Preclass 14: Image Convolution

[SCS4049] Machine Learning and Data Science

Seongsik Park (s.park@dgu.edu)

AI Department, Dongguk University

Linear filter and convolution

Given a filter kernel $\mathcal{H}$, the convolution of the kernel with image $\mathcal{F}$ is an image $\mathcal{R}$. The (i, j)-th component of $\mathcal{R}$ is given by

$$R_{ij} = \sum_{u,v} H_{i-u,j-v} F_{uv}. \tag{1}$$

- **kernel** of the filter: the pattern of weights used for a linear filter
- **convolution**: the process of applying the filter

Linear filter and convolution

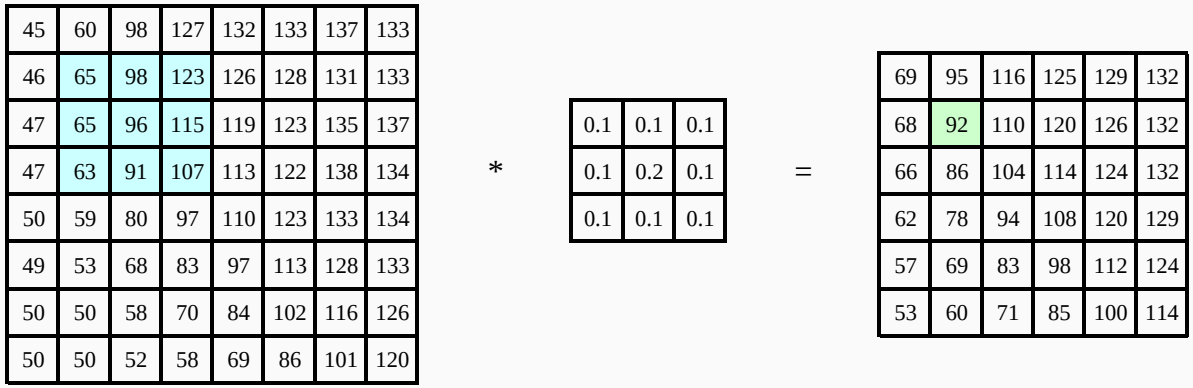


Figure 3.10 *Neighborhood filtering (convolution): The image on the left is convolved with the filter in the middle to yield the image on the right. The light blue pixels indicate the source neighborhood for the light green destination pixel.*

Linear filter and convolution

This operation is called **convolution**

$$R(f) = (h * f) \tag{2}$$

- *commutative*: $(g * h)(x) = (h * g)(x)$
- *associative*: $f * (g * h) = (f * g) * h$
- *distributive*: $f * (g + h) = f * g + f * h$

Linear filter and convolution: Gaussian blur

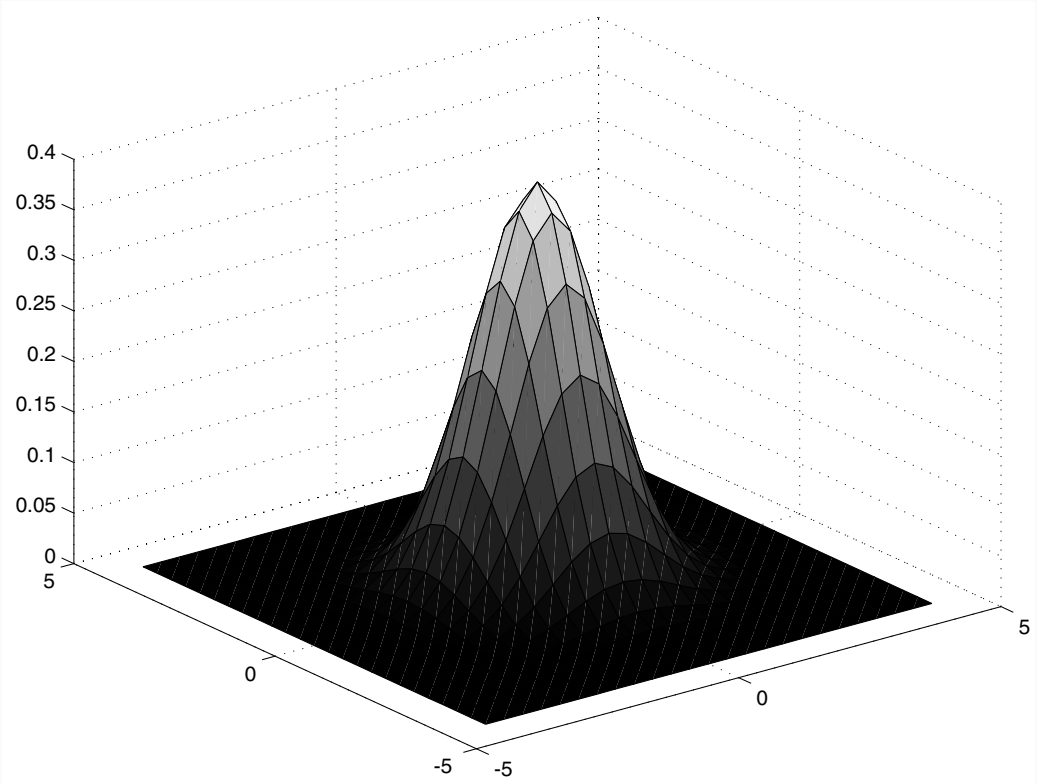


FIGURE 4.2: The symmetric Gaussian kernel in 2D. This view shows a kernel scaled so that its sum is equal to one; this scaling is quite often omitted. The kernel shown has $\sigma = 1$. Convolution with this kernel forms a weighted average that stresses the point at the center of the convolution window and incorporates little contribution from those at the boundary. Notice how the Gaussian is qualitatively similar to our description of the point spread function of image blur: it is circularly symmetric, has strongest response in the center, and dies away near the boundaries.

Linear filter and convolution: Gaussian kernel in 2D

Symmetric Gaussian kernel in 2D (noise reduction smoothing)

$$G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \tag{3}$$

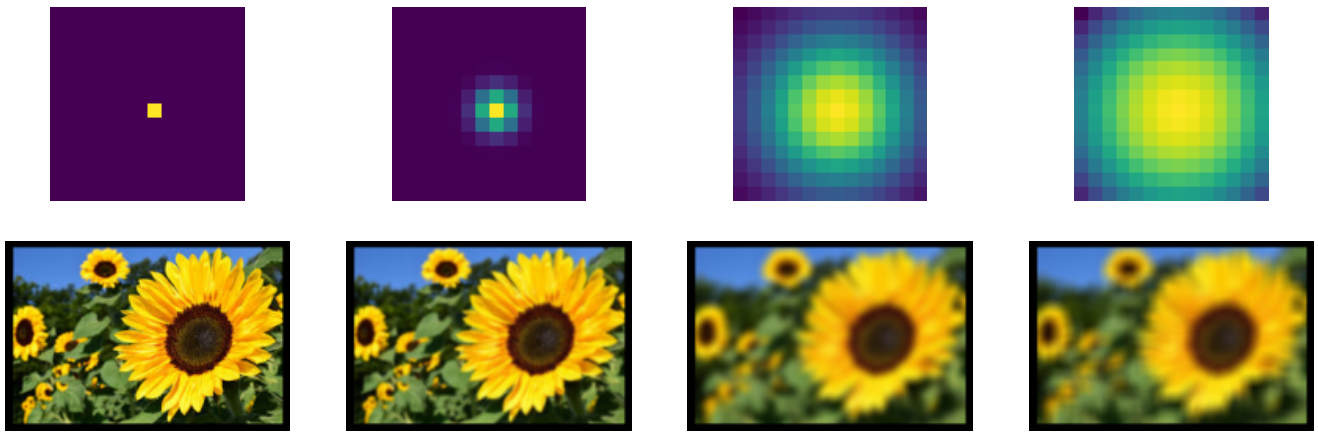


Figure 1: Gaussian blur with σ = [0.1, 1, 4, 10], no padding.

Padding (border effects)

- **zero**: set all pixels outside the source image to 0
- **constant**: set all pixels outside the source image to a specified border value
- **clamp**: repeat edge pixels indefinitely
- **wrap**: loop “around” the image in a “toroidal” configuration
- **mirror**: reflect pixels across the image edge
- **extend**: extend the signal by subtracting the mirrored version of the signal from the edge pixel value

Padding (border effects)

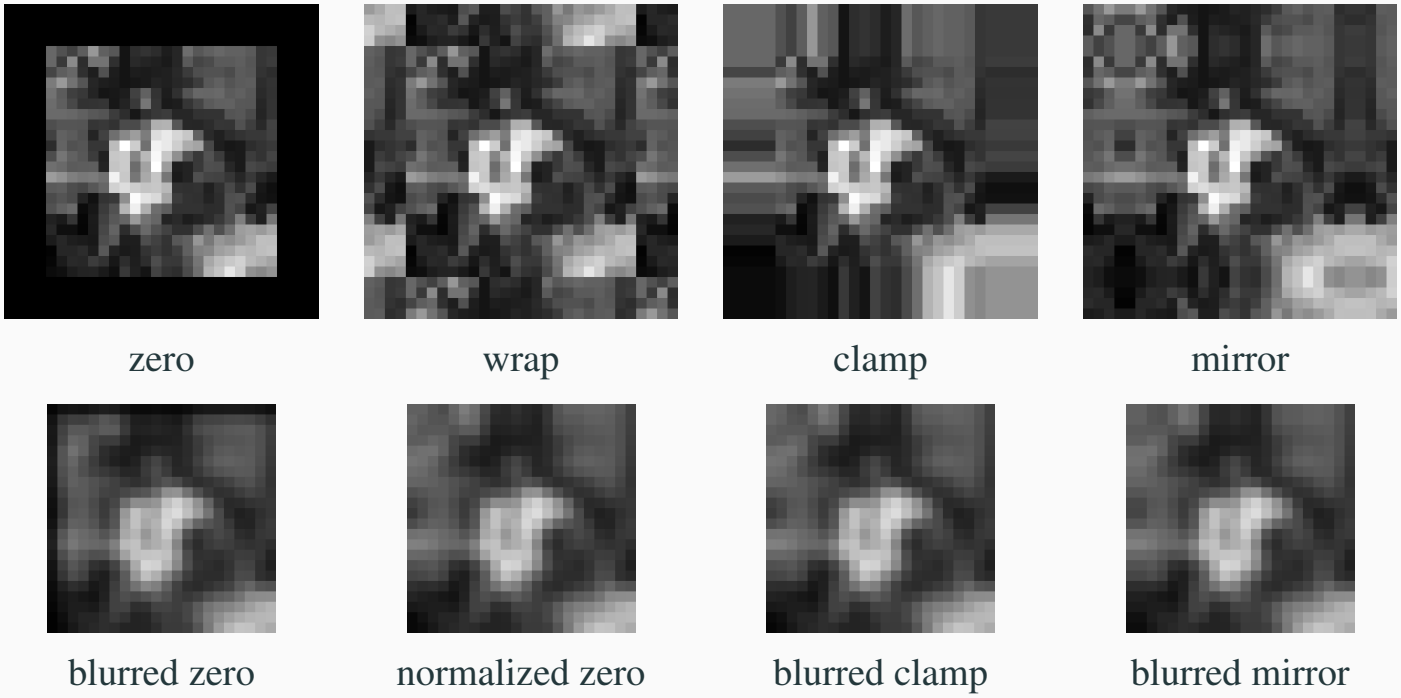
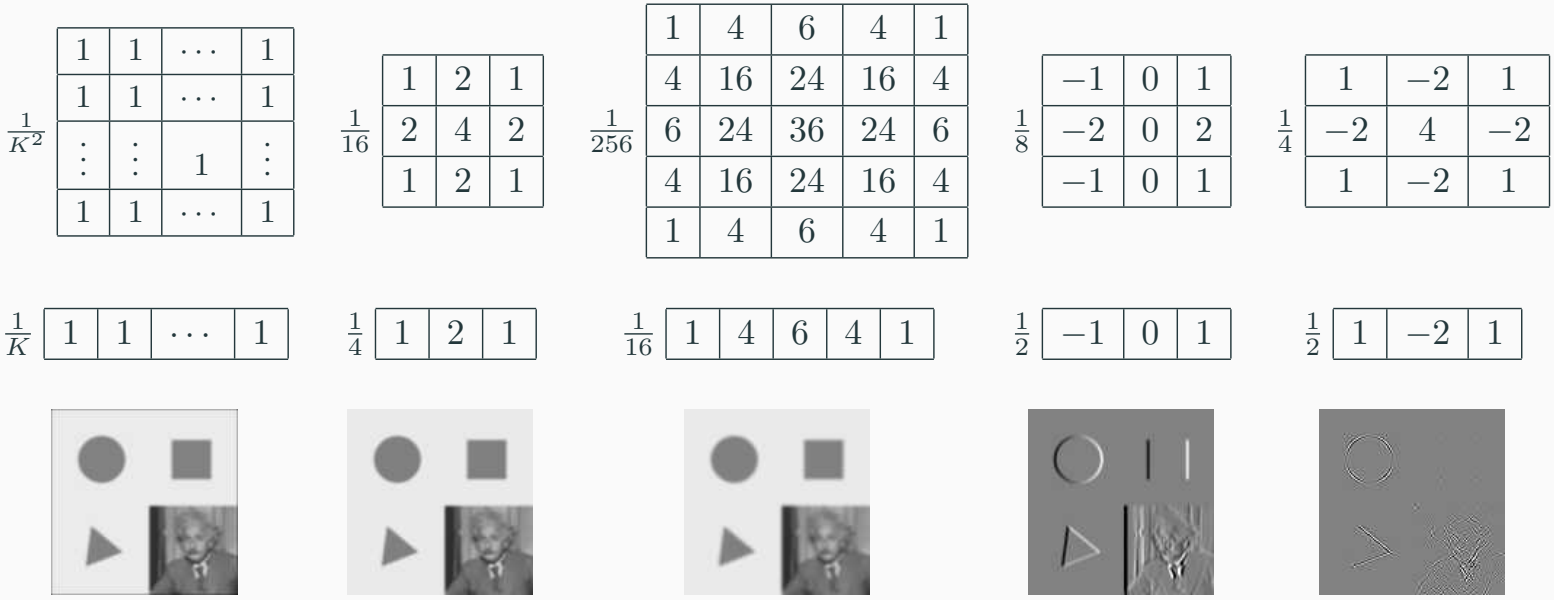


Figure 3.13 *Border padding (top row) and the results of blurring the padded image (bottom row). The normalized zero image is the result of dividing (normalizing) the blurred zero-padded RGBA image by its corresponding soft alpha value.*

Examples of linear filter



(a) box, $K = 5$ (b) bilinear (c) "Gaussian" (d) Sobel (e) corner

Figure 3.14 *Separable linear filters: For each image (a)–(e), we show the 2D filter kernel (top), the corresponding horizontal 1D kernel (middle), and the filtered image (bottom). The filtered Sobel and corner images are signed, scaled up by* $2\times$ *and* $4\times$*, respectively, and added to a gray offset before display.*