Computación Gráfica

Class 4. Images and Signal Processing.

Professor: Eric Biagioli

Today

Discussion on convolutional filters.

Some words on the Fourier Transform of an image.

Discussion about current applications in industry and/or about state of the art in Image Processing.

<u>**References for the class of today:**</u> (part of the first partial exam)

- [MAIN reference] : Velho, L., Frery, A. C., and Gomes, J. Image Processing for Computer Graphics and Vision, 2nd ed. Springer Publishing Company, Incorporated, 2008. → **Chapter 7. Sections 7.3, 7.4 and 7.5 (filters)**

- [ADDITIONAL reference] : Hughes, J. F., van Dam, A., McGuire, M., Sklar, D. F., Foley, J. D., Feiner, S., and Akeley, K. Computer Graphics: Principles and Practice, 3 ed. Addison-Wesley, Upper Saddle River, NJ, 2013. → **Chapter 18**

Spatially Invariant Linear Filters

- Let h(x, y) be the impulse response of the filter T

$$h(x, y) = T(\delta(x, y))$$

- the image f(x, y) can be expressed as an infinite sum of dirac deltas

$$f(x) = \int_{-\infty}^{+\infty} f(u, v)\delta(u - x,\ v - y)\, du\, dv.$$

- If T is a spatially invariant linear transformation, we have

$$\begin{aligned} Tf(x, y) &= T\bigg(\int_{-\infty}^{+\infty} f(u, v)\delta(u - x, v - y)\, du\, dv\bigg) \\ &= \int_{-\infty}^{+\infty} f(u, v) T(\delta(u - x,\ v - y))\, du\, dv. \end{aligned}$$

Spatially Invariant Linear Filters

- If T is a linear transformation, we have

$$
\begin{aligned}
Tf(x,y) &= T\left(\int_{-\infty}^{+\infty} f(u,v)\delta(u-x,v-y)\,du\,dv\right) \\
&= \int_{-\infty}^{+\infty} f(u,v)T(\delta(u-x,\,v-y))\,du\,dv.
\end{aligned}
$$

- If T is spatially invariant, we have:

$$
Tf(x,y) = \int_{-\infty}^{+\infty} f(x,y)h(u-x,\,v-y)\,du\,dv
$$

- This is the CONVOLUTION of f and h at (x, y). We write $f * h$.
- We call ***KERNEL*** to h.

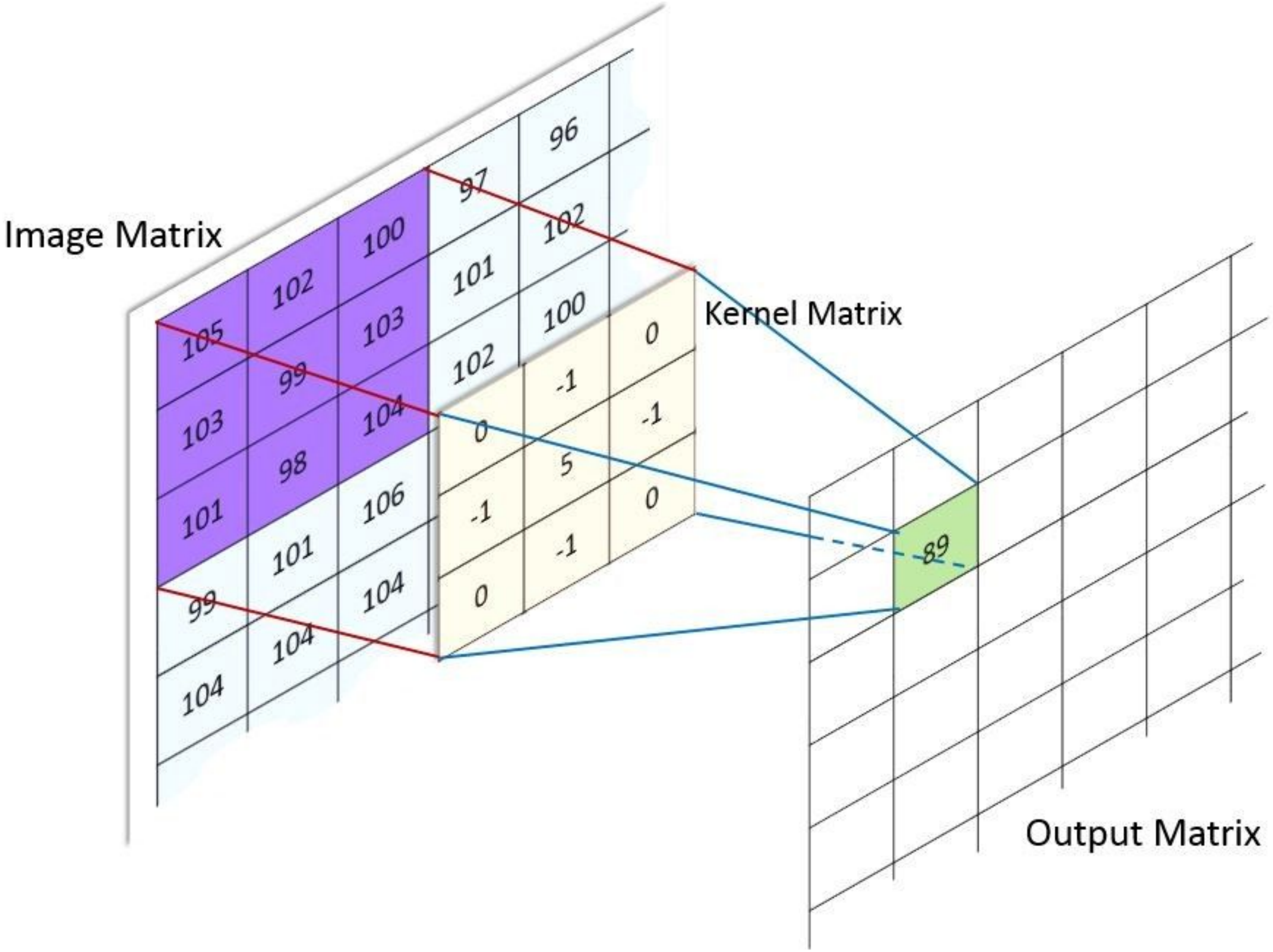
Discrete filters

- In a continuous filter $\int_{\mathbb{R}^2} h(x,y)\,dx\,dy = 1$
- When we discretize $\frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} h_{ij} = 1$
- Kernel: symmetric and odd order

$(-k,k)$	$\cdots$	$(k,0)$	$\cdots$	(k,k)
$\vdots$		$\vdots$		$\vdots$
$(-k,0)$	$\cdots$	$(0,0)$	$\cdots$	$(k,0)$
$\vdots$		$\vdots$		$\vdots$
$(-k,-k)$	$\cdots$	$(-k,0)$	$\cdots$	$(k,-k)$

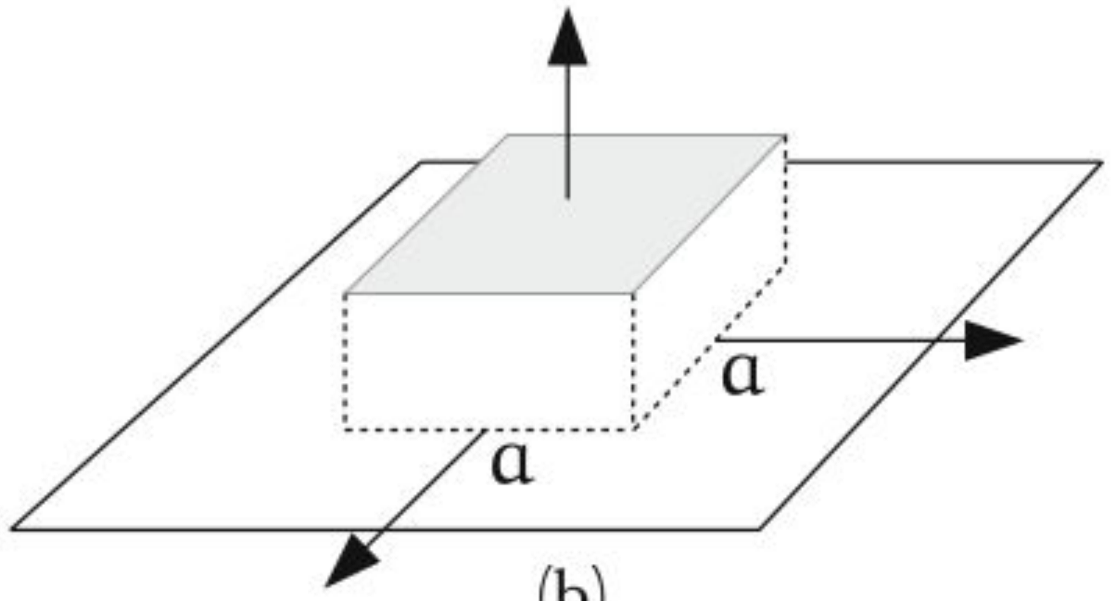
Discrete filters

- How to evaluate them?
- Extensions of the domain
- Complexity?



Discrete filters: examples - BOX FILTERS

BOX FILTER

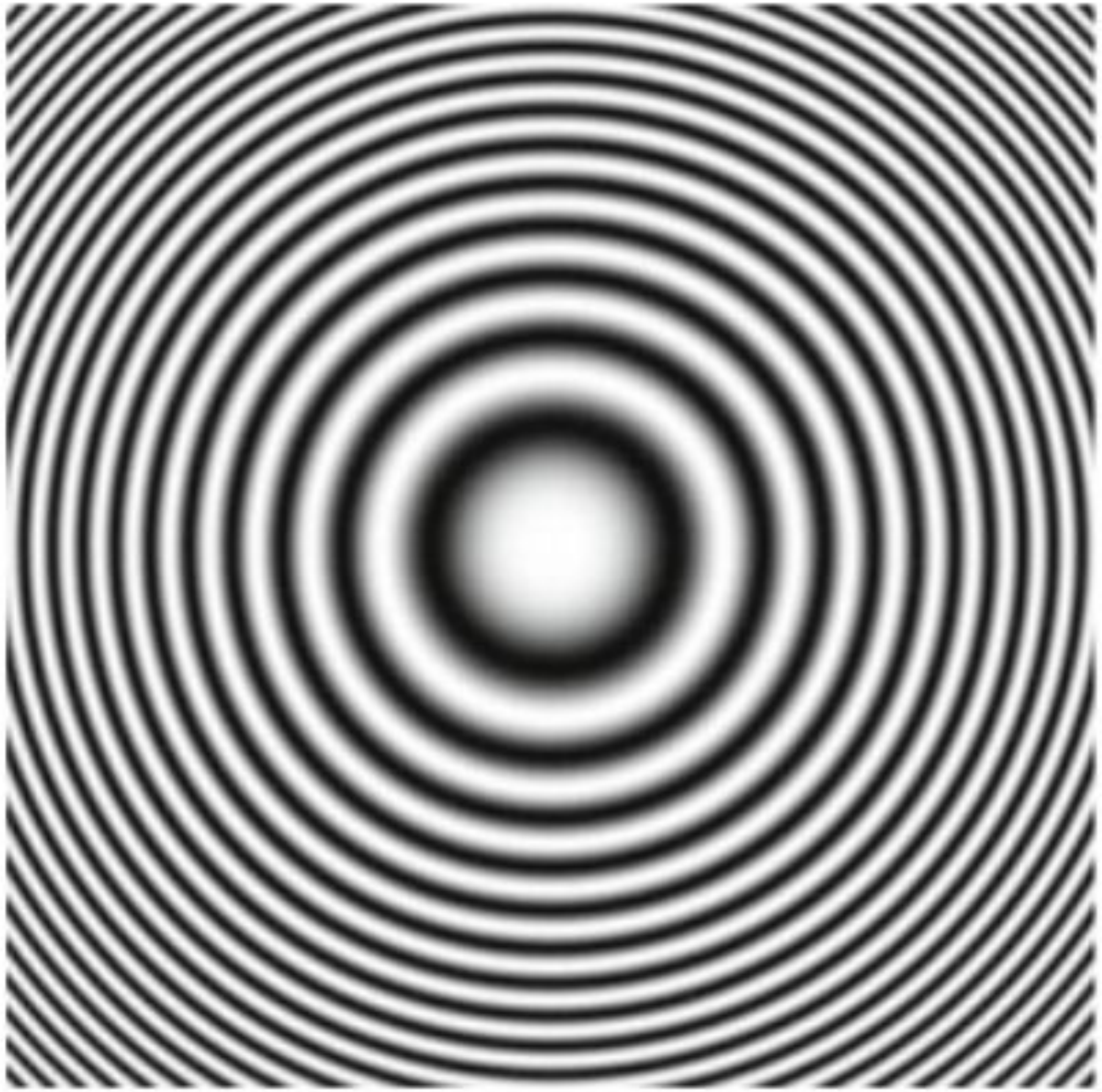


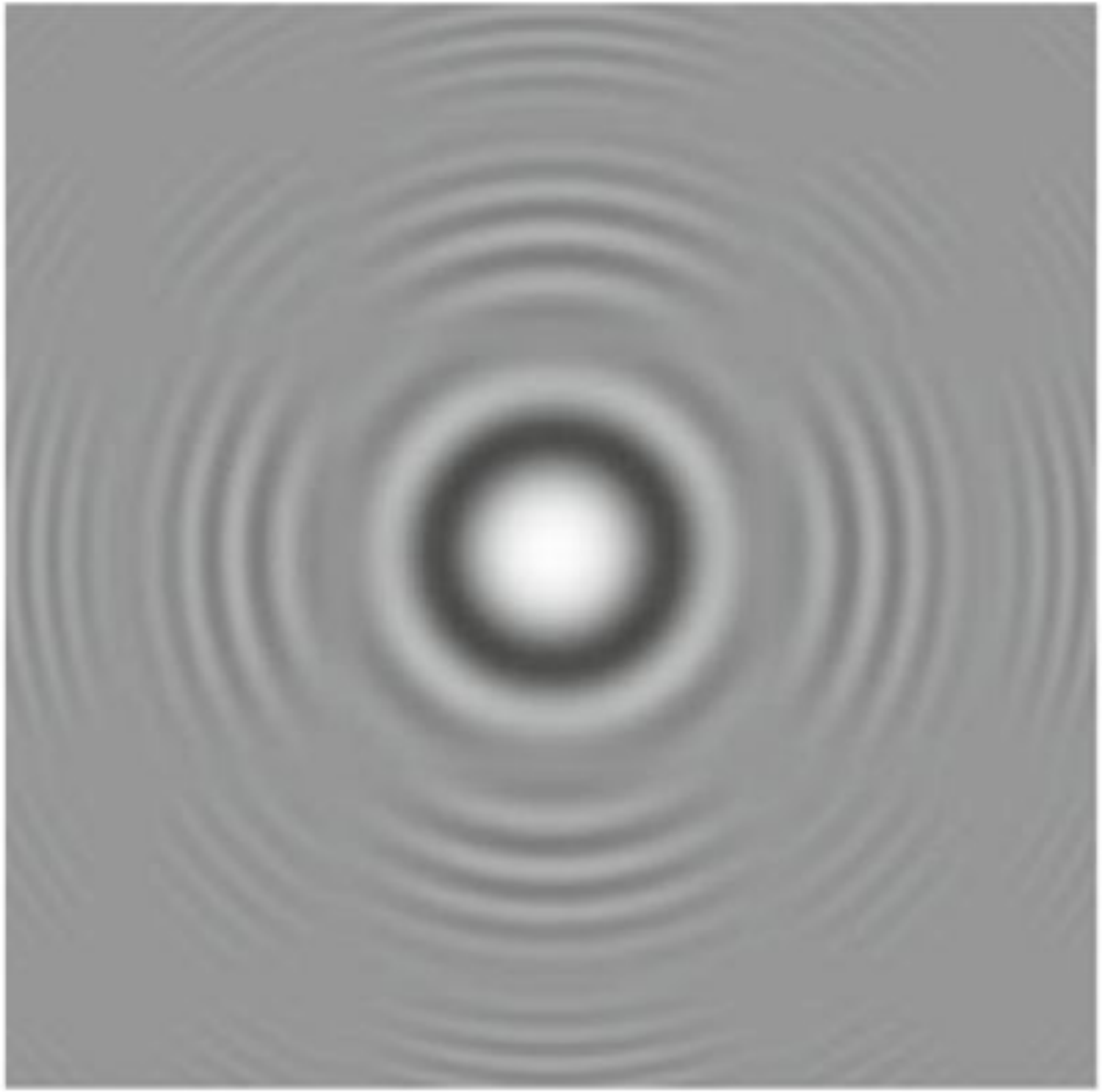
(b)

Box filter of order 3:

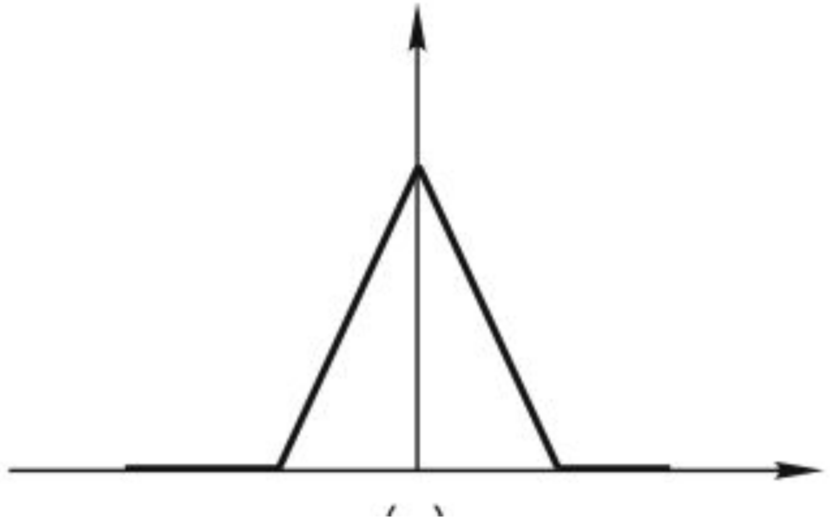
$$\frac{1}{9} \cdot \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

Discrete filters: examples - BOX FILTERS





Discrete filters: examples - BARTLETT FILTERS



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

$$\frac{1}{3}\cdot\begin{array}{|c|c|c|c|c|}\hline \frac{1}{3} & \frac{2}{3} & 1 & \frac{2}{3} & \frac{1}{3} \\ \hline\end{array}$$

$$\frac{1}{16}\cdot\begin{array}{|c|c|c|c|c|c|c|}\hline 1 & 2 & 3 & 4 & 3 & 2 & 1 \\ \hline\end{array}$$

- The Bartlett filter is separable.

$$h_2(x, y) = h_1(x) \cdot h_1(y).$$

$$\begin{array}{|c|}\hline 1 \\ \hline 2 \\ \hline 3 \\ \hline 2 \\ \hline 1 \\ \hline\end{array} \qquad \begin{array}{|c|c|c|c|c|}\hline 1 & 2 & 3 & 2 & 1 \\ \hline 2 & 4 & 6 & 4 & 2 \\ \hline 3 & 6 & 9 & 6 & 3 \\ \hline 2 & 4 & 6 & 4 & 2 \\ \hline 1 & 2 & 3 & 2 & 1 \\ \hline\end{array}$$

$$\begin{array}{|c|c|c|c|c|}\hline 1 & 2 & 3 & 2 & 1 \\ \hline\end{array}$$

Discrete filters: examples - BARTLETT FILTERS

Fig. 7.22. Top: Original image. Bottom: Image after applying a Bartlett filter of order 5.

Discrete filters: examples - GAUSSIAN FILTERS

in 1D $G_\sigma(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-x^2/(2\sigma^2)}$, in 2D $G_\sigma(x, y) = \frac{1}{2\sigma^2\pi} e^{-(x^2+y^2)/(2\sigma^2)}$.

- It is separable. $\rightarrow$ $G_\sigma(x, y) = G_\sigma(x)G_\sigma(y)$

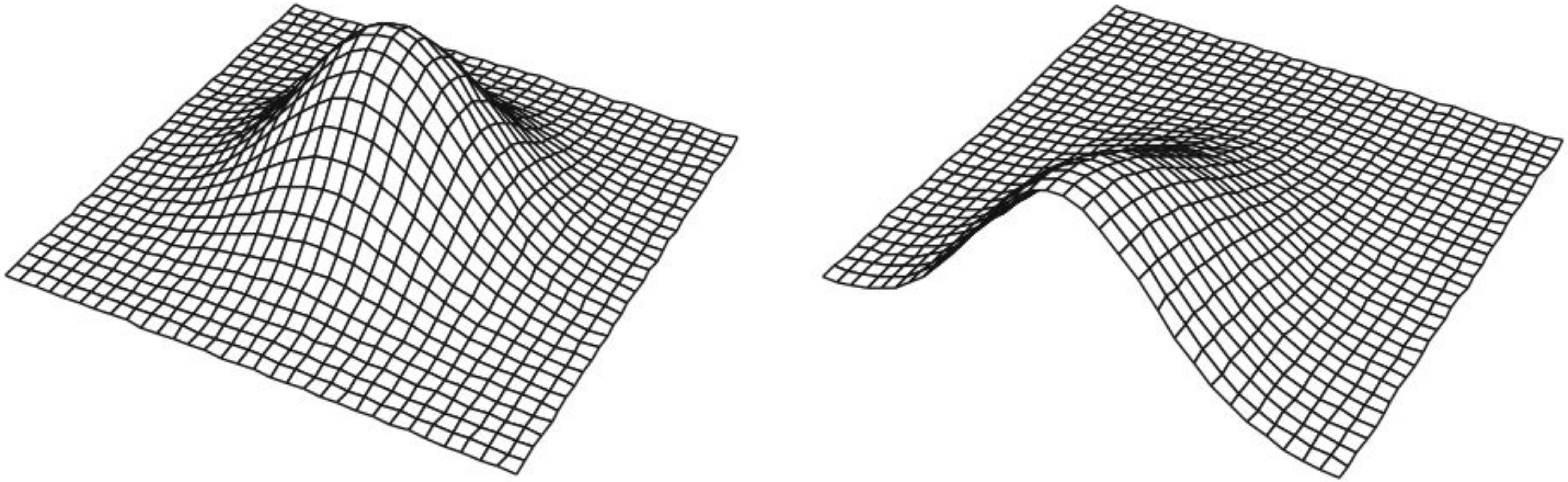


Fig. 7.23. Gaussian distribution function with mean 0 and variance 2.

Discrete filters: examples - GAUSSIAN FILTERS

- Gaussian's kernel can be approximated by using the coefficients in binomial expansions

n	2^n	mask coefficients
1	2	1 1
2	4	1 2 1
3	8	1 3 3 1
4	16	1 4 6 4 1
5	32	1 5 10 10 5 1
6	64	1 6 15 20 15 6 1
7	128	1 7 21 35 35 21 7 1
8	256	1 8 28 56 70 56 28 8 1

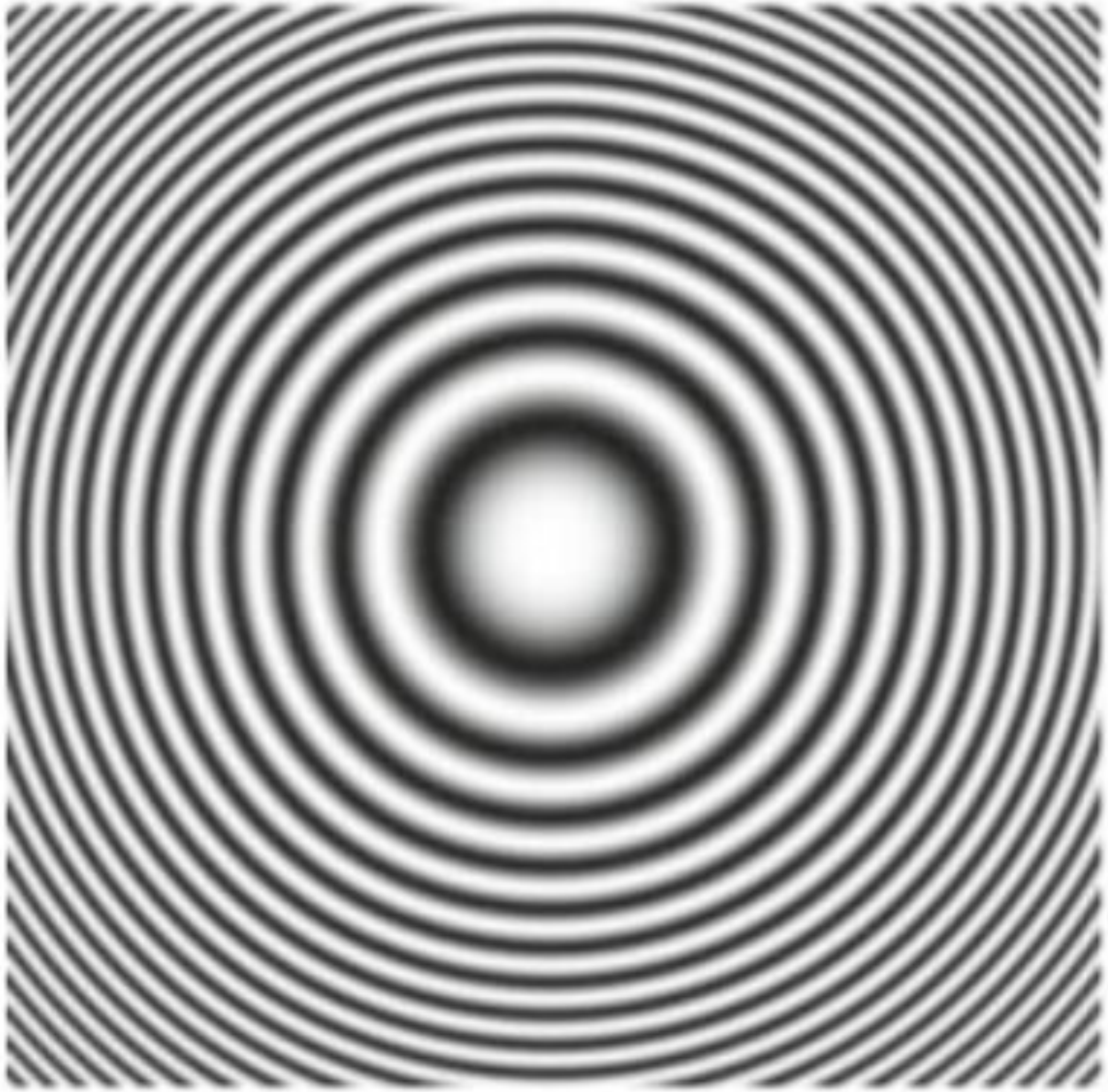
$$\frac{1}{4} \cdot \begin{array}{|c|c|}\hline 1 & 1 \\ \hline 1 & 1 \\ \hline\end{array}$$

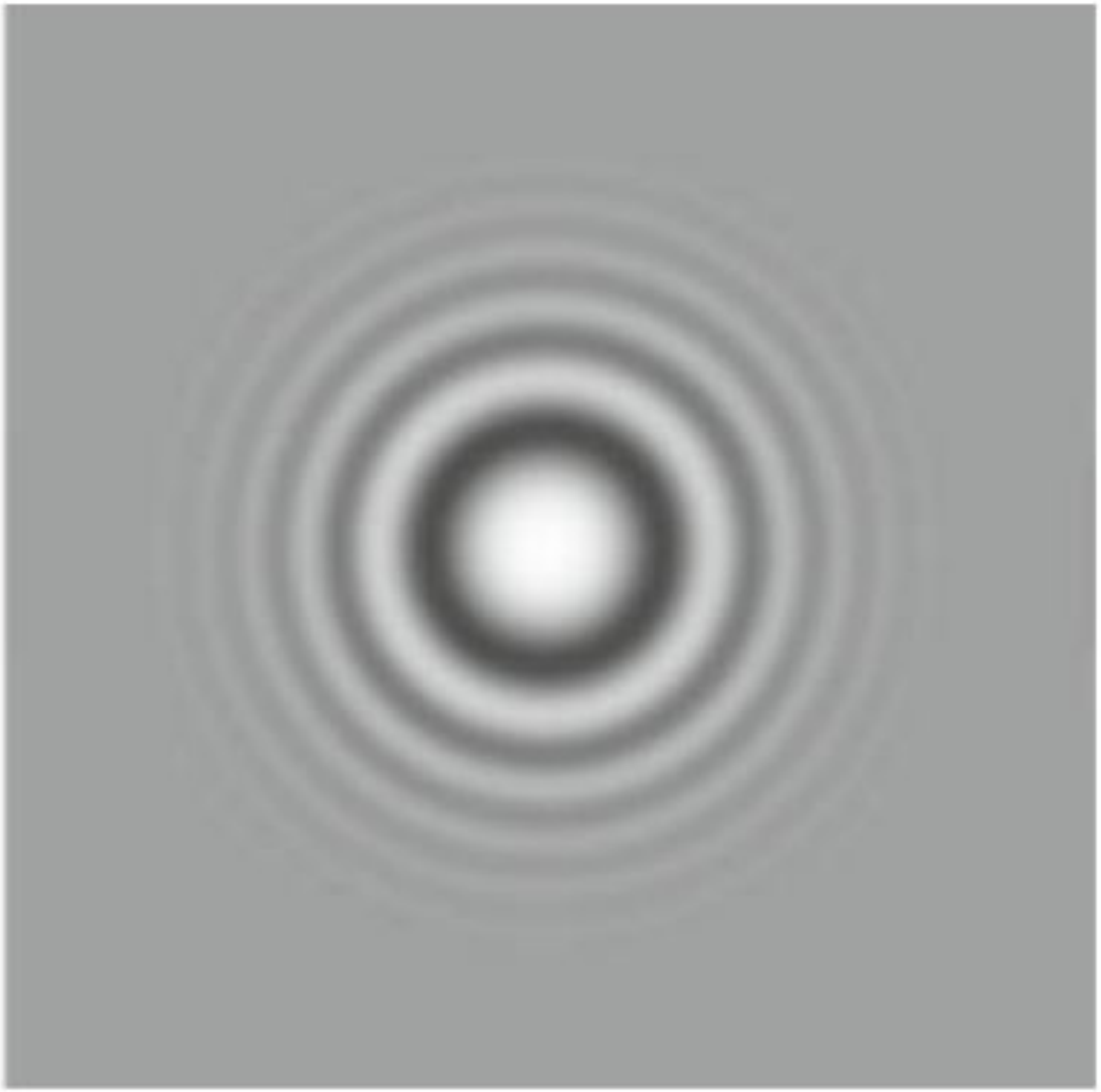
$$\frac{1}{16} \cdot \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline\end{array}$$

$$\frac{1}{64} \cdot \begin{array}{|c|c|c|c|}\hline 1 & 3 & 3 & 1 \\ \hline 3 & 9 & 9 & 3 \\ \hline 3 & 9 & 9 & 3 \\ \hline 1 & 3 & 3 & 1 \\ \hline\end{array}$$

$$\frac{1}{256} \cdot \begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 6 & 4 & 1 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 6 & 24 & 36 & 24 & 6 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 1 & 4 & 6 & 4 & 1 \\ \hline\end{array} \quad .$$

Discrete filters: examples - GAUSSIAN FILTERS

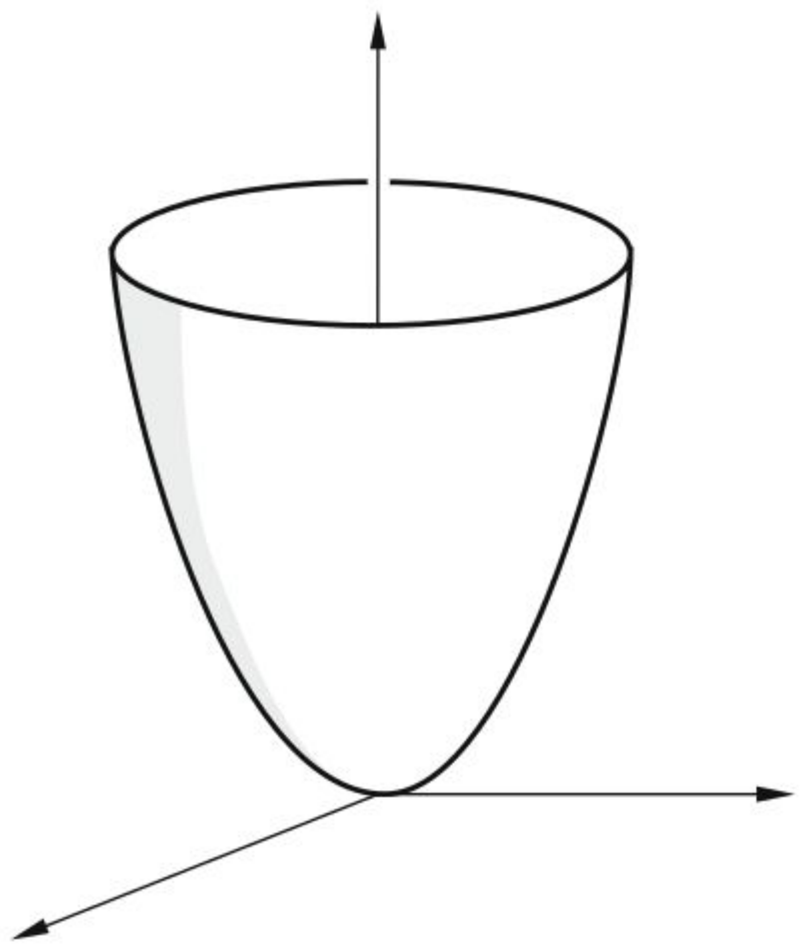




Discrete filters: examples - LAPLACIAN FILTERS

- $H(u, v) = -(2\pi)^2(u^2 + v^2).$

- It is a highpass filter



Discrete filters: examples - LAPLACIAN FILTERS

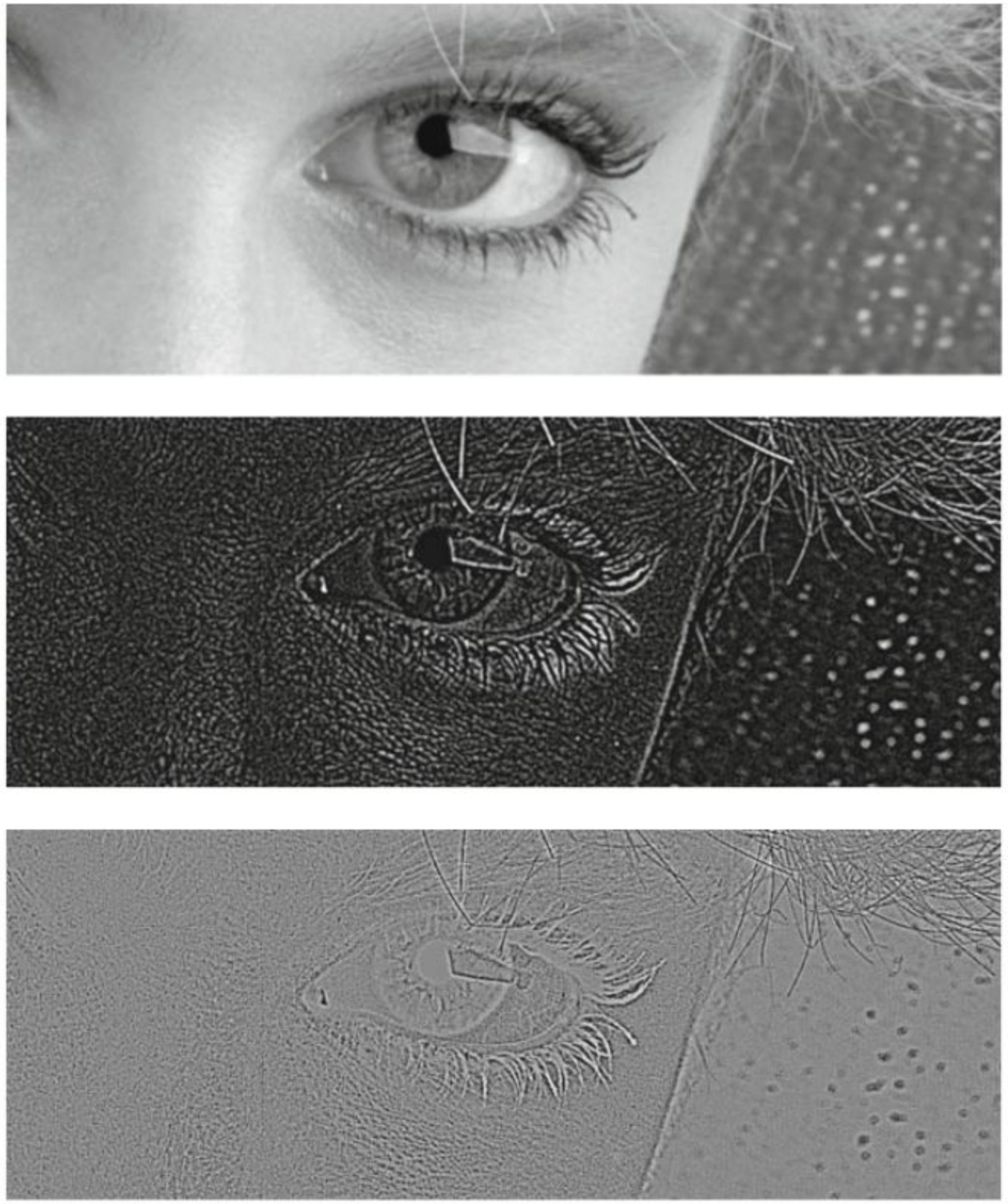
- Laplacian of orders 3 and 5:

$$\begin{array}{|c|c|c|} \hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array} \qquad \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 1 & 2 & -17 & 2 & 1 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

- Example of highpass filter based on a Laplacian:

$$\begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & -8 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} .$$

Discrete filters: examples - LAPLACIAN FILTERS



Discrete filters: Highpass filters

- Highpass filters can be obtained by subtracting the image to a lowpass filter.
- This is a general concept that allows us to create a lot of highpass filters.

  Example: Taking the Gaussian, we can obtain:

$$\frac{1}{16}\cdot\begin{array}{|c|c|c|}\hline 1&2&1\\\hline 2&4&2\\\hline 1&2&1\\\hline\end{array} \quad - \quad \begin{array}{|c|c|c|}\hline 0&0&0\\\hline 0&1&0\\\hline 0&0&0\\\hline\end{array} \quad = \quad \frac{1}{16}\cdot\begin{array}{|c|c|c|}\hline 1&2&1\\\hline 2&-12&2\\\hline 1&2&1\\\hline\end{array}\,.$$

Edge Enhacements Operations

- Laplacian addition

Fourier Transform

- Transforms the image representation into frequential domain instead of spatial domain.

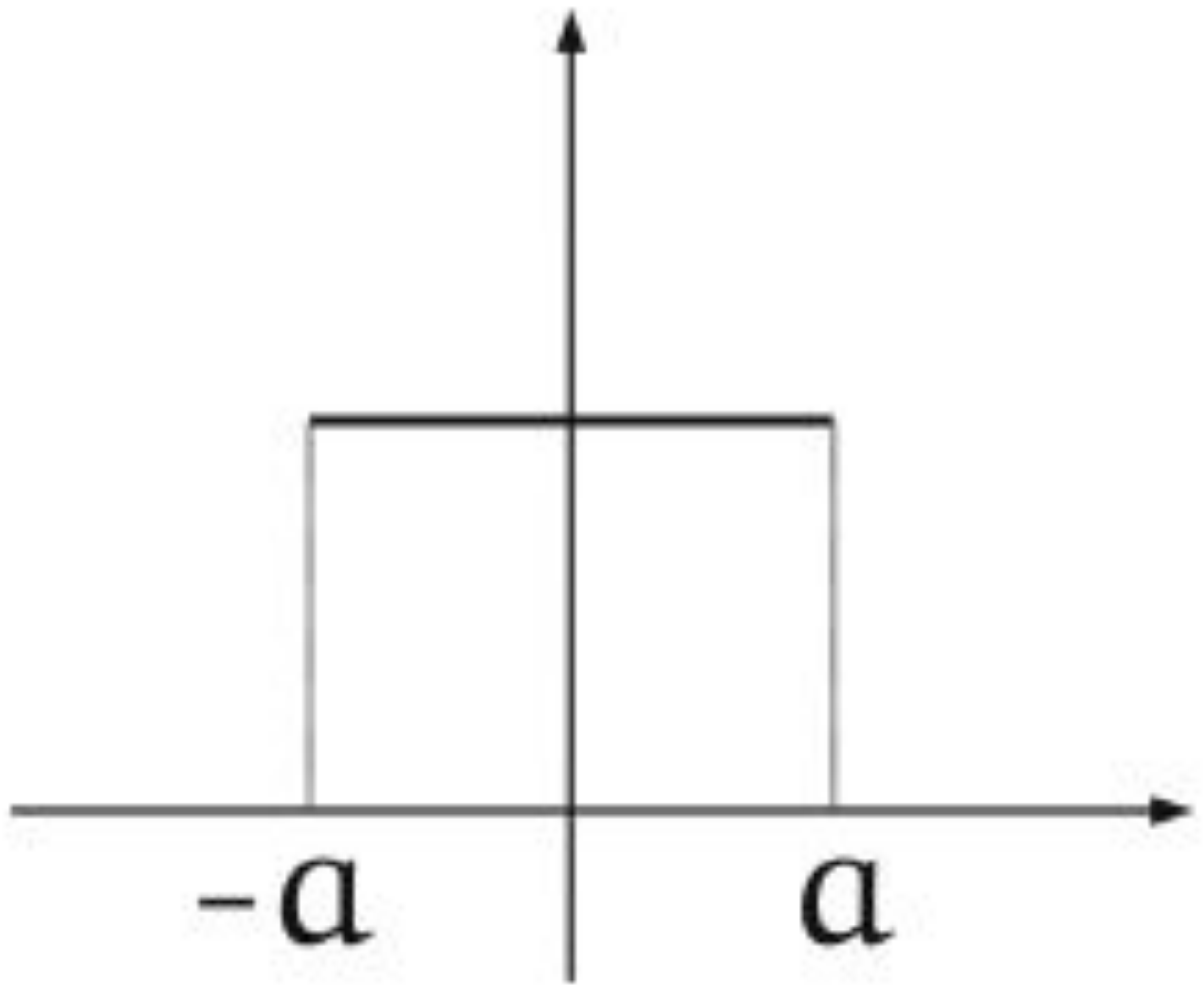
$$F(f)(s) = \hat{f}(s) = \int_{-\infty}^{+\infty} f(t)e^{-2\pi its}\,dt.$$

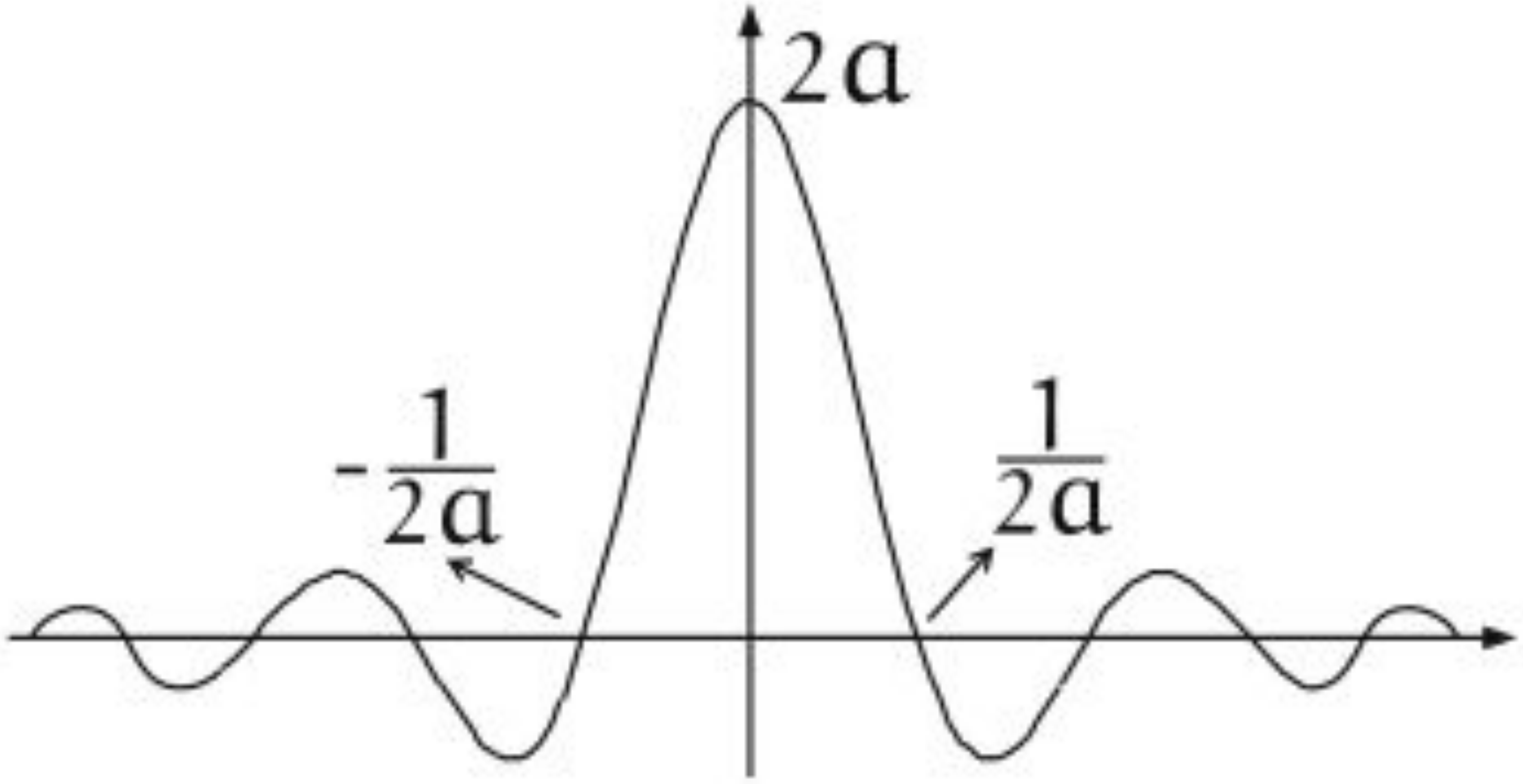
- and the inverse:

$$f(t) = F^{-1}(\hat{f}(s)) = \int_{-\infty}^{+\infty} \hat{f}(s)e^{2\pi ist}\,ds.$$

- **Convolution in the spatial domain → Product in the frequency domain**

Fourier Transform of the pulse





Fourier Transform - Filtering in the frequency domain

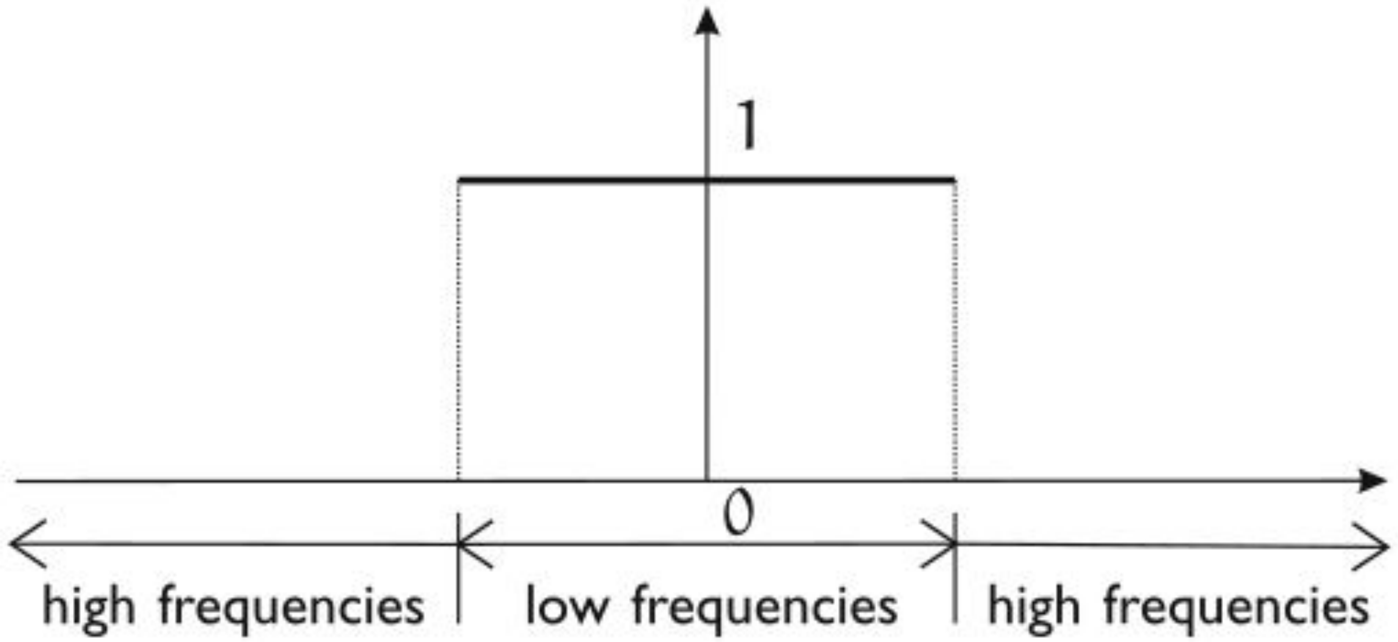


Fig. 2.15. One-dimensional ideal lowpass filter.

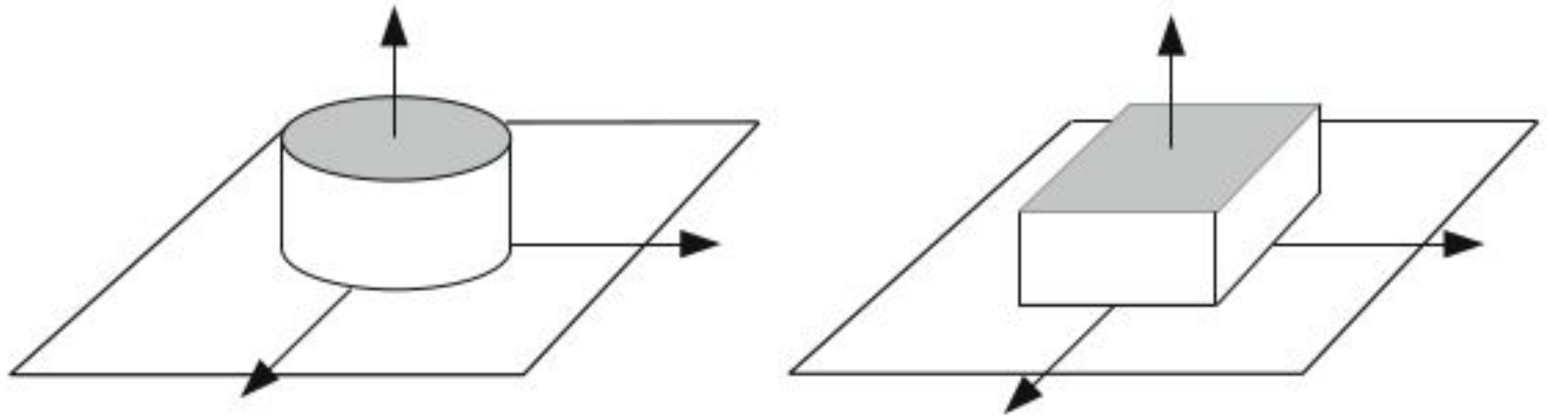


Fig. 2.16. Transfer function of ideal two-dimensional lowpass filters.

Fourier Transform - Filtering in the frequency domain

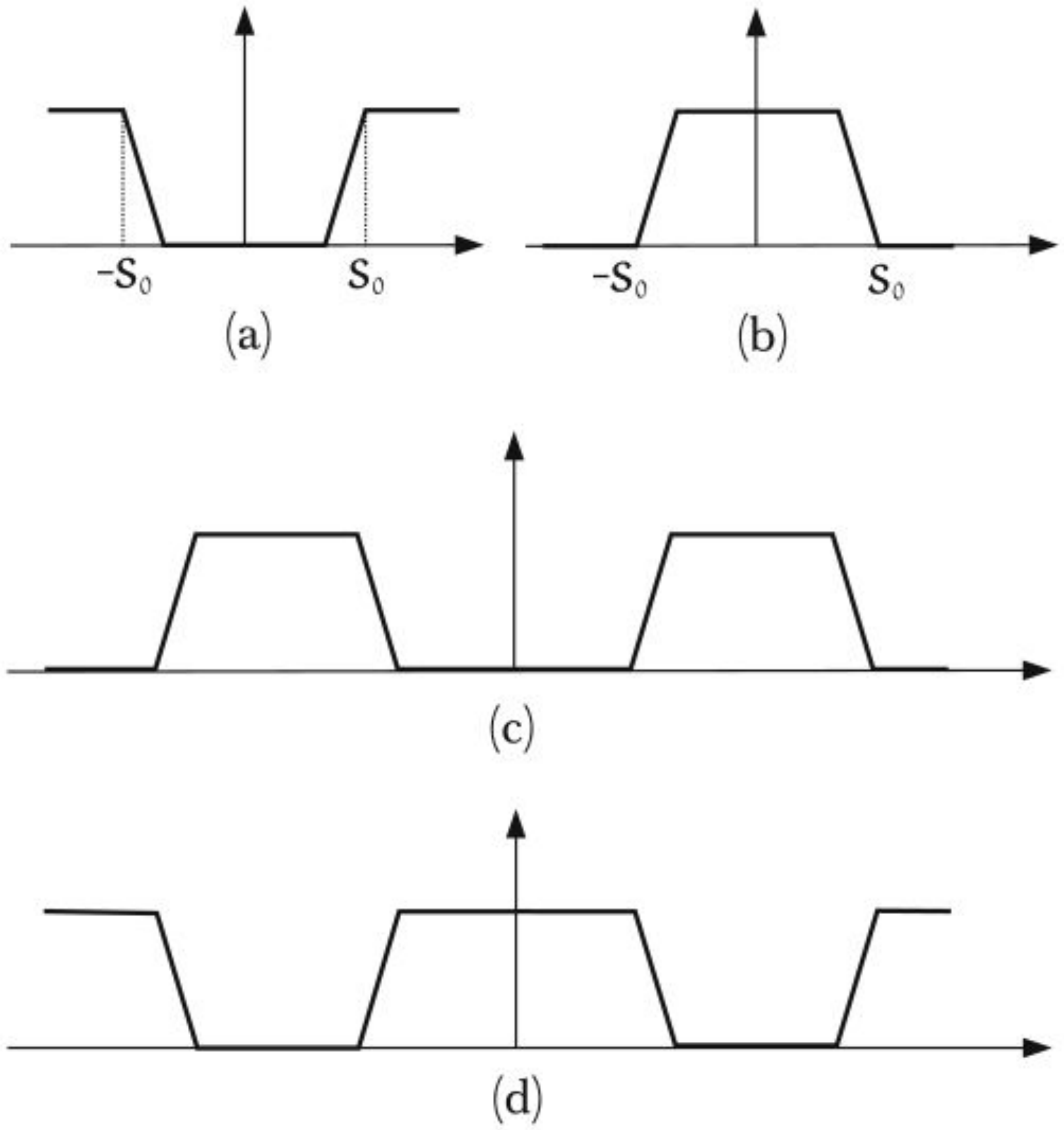


Fig. 2.14. Transfer function for filters: (a) highpass; (b) lowpass; (c) bandpass; (d) bandstop.

Fourier Transform - Practical considerations

- The Fourier Transform can be discretized. Its name is DFT (Discrete Fourier Transform)
- The computation of the DFT of a sequence of N points has complexity proportional to N^2. There are optimizations that reduce this complexity to N log N. This algorithm is known as Fast Fourier Transform (FFT)
- **It can help us to save time during filtering operations.**

Applications & State of the art.

Computational Photography: Gradient domain blending.

Applications & State of the art.
Computational Photography: Gradient domain blending.

Applications & State of the art.
Computational Photography: Gradient domain blending.

Applications & State of the art.

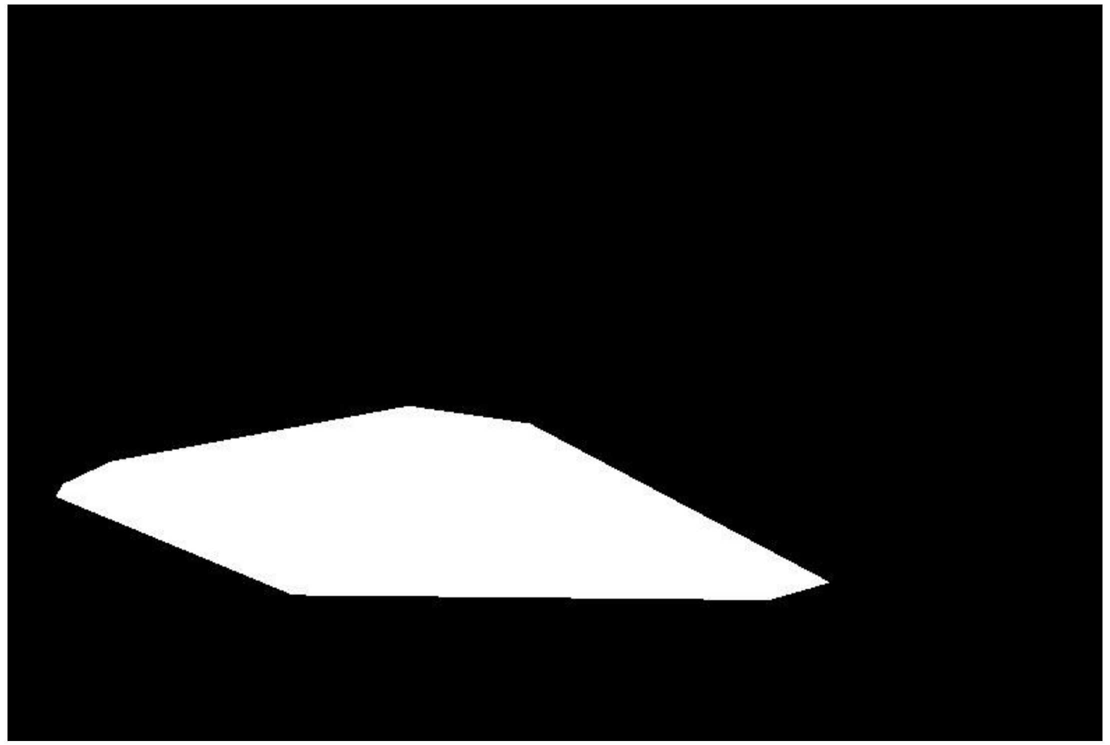
Computational Photography: Gradient domain blending.

Applications & State of the art.
Computational Photography: Gradient domain blending.




Applications & State of the art.

Computational Photography: Gradient domain blending.



Applications & State of the art.
Computational Photography: Gradient domain blending.

- Problems of hole filling, combining multiple blurred images to create an unblurred image, compositing multiple images when no a priori masks are known, etc., are at the heart of the field of **computational photography.**
    - Image-based rendering: synthesis of new views of a scene from one or more photographs or renderings of previous views.
        - What pixel values should I fill in for the parts of the scene that weren't visible in the previous view, but are in this one?
            - If it's a matter of just a pixel or two, filling in with colors from neighboring pixels is good enough to fool the eye, but for larger regions, hole filling is a serious (although obviously underdetermined) problem.