

Digital Image Processing and Pattern Recognition

E1528

Fall 2022-2023

Lecture 5



Lowpass - Smoothing Spatial Filters

INSTRUCTOR

DR / AYMAN SOLIMAN

➤ Contents

- Separable Filter Kernels
- Comparisons Between Filtering in the Spatial and Frequency Domains
- How Spatial Filter Kernels are Constructed
- Box Filter Kernels
- Lowpass Gaussian Filter Kernels



➤ Separable Filter Kernels

- As noted, a 2-D function $G(x,y)$ is said to be separable if it can be written as the product of two 1-D functions, $G_1(x)$ and $G_2(x)$; that is,

$$G(x,y) = G_1(x) G_2(x)$$

- A **spatial filter kernel** is a matrix, and a **separable kernel** is a matrix that can be expressed as the outer product of two vectors. For example, the 3*5 **kernel**

$$w = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

➤ Separable Filter Kernels

$$w = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

- is **separable** because it can be expressed as the outer product of the vectors

$$c = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \text{ and } r = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

- That is,

$$cr^T = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} [1 \quad 1 \quad 1 \quad 1 \quad 1] = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = w$$

➤ Separable Filter Kernels

- A **separable kernel** of size $m \times n$ can be expressed as the outer product of two vectors, v and w :

$$w = VW^T$$

Where v and w are vectors of size $m \times 1$ and $n \times 1$, respectively.

- For a **square** kernel of size $m \times m$, we write

$$w = VV^T$$

- It turns out that the product of a column vector and a row vector is the same as the 2-D convolution of the vectors

➤ Separable Filter Kernels

- The **importance** of separable kernels **lies** in the computational advantages that result from the **associative** property of **convolution**.
- If we have a kernel w that can be decomposed into two simpler kernels, such that $w = w_1 * w_2$, then it follows from the **commutative** and **associative** properties that

$$w * f = (w_1 * w_2) * f = (w_2 * w_1) * f = w_2 * (w_1 * f) = (w_1 * f) * w_2$$

- This equation says that **convolving a separable kernel** with an image is the same as convolving w_1 with f **first**, and **then** convolving the result with w_2 .

➤ Separable Filter Kernels

➤ For an image of size $M \times N$ and a kernel of size $m \times n$, implementation of

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

requires on the order of **MNmn** multiplications and additions. This is because it follows **directly** from that equation that each pixel in the output (filtered) image depends on all the coefficients in the filter kernel.

But if the kernel is separable and we use Eq.

$$w * f = (w_1 * w_2) * f = (w_2 * w_1) * f = w_2 * (w_1 * f) = (w_1 * f) * w_2$$

then the first convolution, $w_1 * f$, requires on the order of MNm multiplications and additions because w_1 is of size $m \times 1$.

➤ Separable Filter Kernels

- The result is of size $M \times N$, so the convolution of w_2 with the result requires MNn such operations, for a total of $MN(m+n)$ multiplication and addition operations.
- Thus, the computational advantage of performing convolution with a **separable**, as opposed to a **non-separable**, kernel is defined as

$$C = \frac{MN(m+n)}{MNmn} = \frac{m+n}{mn}$$

- For kernels with **hundreds** of elements, **execution times** can be **reduced** by a factor of a **hundred or more**, which is significant.

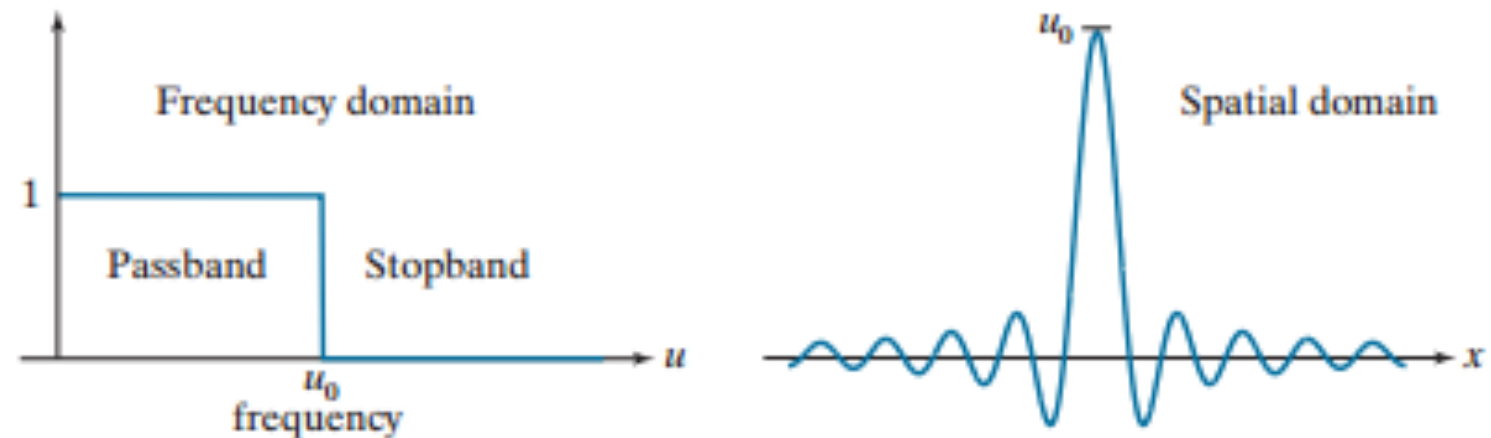
➤ Some Important Comparisons Between Filtering in the Spatial and Frequency Domains

- The tie between spatial and frequency domain processing is the Fourier transform.
- We use the Fourier transform to go from the spatial to the frequency domain; to return to the spatial domain we use the inverse Fourier transform.

a **b**

(a) Ideal 1-D lowpass filter transfer function in the frequency domain.

(b) Corresponding filter kernel in the spatial domain.



➤ **Some Important Comparisons Between Filtering in the Spatial and Frequency Domains**

➤ The focus here is on **two fundamental properties** relating the spatial and frequency domains:

1. **Convolution**, which is the basis for filtering in the spatial domain, is equivalent to multiplication in the frequency domain, and vice versa.
2. **An impulse of strength A** in the spatial domain is a constant of value A in the frequency domain, and vice versa.

➤ How Spatial Filter Kernels are Constructed

- We consider **Three** basic approaches for constructing spatial filters.
- **1- The first approach** is based on **formulating filters based on mathematical properties.**
- For example, a **filter** that computes the **average of pixels** in a neighborhood **blurs an image.** Computing an average is similar to integration.
- Conversely, a filter that computes the **local derivative** of an Image **sharpens the image.**

➤ How Spatial Filter Kernels are Constructed

- 2- The second approach is based on **sampling** a 2-D spatial function whose **shape** has a **desired property**.
- For example, we will later show in the samples from **a Gaussian function** can be used to construct **a weighted-average (lowpass) filter**.
- These **2-D spatial functions** sometimes are generated as the **inverse Fourier** transform of 2-D filters specified in the frequency domain.

➤ How Spatial Filter Kernels are Constructed

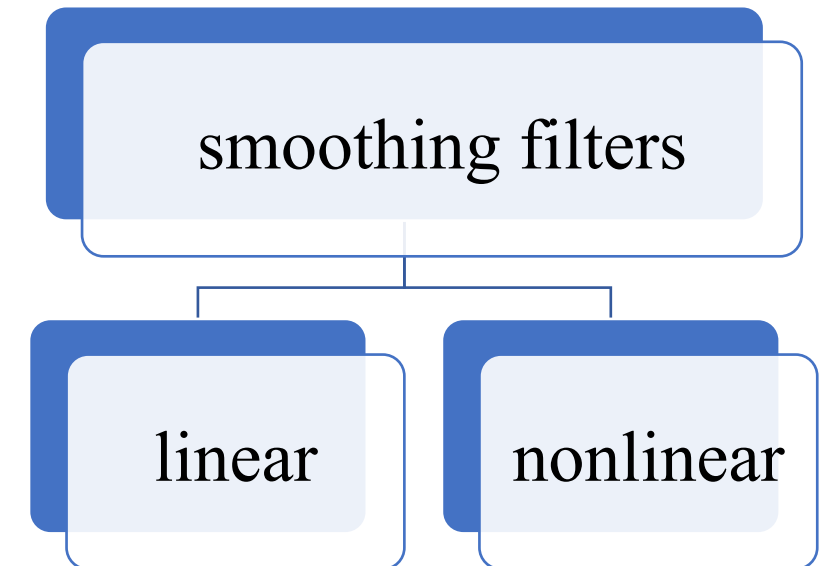
- 3- The third approach is to design a spatial filter with a **specified frequency response**.
- This approach is fallen in the area of **digital filter design**.
- A **1-D spatial filter** with the desired response is obtained (typically using filter design software).
- The **1-D filter** values can be expressed as a **vector v** , and a 2-D separable kernel can then be obtained using the equation $w = VV^T$. Or the 1-D filter can be rotated about its center to generate a 2-D kernel that approximates a circularly symmetric function.

➤ **Smoothing (Lowpass) Spatial Filters**

- Smoothing (also called **averaging**) spatial filters are used to **reduce sharp transitions in intensity**. Because random noise typically consists of sharp transitions in intensity, an obvious **application** of smoothing is **noise reduction**.
- Smoothing prior to image resampling to **reduce aliasing**, is also a common application.
- **Smoothing** is used to **reduce irrelevant detail** in an image, where “irrelevant” refers to pixel regions that are small with respect to the size of the filter kernel.

➤ Smoothing (Lowpass) Spatial Filters

- **Smoothing filters** are used in **combination** with **other techniques** for image enhancement, such as the **histogram** processing techniques, and **unsharp masking**, as discussed later.
- We begin the discussion of smoothing filters by considering **linear** smoothing filters in some detail.
- We will introduce **nonlinear** smoothing filters **later**.



➤ **Smoothing (Lowpass) Spatial Filters**

- As we discussed, **linear spatial filtering** consists of convolving an image with a **filter kernel**.
- Convolution with a smoothing kernel **blurs** the image, with the degree of **blurring being determined by the size of the kernel** and the **values of its coefficients**.
- In addition to being useful in countless applications of image processing, **lowpass filters are fundamental**, in the sense that other important filters, including **sharpening (high-pass)**, **bandpass**, and **band-reject filters**, can be derived from **lowpass filters**.

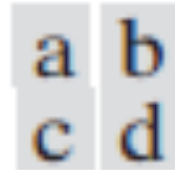
➤ **Smoothing (Lowpass) Spatial Filters**

- We discuss in this section **lowpass filters** based on **box** and **Gaussian** kernels, both of which are **separable**.
- Most of the discussion will center on **Gaussian** kernels because of their **numerous useful properties** and **extensiveness of applicability**.

➤ **Box Filter Kernels**

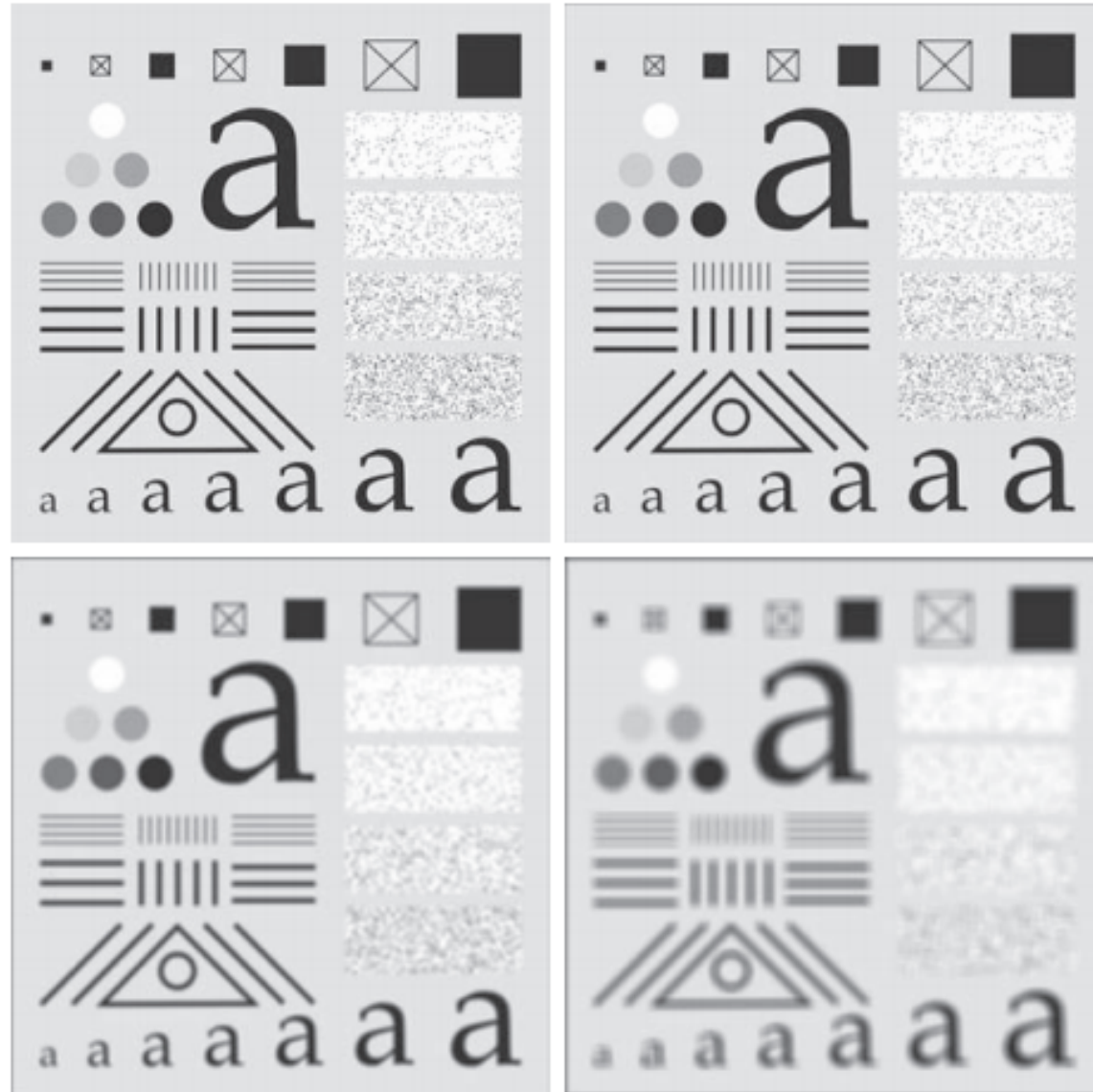
- The simplest, **separable lowpass filter** kernel is the **box** kernel, whose coefficients **have the same value (typically 1)**.
- The name “**box kernel**” comes from a constant kernel resembling a box when viewed in 3-D.
- We showed a 3×3 box filter in next Figure. An $m \times n$ box filter is an $m \times n$ array of 1's, with a normalizing constant in front, whose value is 1 divided by the sum of the values of the coefficients (i.e., $1/mn$ when all the coefficients are 1's).

➤ Box Filter Kernels



(a) Test pattern of size 1024×1024 pixels.

(b)-(d) Results of lowpass filtering with box kernels of sizes 3×3 , 11×11 , and 21×21 , respectively.



➤ **Box Filter Kernels**

- Figure(a) shows a test pattern image of size 1024×1024 pixels.
- Figures (b)-(d) are the results obtained using box filters of size $m \times m$ with $m = 3, 11,$ and 21 respectively. For $m = 3$, we note a **slight overall blurring** of the image, with the image features whose sizes are comparable to the size of the kernel being affected significantly more.
- Such features include the thinner lines in the image and the noise pixels contained in the boxes on the right side of the image. The **filtered image** also has a **thin gray border**, the **result of zero-padding** the image prior to filtering.

➤ **Box Filter Kernels**

- As indicated earlier, **padding** extends the **boundaries** of an image **to avoid undefined operations when parts of a kernel lie outside the border** of the image during filtering.
- When **zero (black)** padding is used, the net result of smoothing at or **near** the border is a **dark gray** border that arises from including black pixels in the averaging process.
- Using the 11×11 kernel resulted in more pronounced blurring throughout the image, including a more prominent dark border.

➤ **Box Filter Kernels**

- The result with the 21×21 kernel **shows significant blurring** of all components of the image, including the **loss of the characteristic shape** of some components, including, for example, the small square on the top left and the small character on the bottom left.
- The **dark border** resulting from zero padding is proportionally **thicker than before**.
- We used zero padding here, and will use it a few more times, so that you can become familiar with its effects.

➤ **Box Filter Kernels important**

- Because of **their simplicity**, **box** filters are suitable for **quick experimentation**, and they often yield **smoothing** results that are visually **acceptable**.
- They are useful also when it is desired **to reduce the effect of smoothing on edges**.
- However, box filters have **limitations** that make them **poor choices** in many applications. For example, a defocused lens is often modeled as a lowpass filter, but box filters are **poor approximations to the blurring characteristics of lenses**.

➤ **Box Filter Kernel limitations**

- Another **limitation** is the fact that box filters **favor blurring along perpendicular directions**.
- In applications **involving images with a high level of detail**, or with **strong geometrical components**, the directionality of box filters often **produces undesirable results**.

➤ Lowpass Gaussian Filter Kernels

- The **kernels** of choice in applications such as those just mentioned are **circularly symmetric** (also called **isotropic**, meaning their response is independent of orientation). As it turns out, Gaussian kernels of the form

$$w(s, t) = G(s, t) = K e^{-\frac{s^2+t^2}{2\sigma^2}}$$

are the only circularly symmetric kernels that are also separable.

Thus, because Gaussian kernels of this form are **separable**, Gaussian filters enjoy the same computational advantages as box filters but **have a host of additional properties** that make them ideal for image processing.

➤ Lowpass Gaussian Filter Kernels

➤ Variables s and t in last Eq. , are real (**typically discrete**) numbers.

➤ By letting $r = \sqrt{s^2 + t^2}$ we can write last equation as

$$G(r) = K e^{-\frac{r^2}{2\sigma^2}}$$

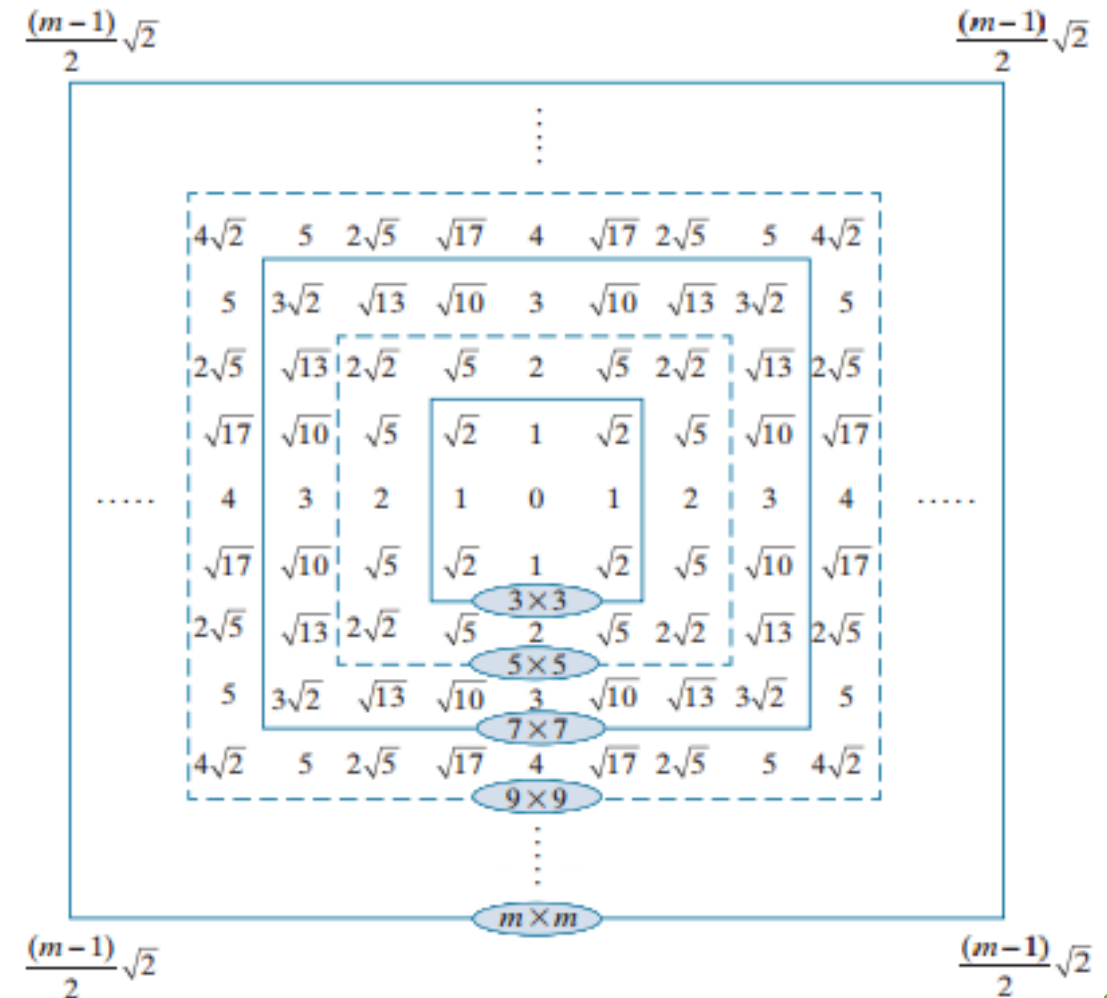
➤ This equivalent form simplifies derivation of expressions later in this lecture.

➤ This form also reminds us that the function is **circularly symmetric**.

Variable r is the distance from the center to any point on function G .

➤ Lowpass Gaussian Filter Kernels

- Figure shows values of r for several kernel sizes using integer values for s and t .
- Because we work generally with **odd** kernel sizes, the centers of such kernels fall on integer values, and it follows that all values of r^2 are integers also.



➤ Lowpass Gaussian Filter Kernels



(a) A test pattern of size 1024×1024 . (b) Result of lowpass filtering the pattern with a Gaussian kernel of size 21×21 , with standard deviations $\sigma = 3.5$. (c) Result of using a kernel of size 43×43 , with $\sigma = 7$. We used $K = 1$ in all cases.

➤ Lowpass Gaussian Filter Kernels



Result of filtering the test pattern in Fig. using (a) zero padding, (b) mirror padding, and (c) replicate padding. A Gaussian kernel of size 187×187 , with $K = 1$ and $\sigma = 31$ was used in all three cases.

➤ **Smoothing performance as a function of kernel and image size.**



a b c

(a) Test pattern of size 4096×4096 pixels. (b) Result of filtering the test pattern with the same Gaussian kernel used in last Fig. (c) Result of filtering the pattern using a Gaussian kernel of size 745×745 elements, with $K = 1$ and $\sigma = 124$.

Mirror padding was used throughout.

*Thank
you*

