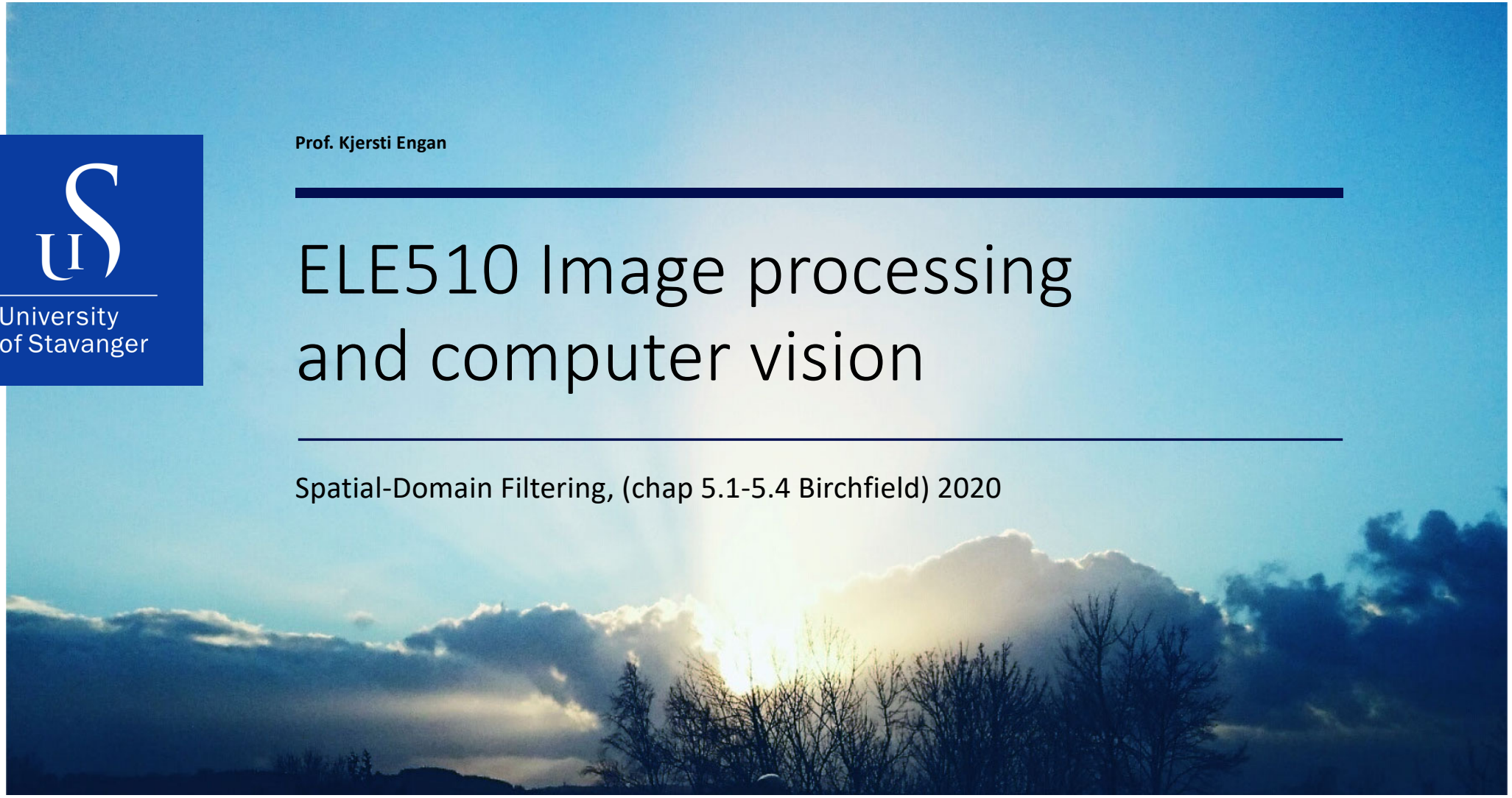
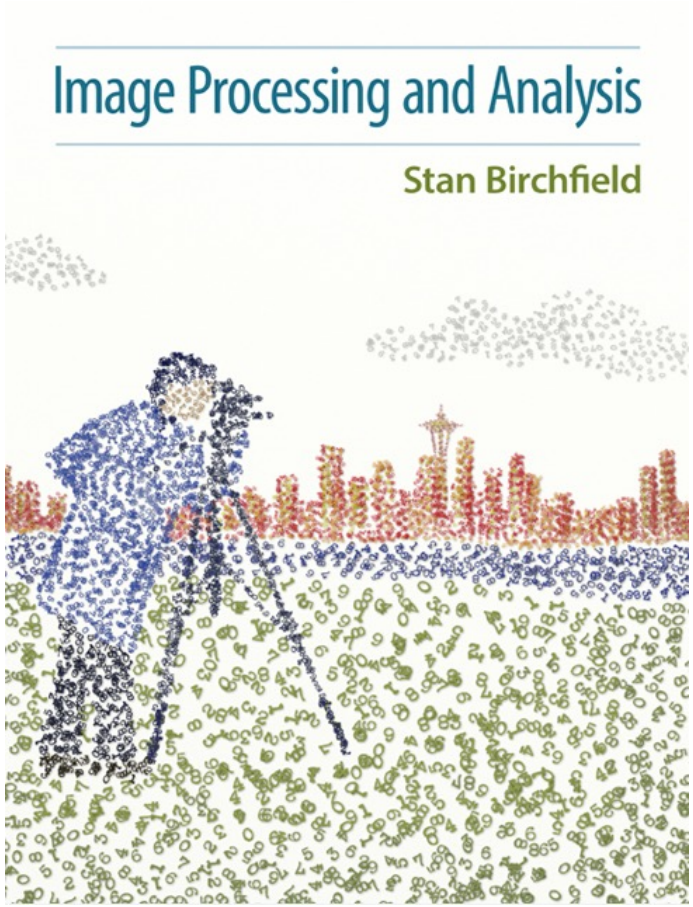


University of Stavanger

Prof. Kjersti Engan

ELE510 Image processing and computer vision

Spatial-Domain Filtering, (chap 5.1-5.4 Birchfield) 2020



Parts in course presentations is material from Cengage learning. It can be used for teaching, and it can be share it with students on access controlled web-sites (Canvas) for use in THIS course. **But it should not be copied or distributed further in any way** (by you or anybody).

Spatial domain filtering – linear filters - smoothing filters

Three points from the topic:

1. What do we mean by spatial domain filtering ?
2. What is a linear filter? How does that relate to convolution ?
3. Which filter kernels can give a smoothing/blurring effect?

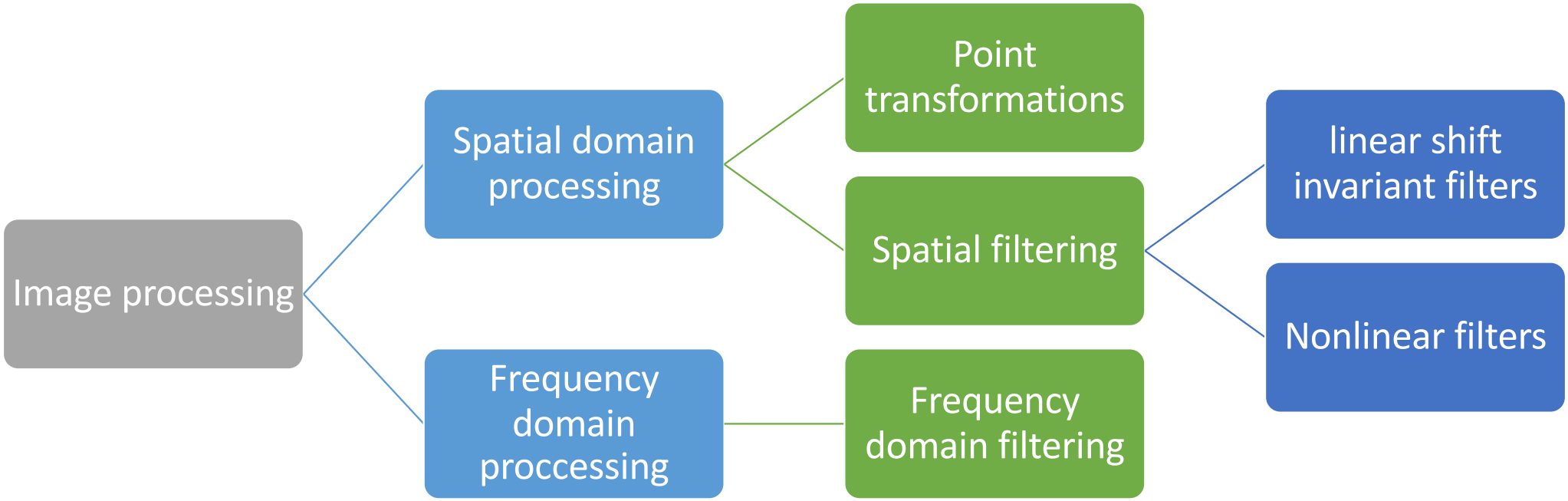
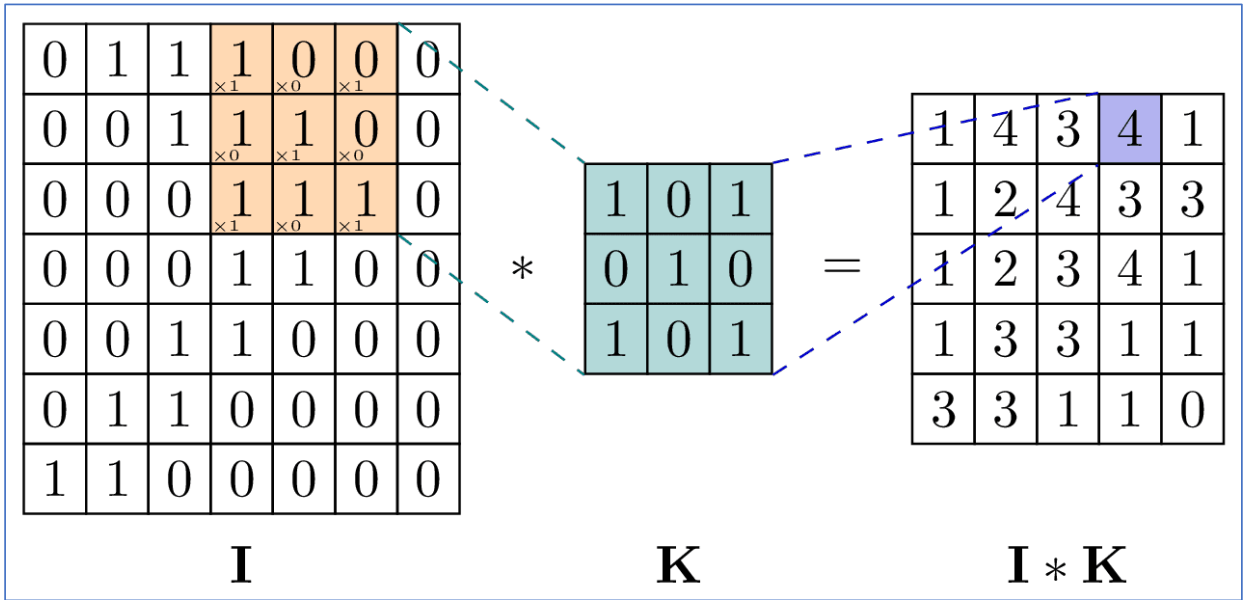


Image processing
Spatial domain processing
Frequency domain proccessing
Point transformations
Spatial filtering
Frequency domain filtering
linear shift invariant filters
Nonlinear filters

Spatial Domain Filtering

- Remember for point transformations, output value at a position (x,y) is dependent on the input value at (x,y) (Examples: thresholding, histogram operations) : $I'(x,y) = f(I(x,y))$

- For spatial domain filtering in general, output pixel at position (x,y) depends on input value at (x,y) AND neighbouring pixels. This can be defined by convolution for a special subset of filters (linear shift-invariant filters).



Convolution

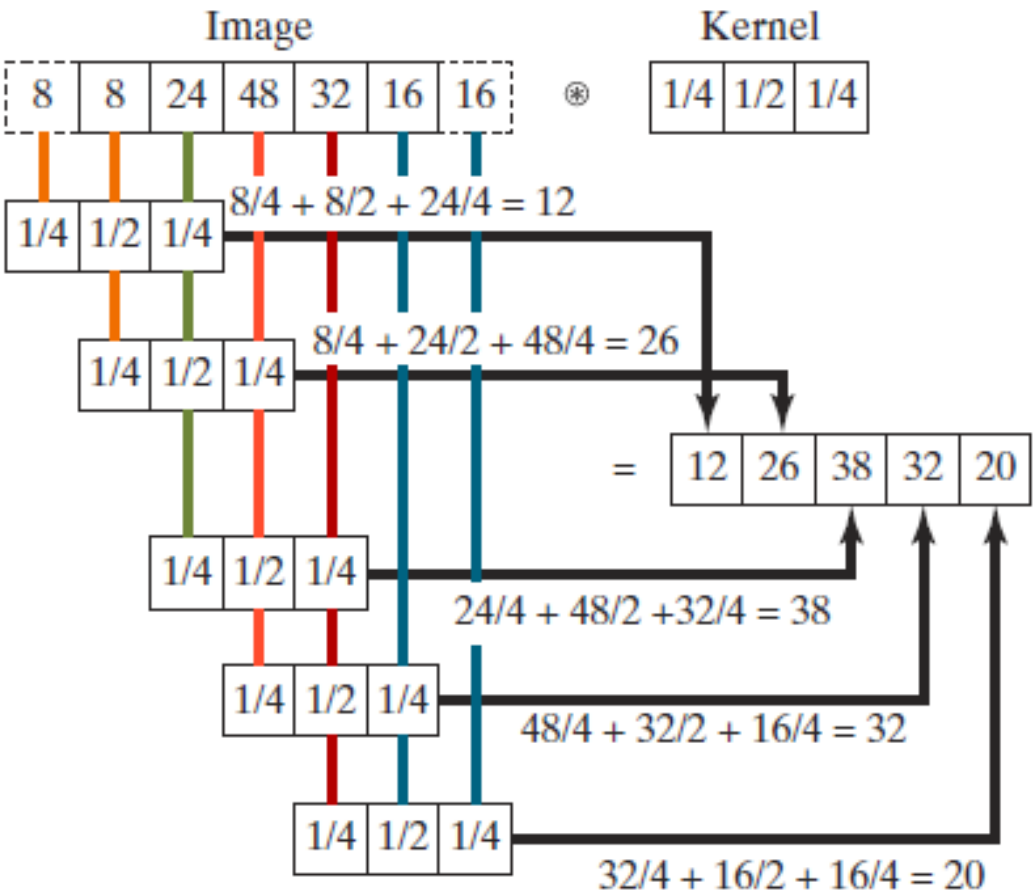
- The discrete convolution of a 1D signal *f* with a kernel *g* is defined as:

$$f'(x) = f(x) \circledast g(x) \equiv \sum_{i=-\infty}^{\infty} f(x-i)g(i)$$

$$= \sum_{i=-\tilde{w}}^{w-\tilde{w}-1} f(x-i)g(i)$$

- *w* is the width of the kernel
- The **origin** $\tilde{w}$ of the kernel indicates the location where the result is stored, often the index nearest the center.
- Usually *w* is odd and $\tilde{w}$ is in the middle of the kernel.

Example of 1-D convolution



Have to flip the kernel (here: symetric..) and slide it over the signal

Border/edge strategy

- How to deal with the edges of the signal/image?
- Zero-padding
- Mirror
- Repeat
- truncate

Convolution or correlation?

Convolution is closely related to **cross-correlation**, which is defined as:

$$\tilde{w} \equiv \left\lfloor \tfrac{1}{2}(w-1) \right\rfloor$$

$$f'_{corr}(x) = f(x) \check{\circledast} g(x) \equiv \sum_{i=-\infty}^{\infty} f^*(x+i)g(i) = \sum_{i=-\tilde{w}}^{w-\tilde{w}-1} f^*(x+i)g(i)$$

If f(x) is real, the only difference is that convolution flips the kernel.

If kernel is symmetric and f(x) is real -> No difference

(Some filtering functions might be implemented with correlation)

Kernels – some intuition

- Smoothing kernels: averaging, noise reduction, low pass filtering:

$$\sum_i g(i) = 1$$

usually symmetric

Differentiating kernels: Extract boundaries, edgess, high pass filters

$$\sum_i g(i) = 0$$

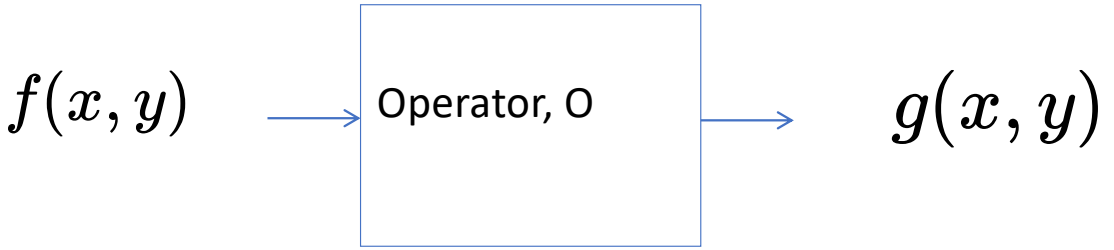
usually symmetric OR antisymmetric

Convolution as matrix multiplication

- Convolution can be written as a matrix – vector multiplication

$$\underbrace{\begin{bmatrix} 12 \\ 26 \\ 38 \\ 32 \\ 20 \end{bmatrix}}_{f'} = \underbrace{\frac{1}{4}\begin{bmatrix} 1 & 2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 1 \end{bmatrix}}_{\substack{\textit{convolution matrix} \\ \mathbf{G}}} \underbrace{\begin{bmatrix} 8 \\ 8 \\ 24 \\ 48 \\ 32 \\ 16 \\ 16 \end{bmatrix}}_{f}$$

Image transformations, linear shift-invariant systems



Linear operator: $$O[af_1 + bf_2] = aO[f_1] + bO[f_2]$$

A system is called **shift-invariant** if a shift in the input causes a shift in the output by the same amount.

$$f'(x - x_0) = \mathcal{L}(f(x - x_0))$$

Linear shift-invariant systems: systems that are particularly important due to their convenient mathematical properties. Often called LTI system (linear time-invariant systems) .

If a system is not linear, then it is said to be **nonlinear**.

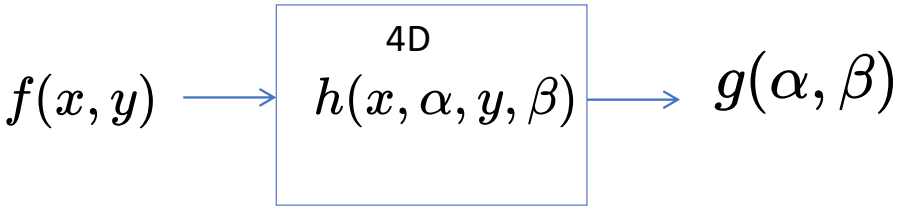
Point spread function

Point source, δ(α-x,β-y)

Point spread function (PSF) of the operator.

The PSF *defines* a linear operator
Equivalent to impulse response in signal processing

Point spread function



In general; all the pixel in the input function can contribute to the output function

The point spread function express how much the input value at position (x,y) influences the output value at position (α,β)

$$g(\alpha,\beta) = \sum_{x=1}^{N}\sum_{y=1}^{M} f(x,y)h(x,\alpha,y,\beta)$$

If the system is linear shift-invariant (LTI), the Point Spread Function becomes a 2D function

$$h(\alpha - x, \beta - y)$$

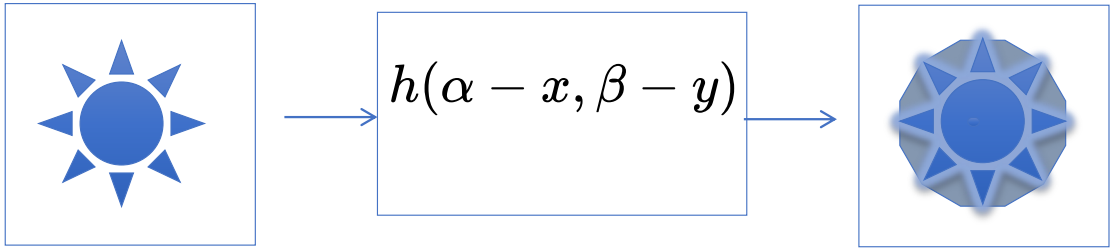
Linear systems/filters

- Linear systems / filters are completely defined by the impuls response / point spread function.

- FIR: Finite Impulse Response. A FIR filter is a linear system or filter where h(x) is finite in duration. Can be implemented with a kernel and convolution.
- IIR: Infinite Impulse Response. Typically implemented in frequency domain or as a reqursive system.

Point spread function (PSF) and convolution

Shift invariant point spread function : LTI system



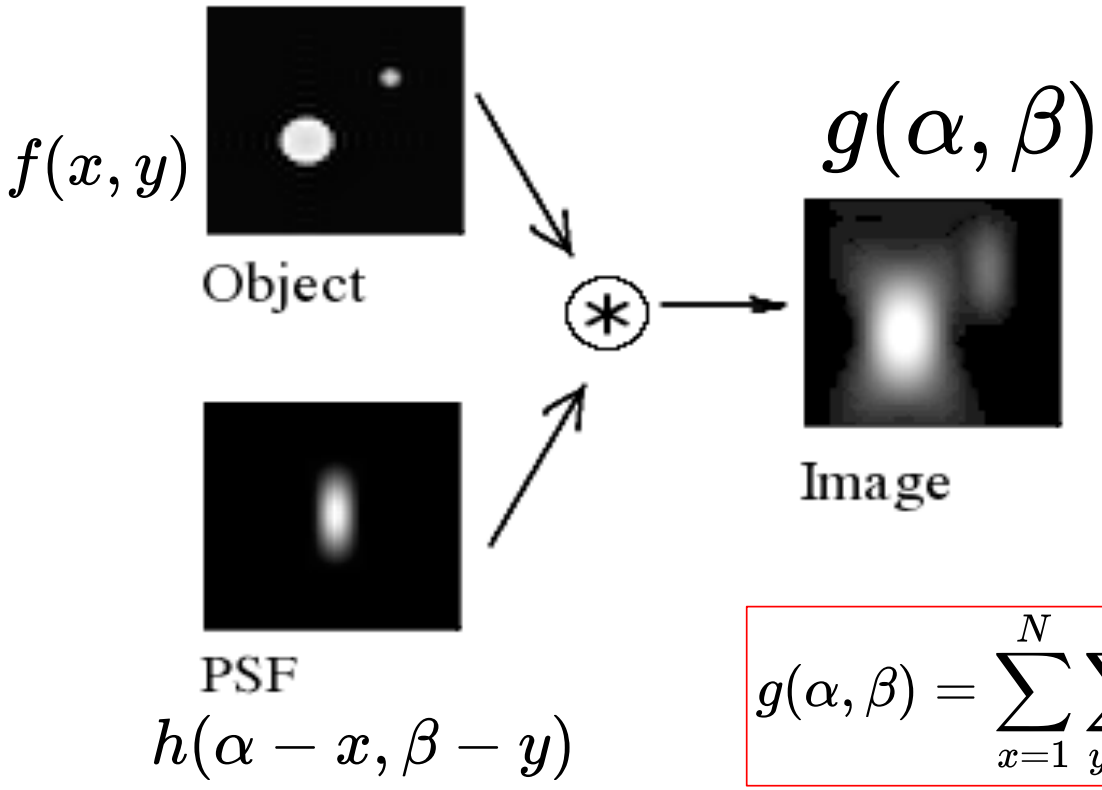
$f(x, y)$ $g(\alpha, \beta)$

$$g(\alpha, \beta) = \sum_{x=1}^{N} \sum_{y=1}^{N} f(x, y) h(\alpha - x, \beta - y)$$

Convolution is commutative:

$$g(\alpha, \beta) = \sum_{x=1}^{N} \sum_{y=1}^{N} h(x, y) f(\alpha - x, \beta - y)$$

To use the convolution formula:
System/filter needs to be linear AND shift(time)-invariant.



$$g(\alpha,\beta)=\sum_{x=1}^{N}\sum_{y=1}^{N}f(x,y)h(\alpha-x,\beta-y)$$

Convolution as Fourier multiplication

- Convolution in the spatial domain is equivalent to multiplication in the frequency domain.

$$f'(x) = f(x) \circledast g(x) = \sum_{i=-\infty}^{\infty} f(x-i)g(i)$$

$$f'(x) = f(x) \circledast g(x) = \mathcal{F}^{-1}\{\mathcal{F}\{f(x)\} \cdot \mathcal{F}\{g(x)\}\}$$

2D convolution

2D Convolution: used to perform filtering on a 2D image, books notation.

$$I'(x,y) = I(x,y) \circledast G(x,y) = \sum_{i=0}^{w-1} \sum_{j=0}^{h-1} I(x+\tilde{w}-i, y+\tilde{h}-j) G(i,j)$$

where *w* and *h* are the width and height of the kernel G(i,j).

Flip-flop the kernel and slide it over the image

In image processing we usually use FIR filters with relatively small filter kernels (can use convolution implementation, can use LTI system theory)

OR nonlinear filters (next week).

Mean / box filter (LTI filter; we use convolution)

A kernel with all values h(x) or h(x,y) equal gives a box filter.

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

$f[x, y]$

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

$h[u, v]$

If we want the output to have approximately the same amount of energy, the kernel values should sum to one.

Equal weight to all pixels within the neighborhood

Mean kernel – box filter

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

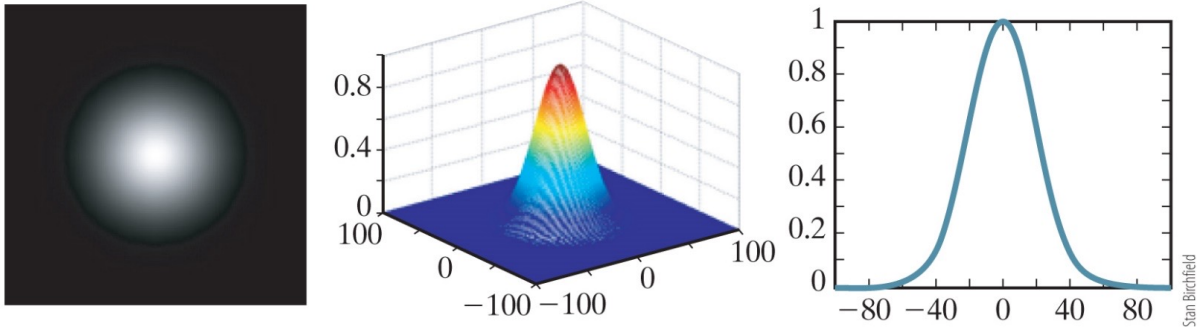
$f[x, y]$

	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	

$g[x, y]$

(5.2) Gaussian filters

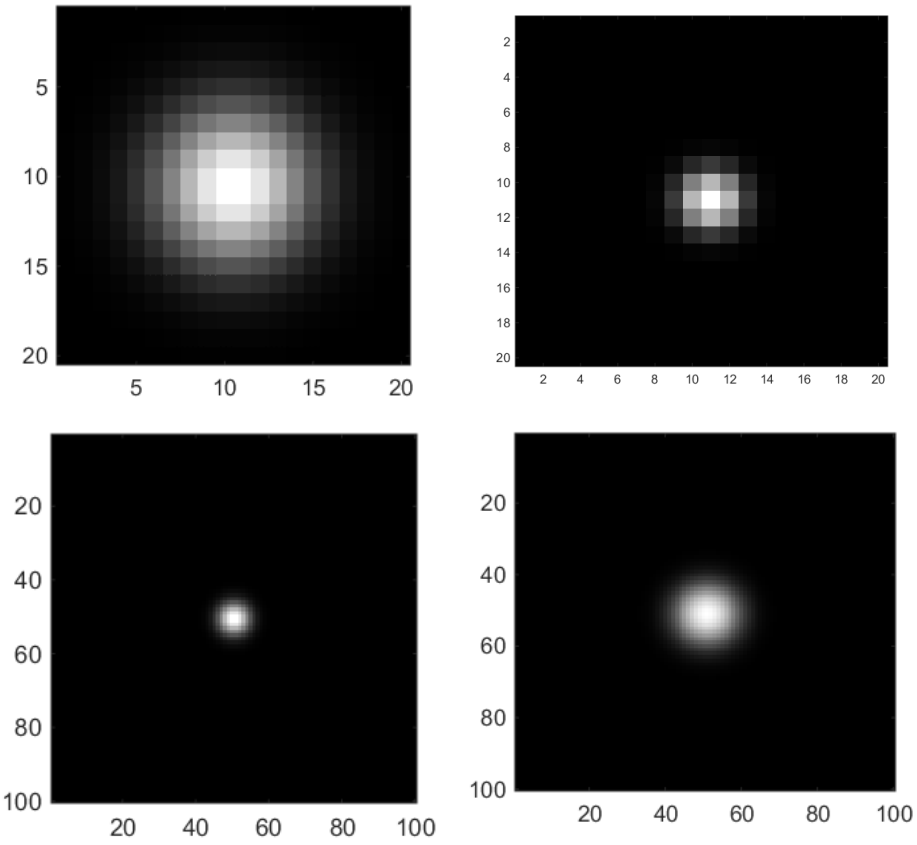
Figure 5.3 A Gaussian is a bell curve. From left to right: The 2D isotropic Gaussian viewed as an image where the gray level of each pixel is proportional to the value of the Gaussian function at that point, the 2D isotropic Gaussian viewed as a surface in 3D, and the 1D Gaussian function (or, equivalently, a slice through the 2D Gaussian function, obtained by intersecting it with a vertical plane).



$$gauss_{\sigma^2}(x) \equiv \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-x^2}{2\sigma^2}\right)$$

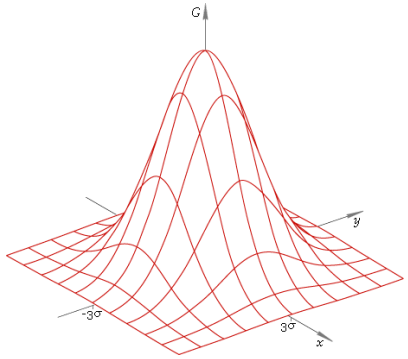
Convolving a box filter with itself yields an approximation to a Gaussian kernel.

Gaussian filters



Space domain

Frequency domain
(magnitude of DFT)



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

Gaussian function in space domain becomes gaussian in frequency domain.

Gaussian filtering

A **Gaussian kernel** gives less weight to pixels further from the center of the window

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

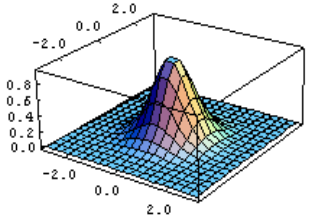
$f[x, y]$

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

$h[u, v]$

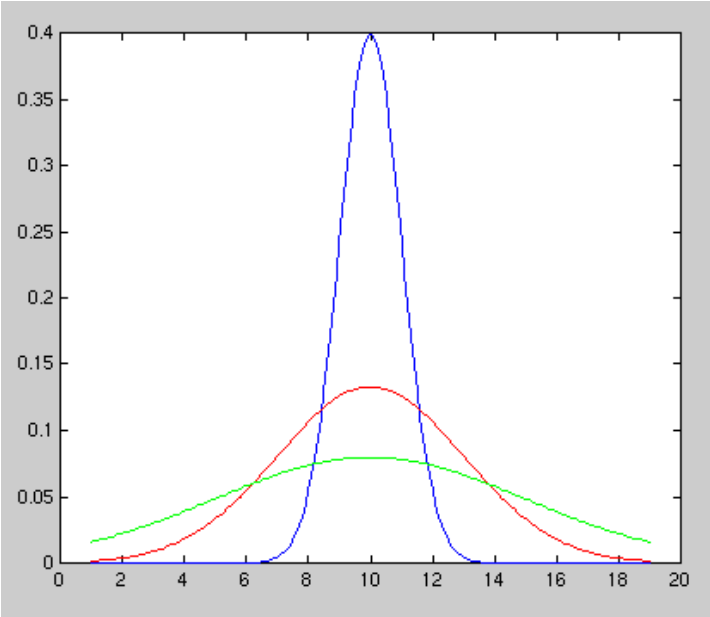
is a discrete approximation of a Gaussian function:

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

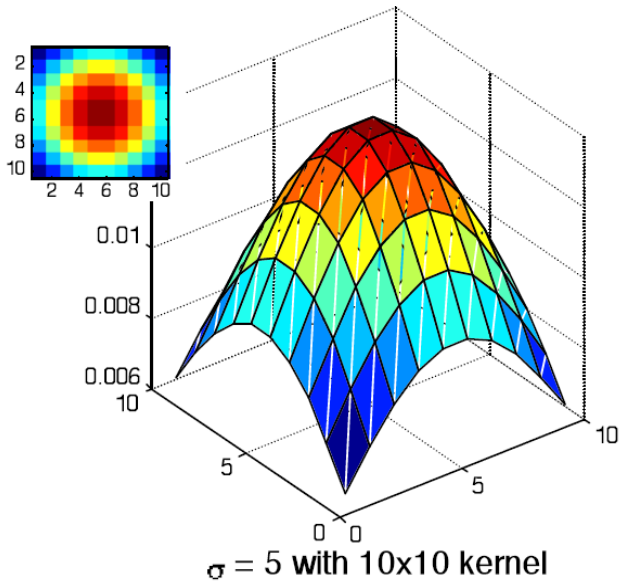


Note about Finite Kernel Support

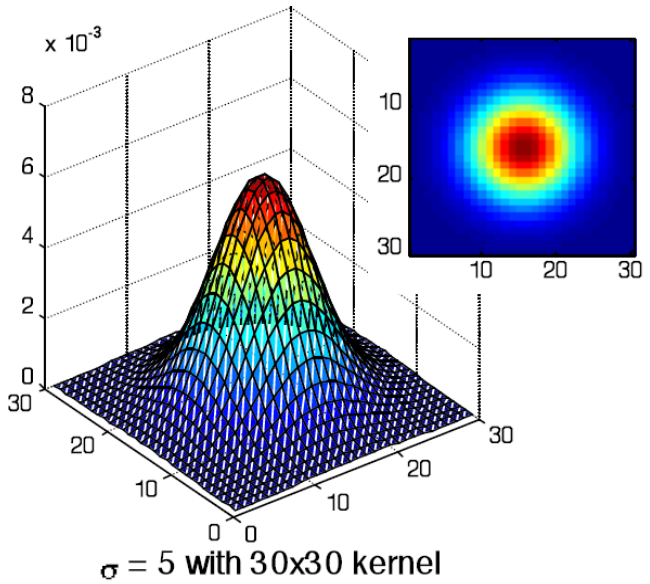
Gaussian function has infinite support. In discrete filtering, we want a finite kernel



Shape of Gaussian with different σ

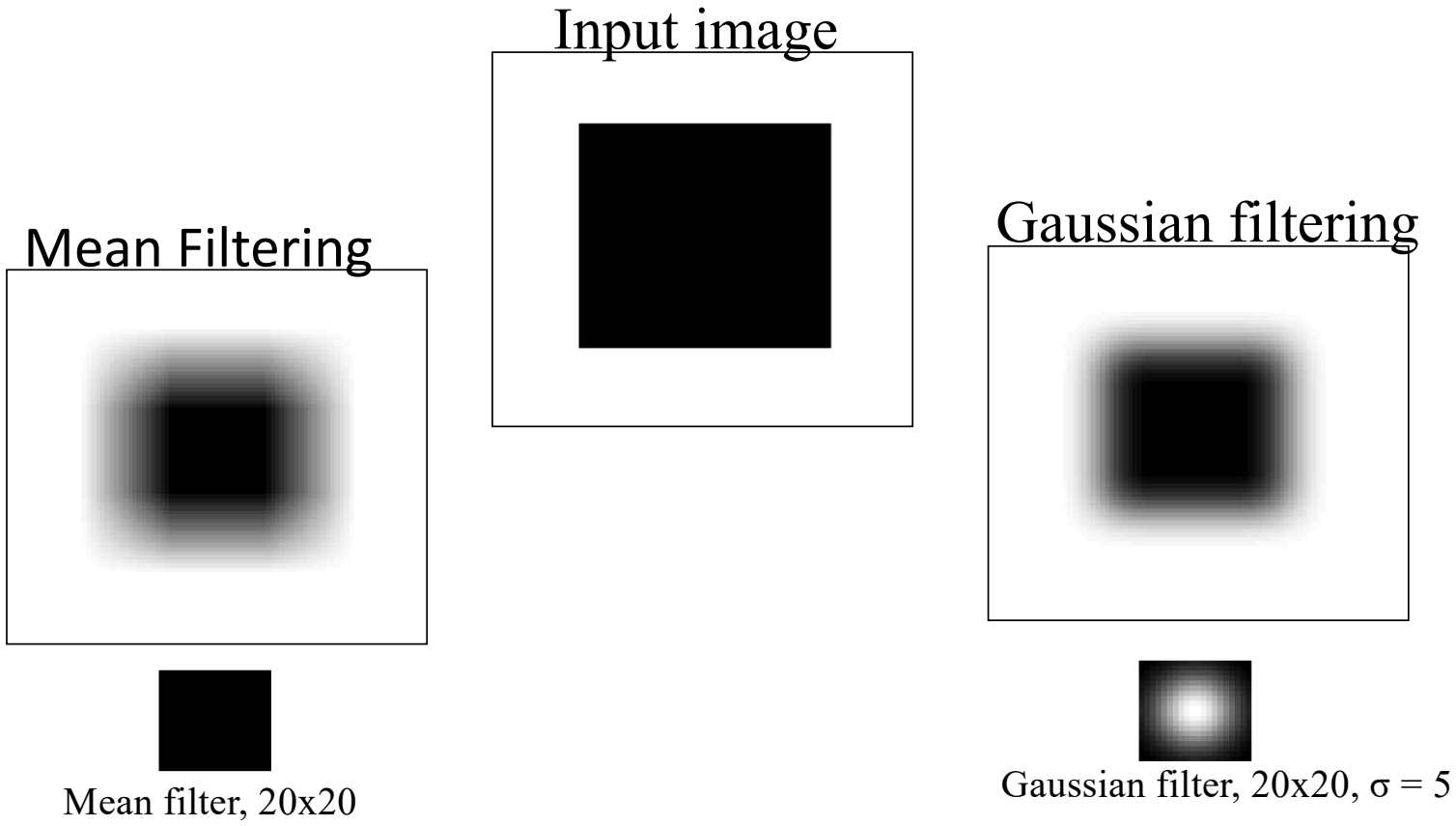


$\sigma = 5$ with 10x10 kernel



$\sigma = 5$ with 30x30 kernel

Box filter vs. Gauss filter



Separable filters

- Gaussian filters are separable.
- If the 2D filter kernel can be written as convolution of 2 1D filter kernels, then the 2D filter is separable. More computational effective to implement!
- If kernel viewed as matrix, separability occurs if all rows and columns are linearly dependent. In other words, 2D kernel has rank = 1. In this case the 2D kernel can be viewed as outer product of the two 1D kernels when viewed as vectors.

2D Convolution with separable kernel

- When the kernel is separable there is a compact, elegant notation for the 2D convolution:

$$\mathbf{I}' = \mathbf{G}_h \mathbf{I} \mathbf{G}_v^\top$$

Where $\mathbf{G}_h$ is the convolution matrix constructed from the 1D kernel $\mathbf{g}_h$.

Instead of doing a 2D convolution we do 2 X 1D convolution:

First convolve over the columns.

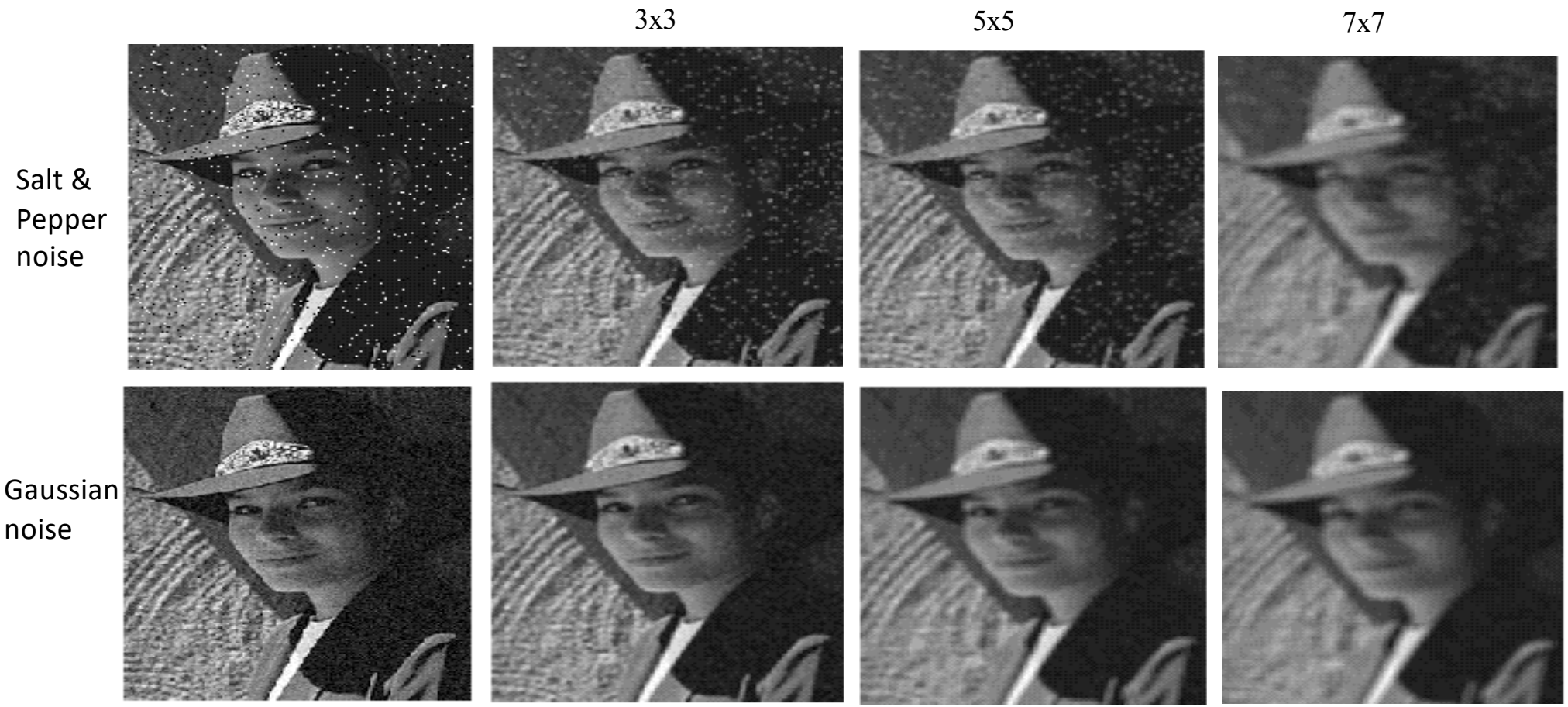
Then convolve results of first convolution ower the rows.

Noise

- Additive noise: In(x,y) = I(x,y) +n(x,y)
- Additive gaussian noise: the noise follows a random gaussian distribution with some sigma and typically mean=0.
- **Salt-and-pepper noise**: each pixel is set to either the minimum (“pepper”) or maximum (“salt”) possible gray level, or it remains unchanged:

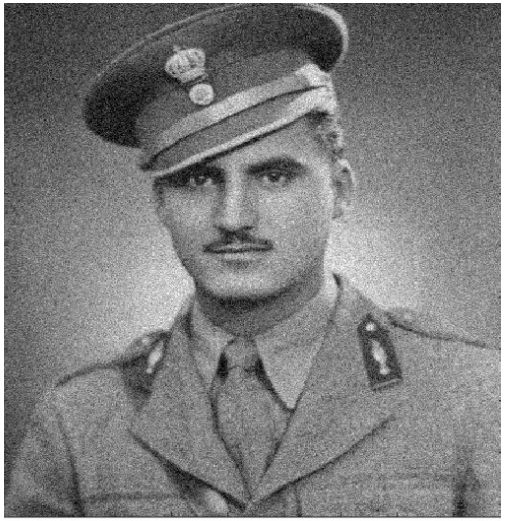
$$I'(x,y) = \begin{cases} 0 & \text{if } 0 \le \xi < p \\ 255 & \text{if } p \le \xi < p+q \\ I(x,y) & \text{otherwise} \end{cases} \qquad \xi \sim U(0,1)$$

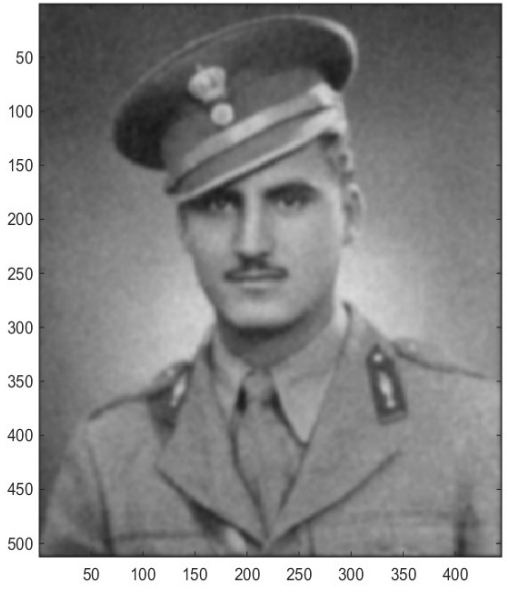
Effect of mean /box filtering



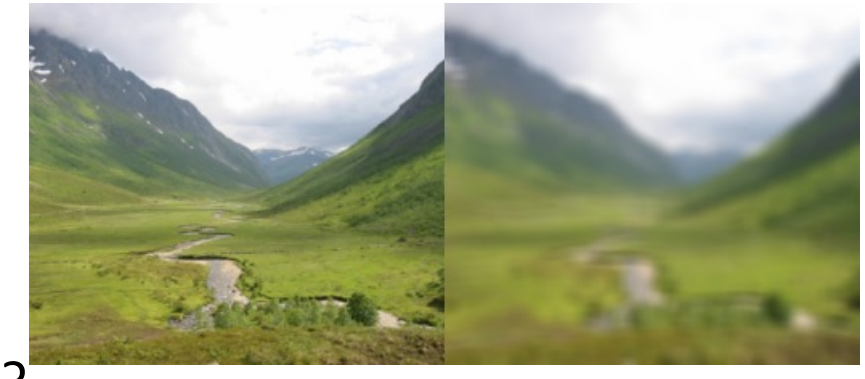
Side effect - blur

Additiv Gaussian noise
Linear Gaussian filtering





Spatial domain filtering – linear filters - smoothing filters



Three points from the topic:

1. What do we mean by spatial domain filtering ?
   - ✓ Manipulating the pixels directly (point or filter/kernel)

2. What is a linear filter? How does that relate to convolution?
   - ✓ Linear filter def.. To use the convolution formula; system has to be linear AND shift invariant

3. Which filter kernels can give a smoothing/blurring effect?
   - ✓ Box filters, gaussian filters