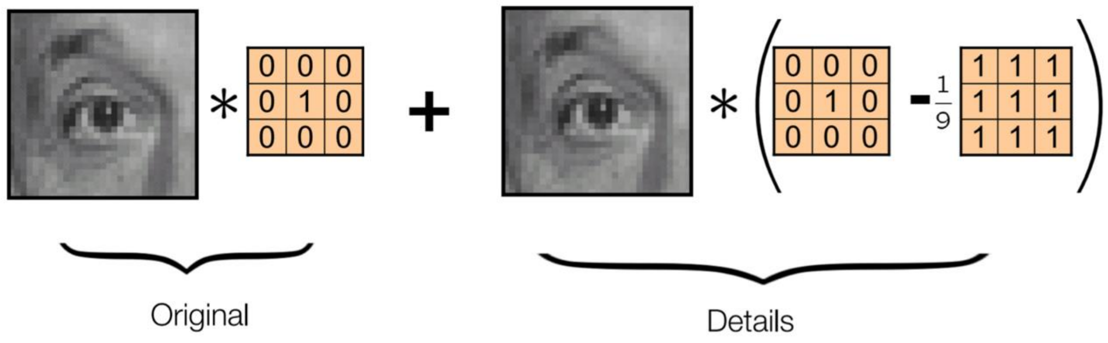
Box filter variants: Sharpen filter



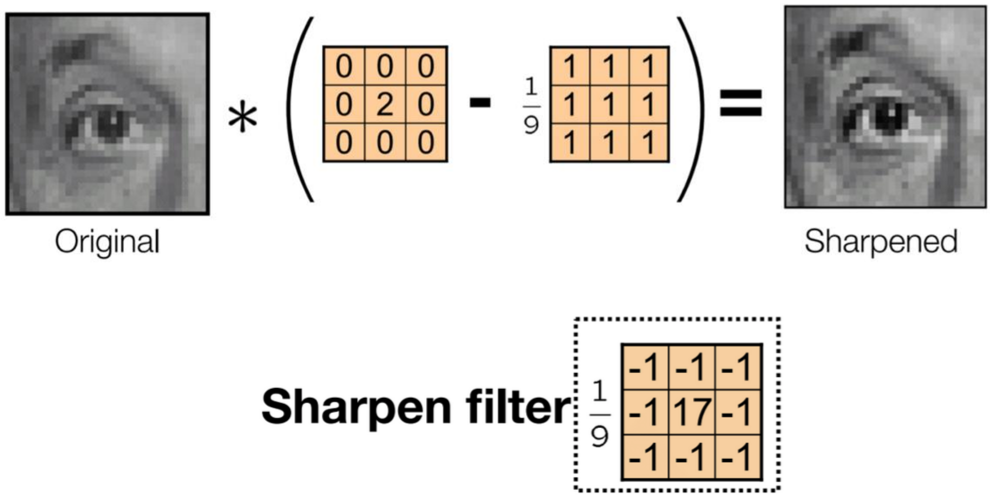


Box filter variants: Sharpen filter





Convolution filters are linear!



Gaussian filter

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

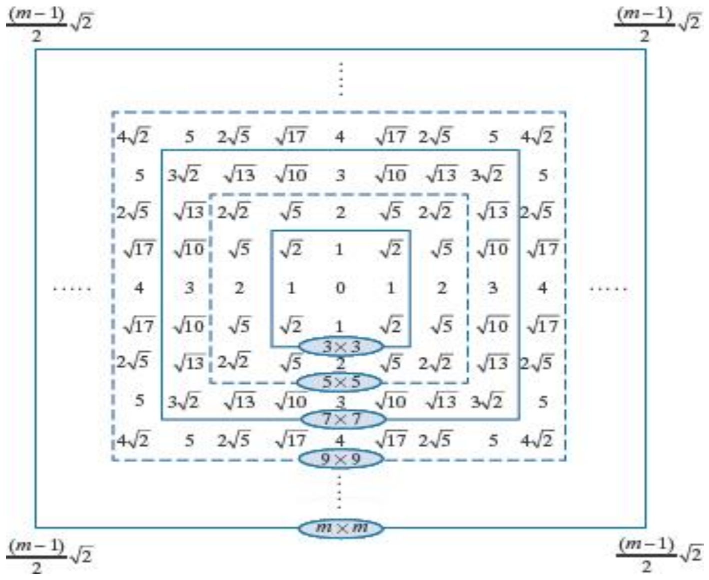
$$\frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

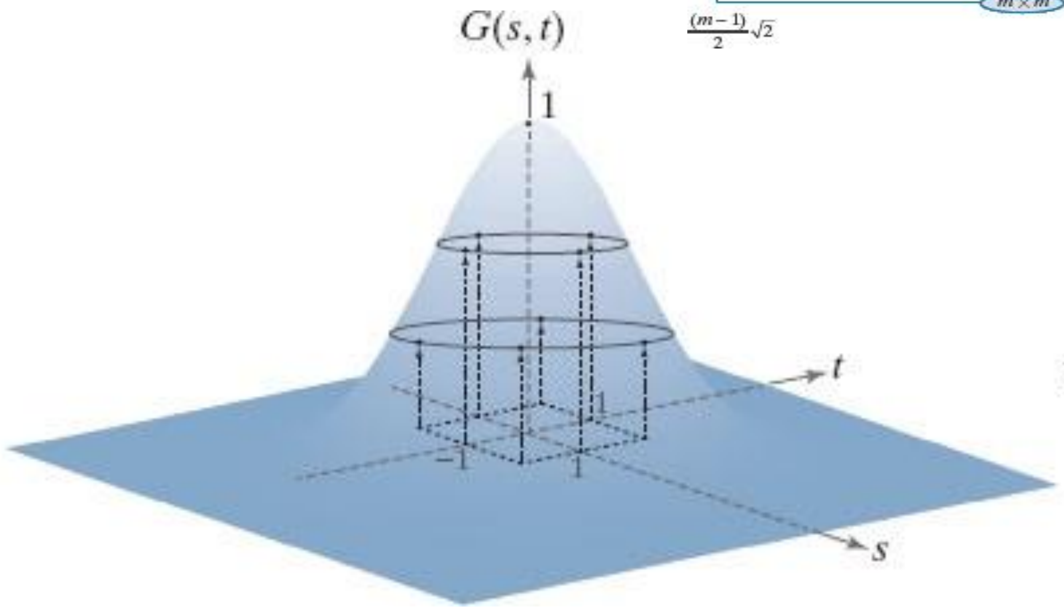
Box filter

$$\frac{1}{4.8976} \times \begin{bmatrix} 0.3679 & 0.6065 & 0.3679 \\ 0.6065 & 1.0000 & 0.6065 \\ 0.3679 & 0.6065 & 0.3679 \end{bmatrix}$$

Gaussian filter

Gaussian filter





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

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

0.3679	0.6065	0.3679
0.6065	1.0000	0.6065
0.3679	0.6065	0.3679

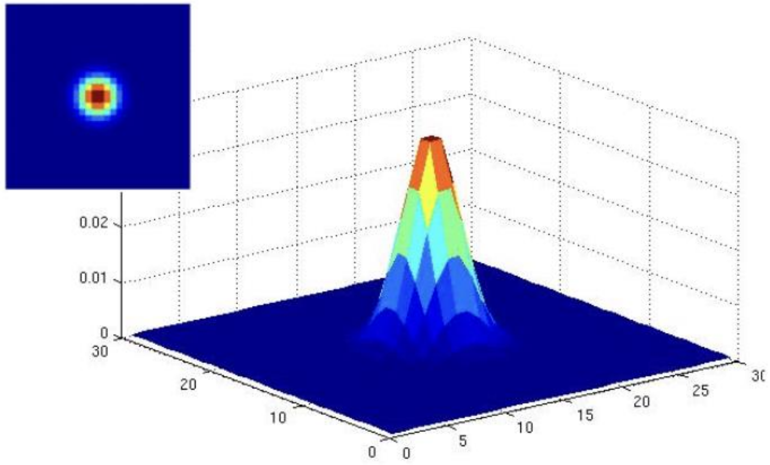
Resulting 3×3 kernel

Result comparison

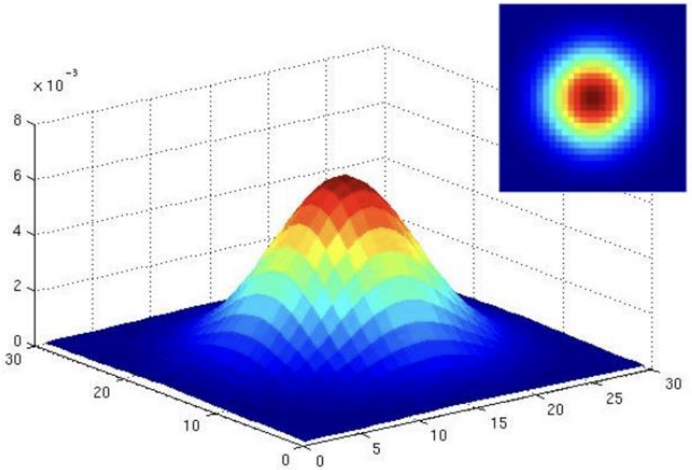
Ghosting grid artifact

Gaussian filter parameter: standard deviation

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



kernel: σ = 2 with 30 x 30

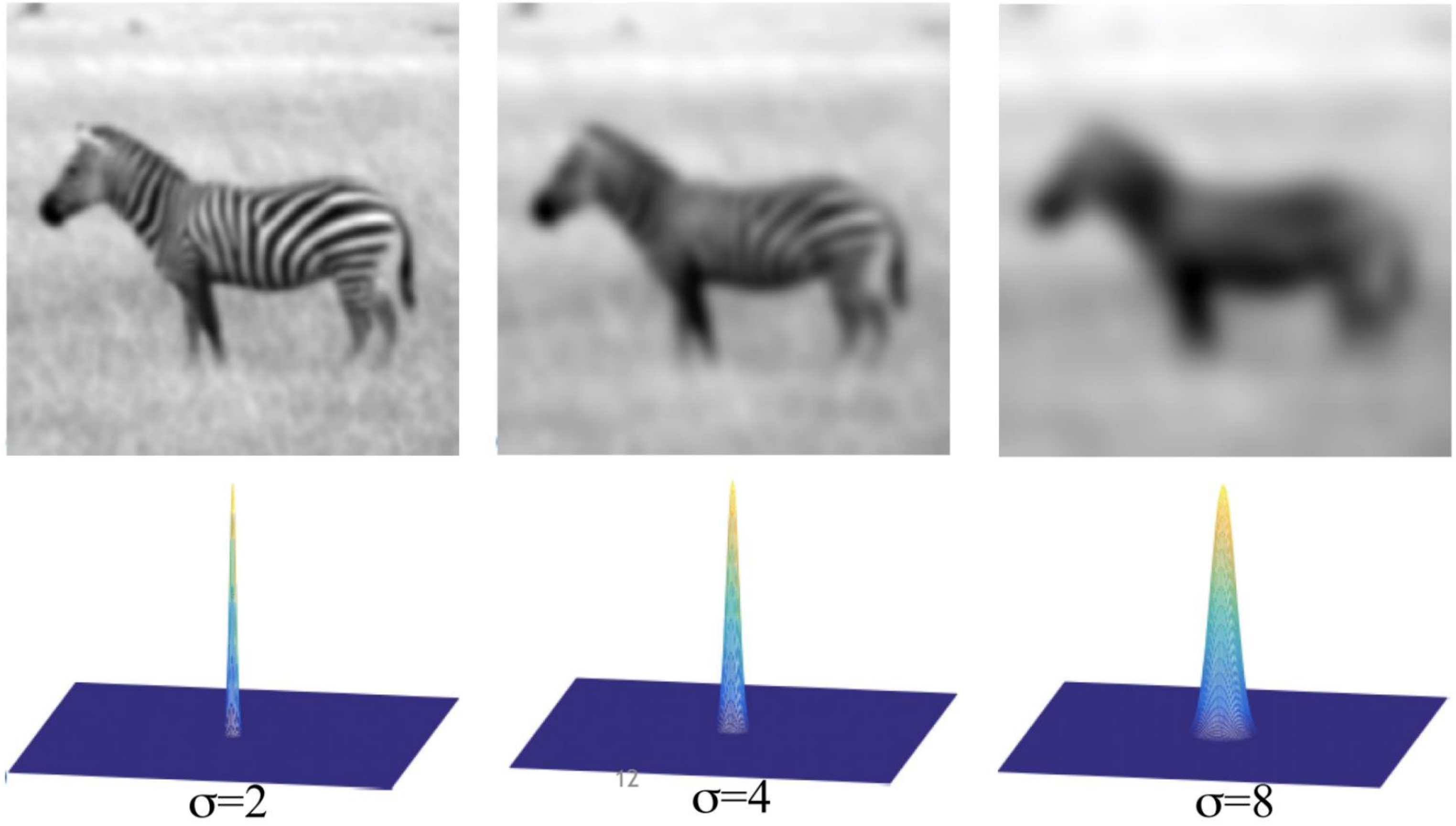


kernel: σ = 5 with 30 x 30

Standard deviation σ: determines extent of smoothing

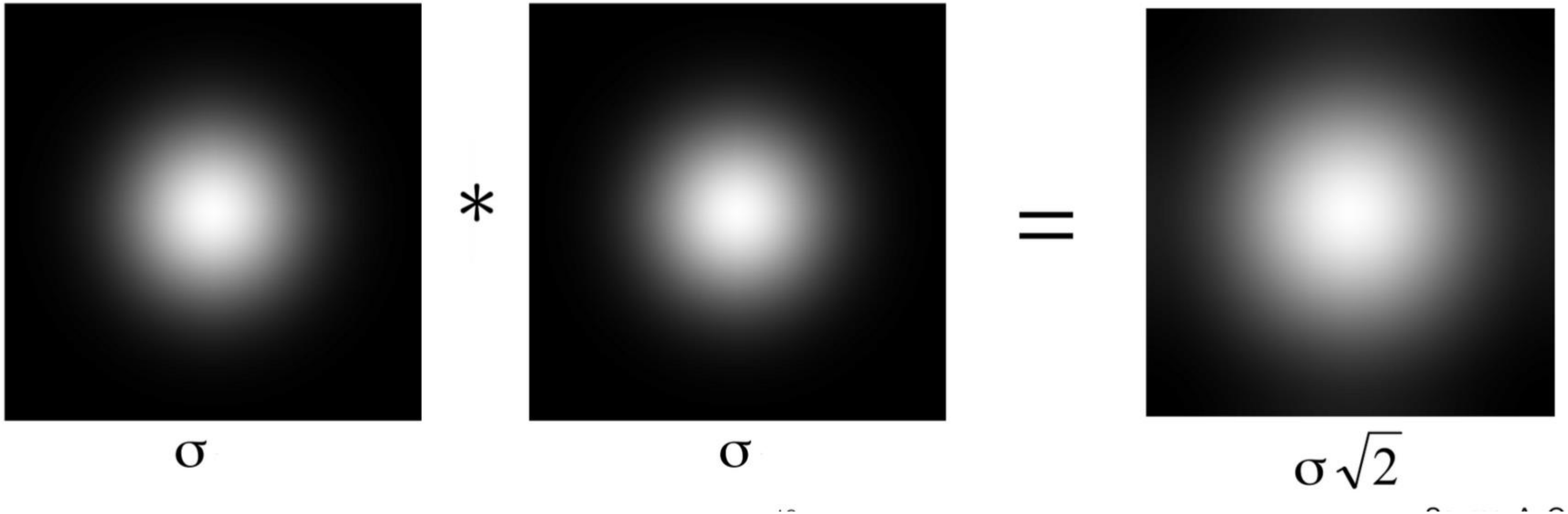
Gaussian filter parameter: standard deviation

- Bigger σ
  - More blurry
  - Bigger effective kernel size



Gaussian filter property: Recursive

- Convolve a Gaussian filter with itself?



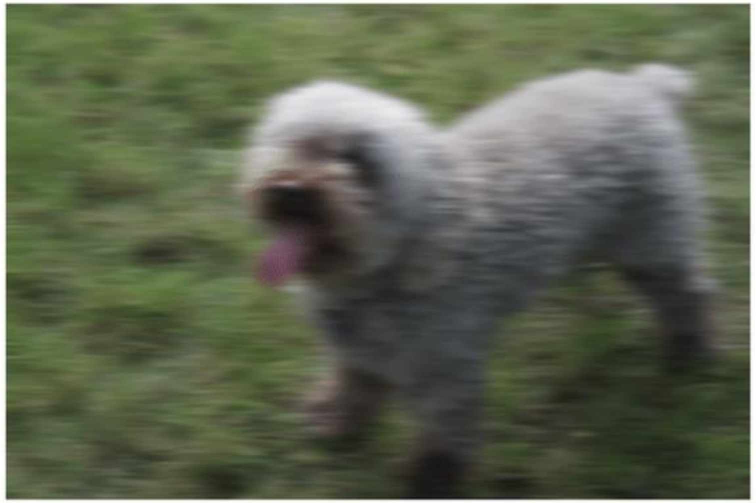
Gaussian filter property: separability

- 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 $O(n^2)$

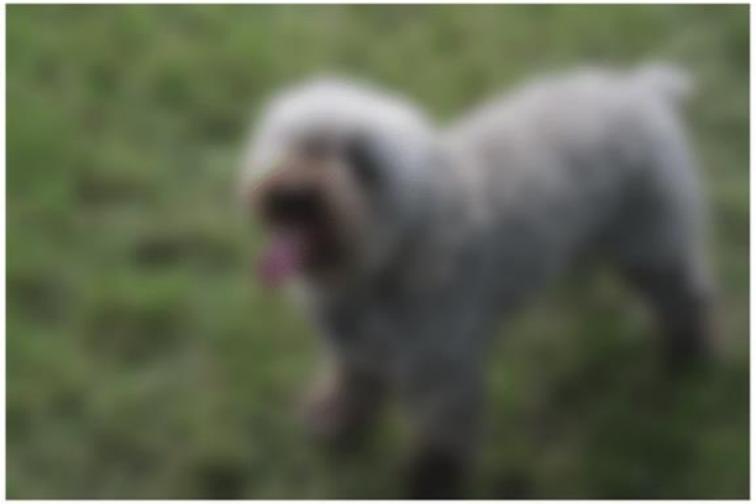
$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

Separable kernel

I



$\mathrm{blur}_x(\mathrm{I})$



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