Computer vision: image filtering

Computer Vision: Fundamentals II

Explains

- Linear filters
- Image Gradients
- Edge detection

Prerequisites

- Matrix operations
- Coordinate systems
- Reference Frames
- Transformations

Credits

- Matthew R. Walter (TTIC) - September 2017
- Liam Paull (UdeM) - September 2018

Some slides adapted from Ayan Chakrabarti and Lana Lazebnik

These slides are part of the Duckietown project.
For more information about Duckietown, see the website http://duckietown.org

Filtering

We want to remove unwanted sources of variation, and keep the information relevant for whatever task we need to solve

Linear filtering

$$I[n_x, n_y] \longrightarrow \boxed{h[k,l]} \longrightarrow Y[n_x, n_y]$$

For a linear system, each output is a linear combination of all the input values:

$$Y[m,n] = \sum_{k,l} h[k,l] I[m-k, n-l]$$

In matrix form:

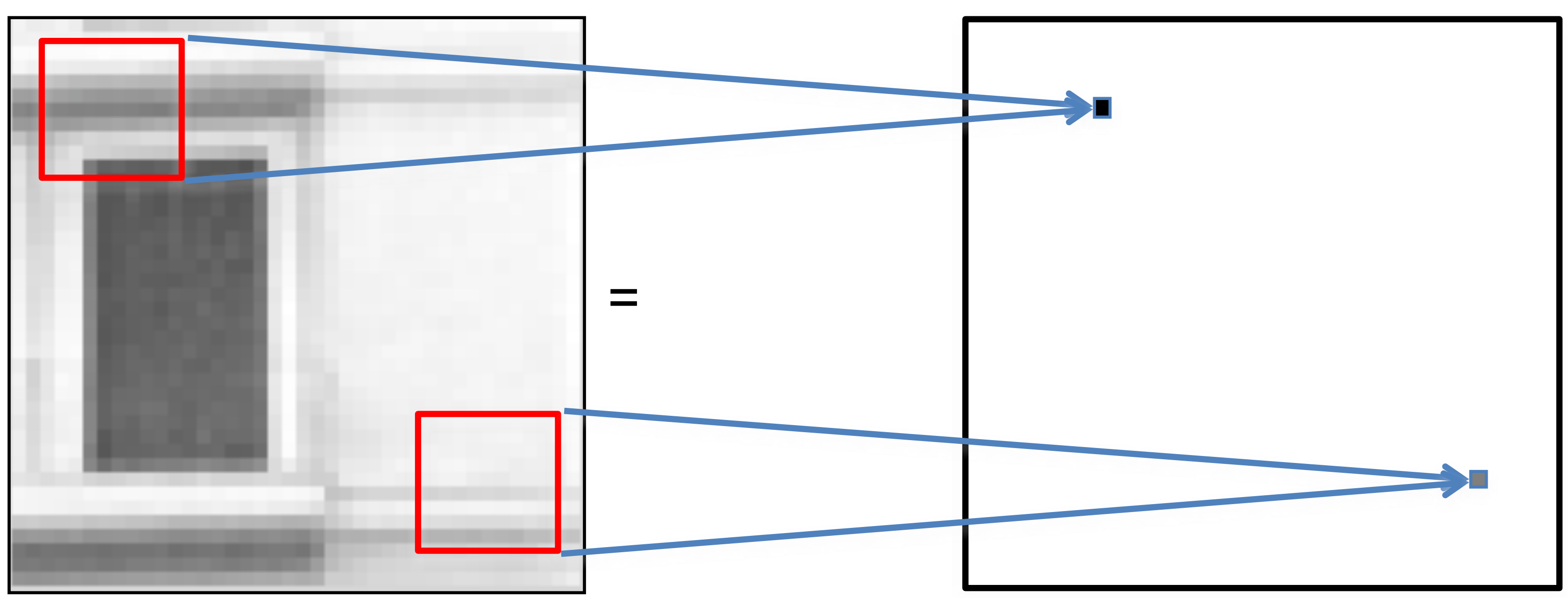
$$Y = HI$$

Linear filtering: Convolutions

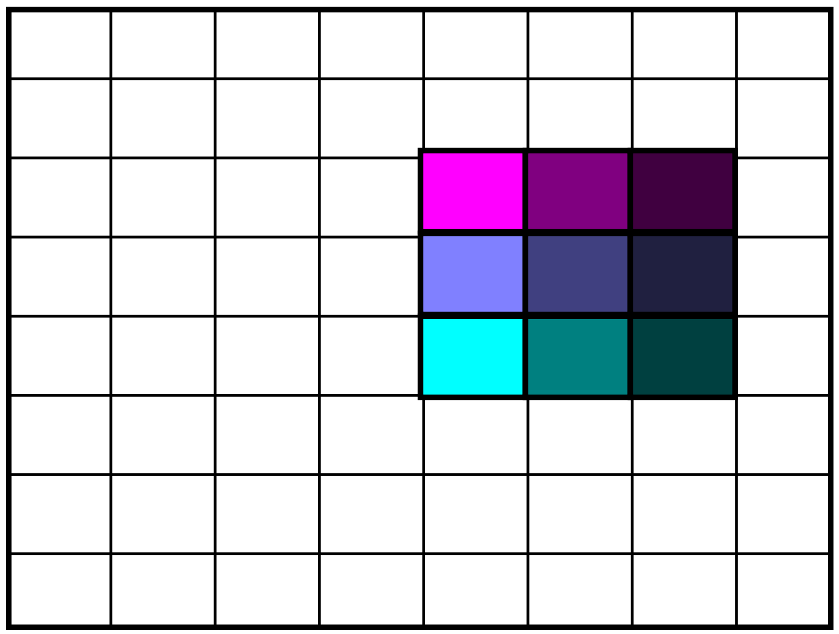
$$I[n_x, n_y] \longrightarrow \boxed{h[k, l]} \longrightarrow Y[n_x, n_y]$$

For a linear system, each output is a linear combination of all the input values:

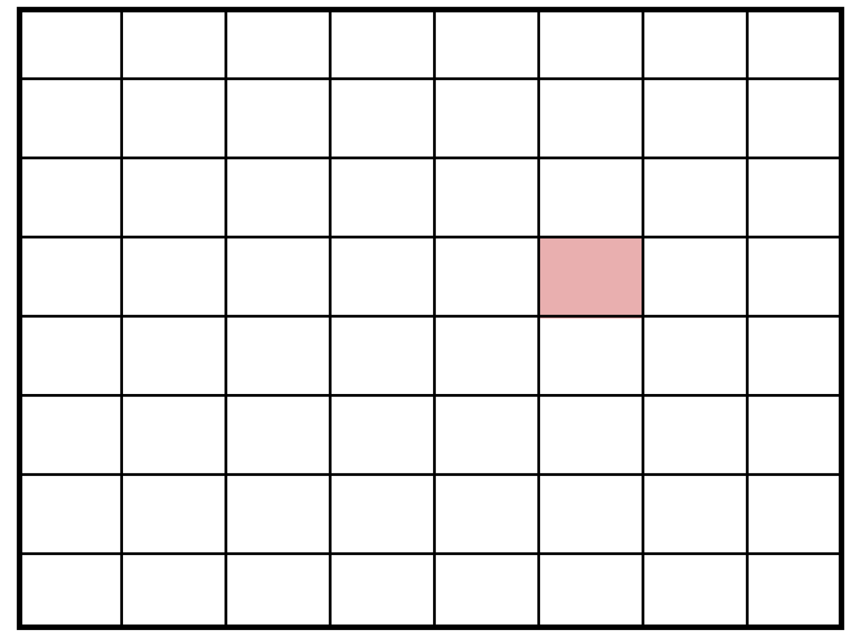
$$Y[m, n] = \sum_{k,l} h[k, l] I[m - k, n - l]$$



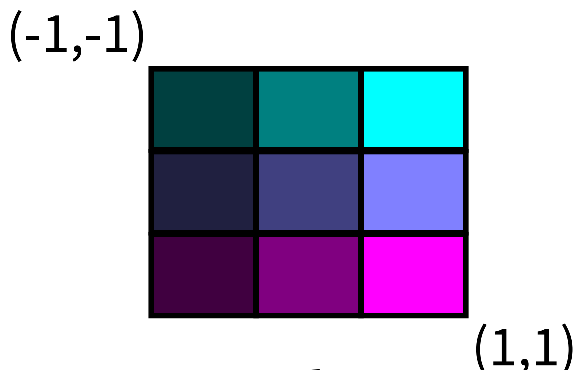
Convolutions



I



Y



h

$$Y[m,n] = \sum_{k,l} h[k,l] I[m-k, n-l]$$

Equivalent to cross-correlation with flipped filter (bottom-to-top & left-to-right)

Convolutions: Key Properties

- **Linearity:** $\text{filter}(f_1 + f_2) = \text{filter}(f_1) + \text{filter}(f_2)$
- **Shift invariance:** Same behavior irrespective of pixel location
  - Any shift-invariant operator can be represented by a convolution
- **Commutative:** $a * b = b * a$
- **Associative:** $a * (b * c) = (a * b) * c$
  - You an apply several filters one after the other (equivalent to one filter)
- **Scalars factor out:** $ka * b = a * kb = k(a * b)$

Convolutions: Impulse

Original

$$* \quad \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \quad =$$

Filtered
(no change)

Convolutions: Shifts

Original

$$* \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 0 & 1 \\ \hline 0 & 0 & 0 \\ \hline \end{array} =$$

Shifted right by 1 pixel

Convolutions: Blur (box filter)

Original

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

Shifted right by 1 pixel

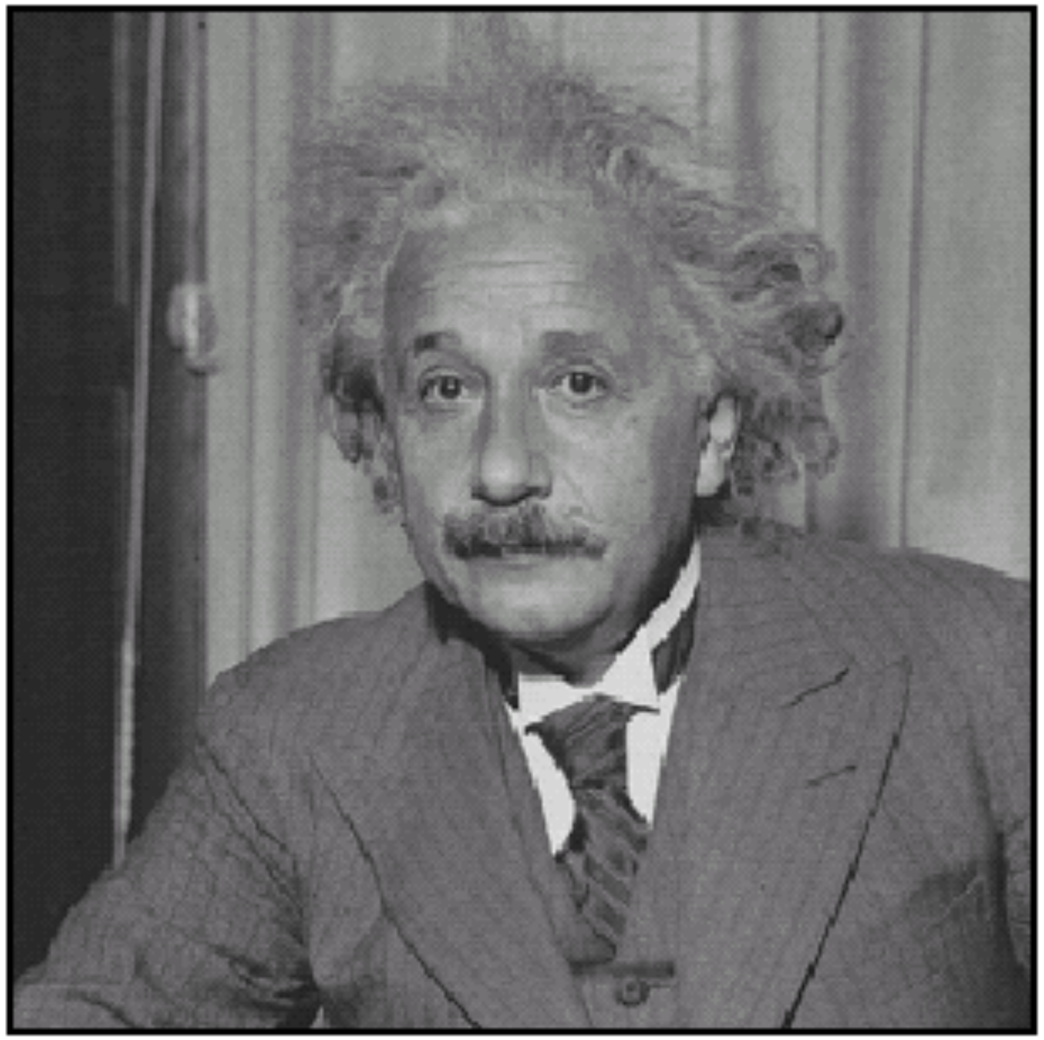
Convolutions: Sharpening

$$* \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right) =$$

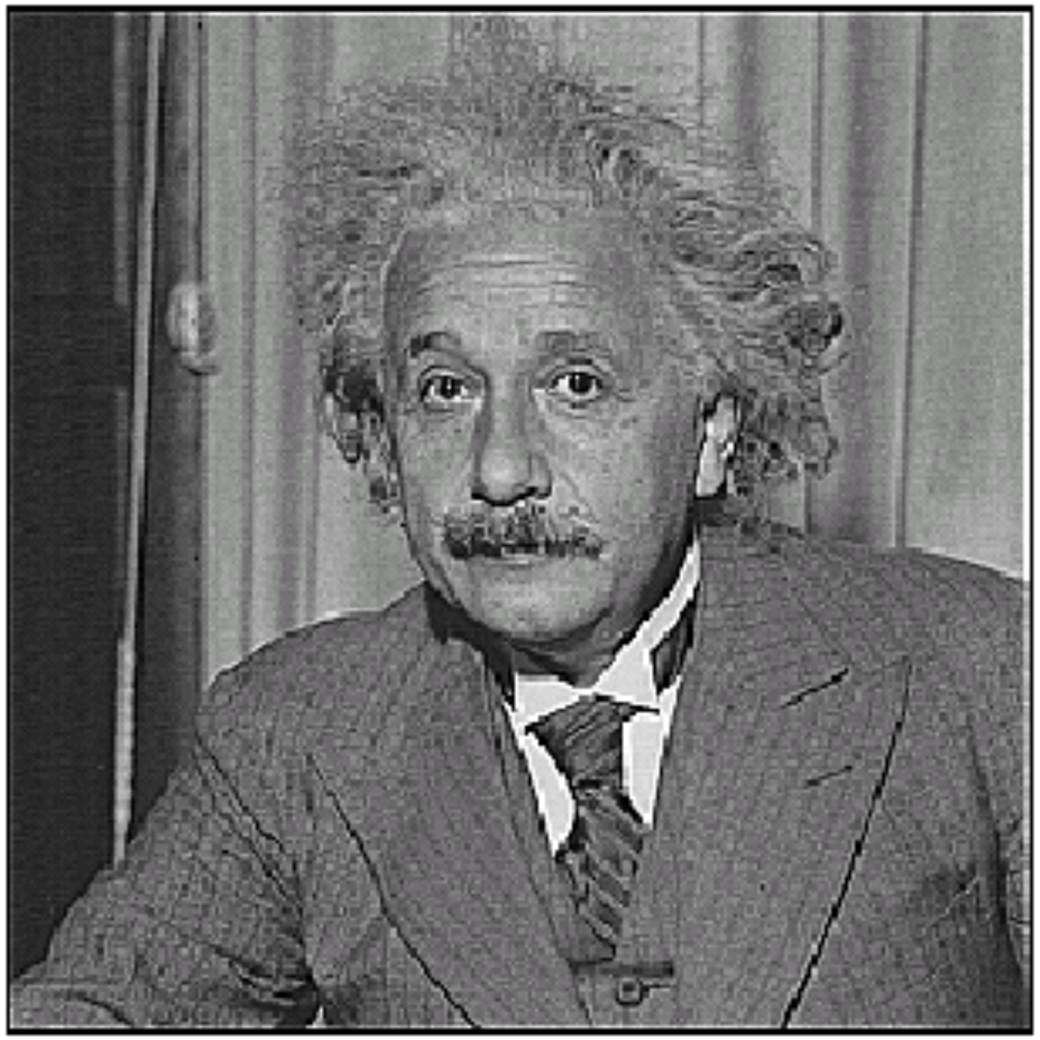
Original

Sharpening filter (accentuates differences with local averages)

Convolutions: Sharpening

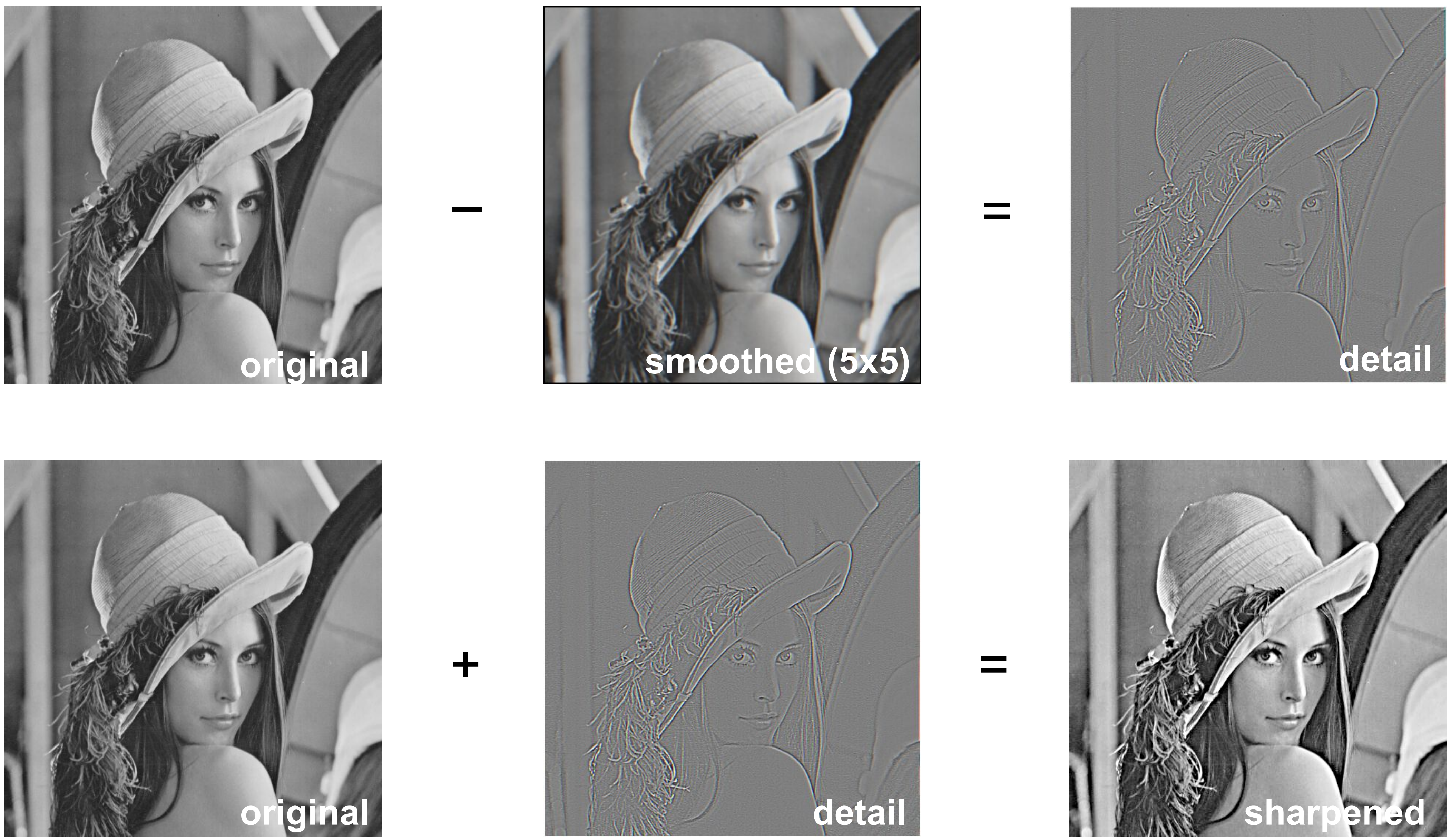


before



after

Convolutions: What does blurring remove?



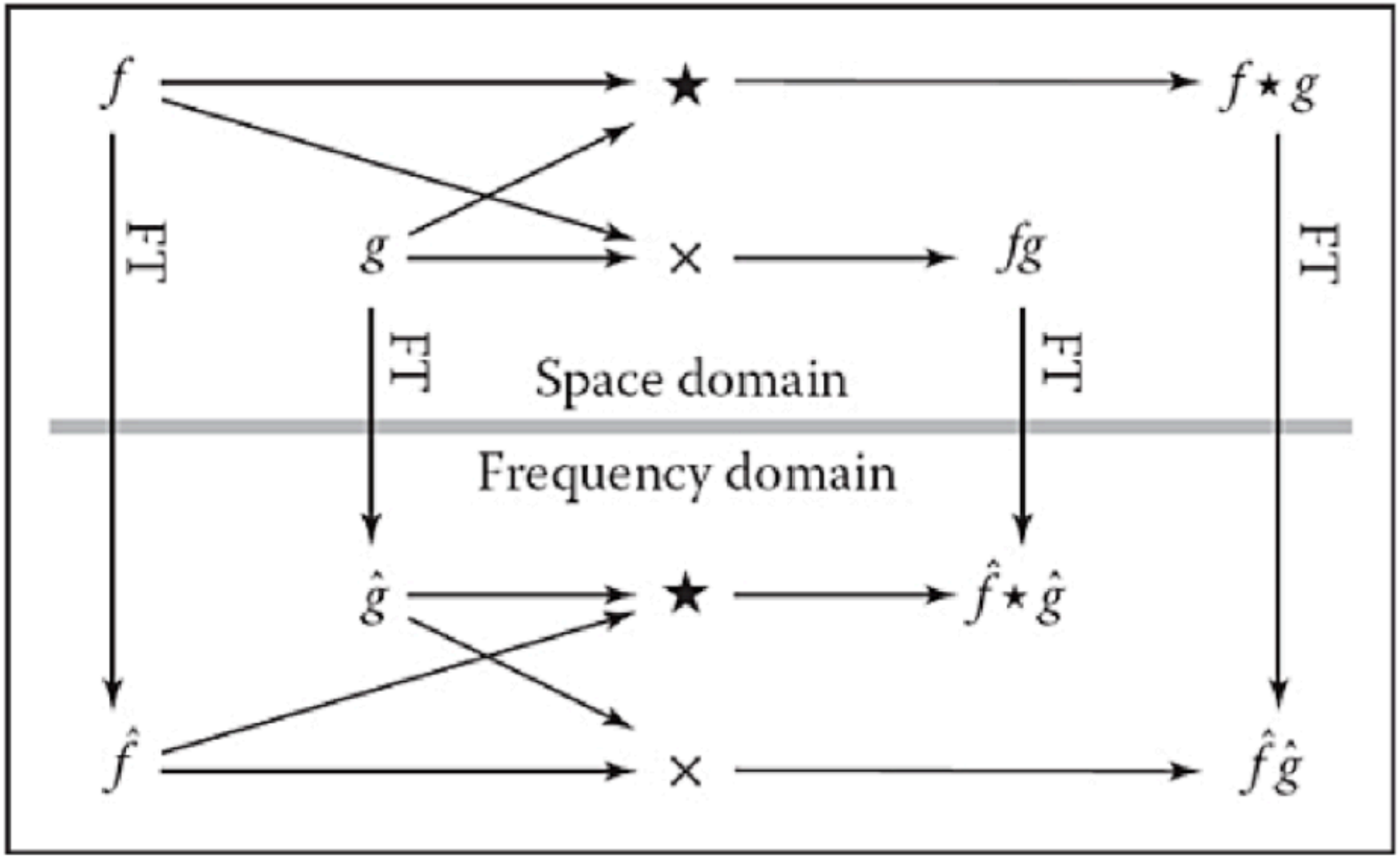
An example of unsharp masking

Fourier Transform

$$F(u) = \int_{-x}^{x} f(x) e^{-2\pi i u} dx$$

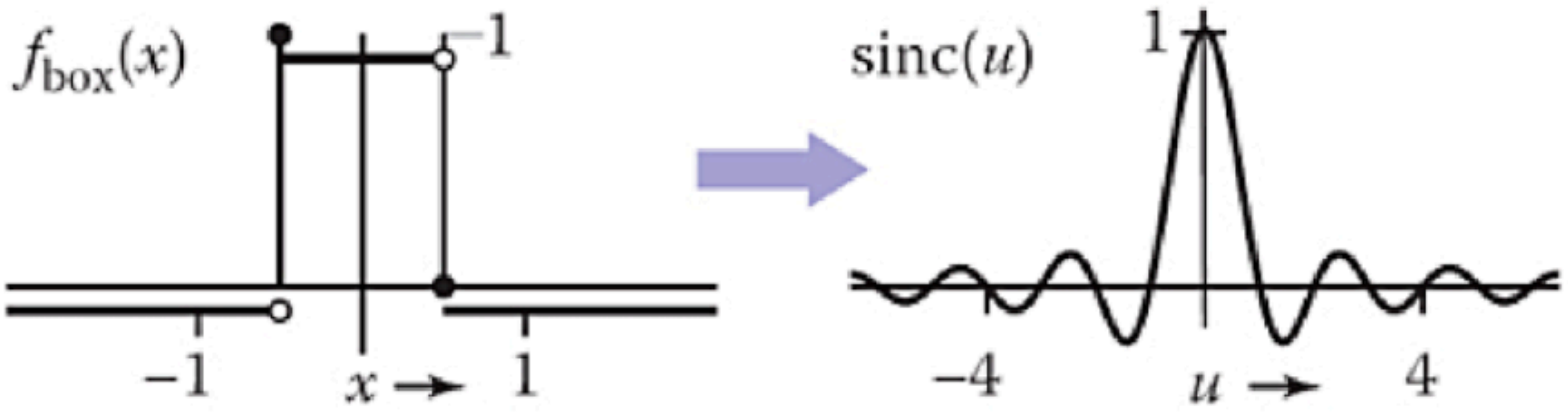
$$F(k) = \sum_{n=0}^{N-1} x_n e^{\frac{-2\pi k n}{N}}$$

Time and Frequency Domains



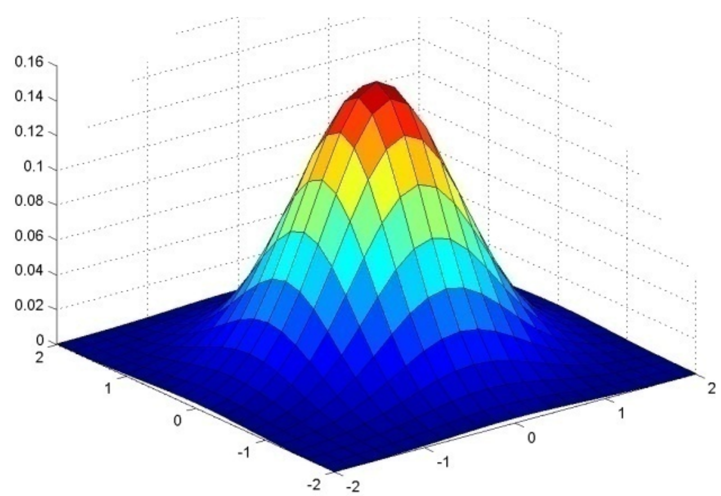
Taken from "Fundamentals of Computer Graphics", 4th ed.

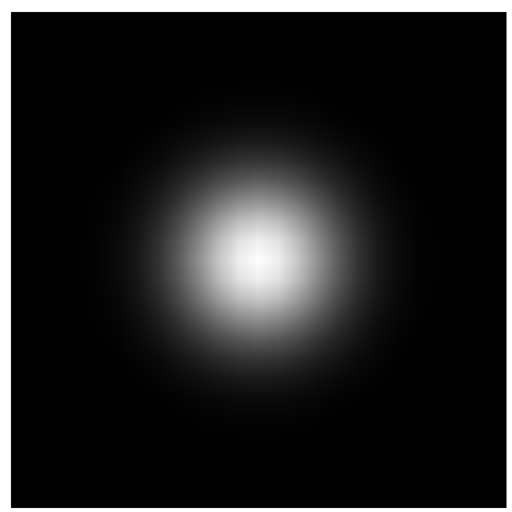
The Problem with the Box Filter



Gaussian kernels

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





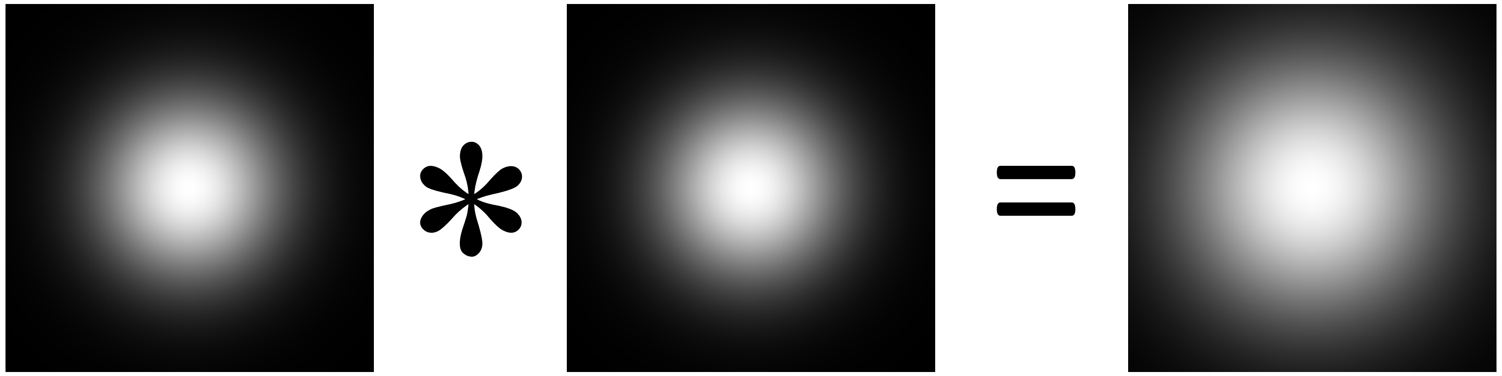
0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, σ = 1

A Gaussian has infinite support, but filters have finite support

Gaussian kernels: Properties

- Removes high-frequency components (low-pass filter)
- Convolution with another Gaussian is also Gaussian
    - Repeatedly smoothing with small std. dev. kernel is the same as convolving with a kernel with larger std. dev.
    - We can approximate heavy smoothing by repeatedly smoothing using a small kernel (e.g., smooth, subsample, smooth, subsample, etc.), which is more efficient



σ $\quad\quad$ σ $\quad\quad$ $\sqrt{2}\sigma$

Gaussian kernels: Properties

- Removes high-frequency components (low-pass filter)
- Convolution with another Gaussian is also Gaussian
  - Repeatedly smoothing with small std. dev. kernel is the same as convolving with a kernel with larger std. dev.
  - We can approximate heavy smoothing by repeatedly smoothing using a small kernel (e.g., smooth, subsample, smooth, subsample, etc.), which is more efficient
- Separability: 2D Gaussian factors into product of two 1D Gaussians —> separable kernel

$$
\begin{aligned}
G_\sigma(x, y) &= \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \\
&= \left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \right) \left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{2\sigma^2}} \right)
\end{aligned}
$$

Gaussian kernels: Properties

- Removes high-frequency components (low-pass filter)
- Convolution with another Gaussian is also Gaussian
    - Repeatedly smoothing with small std. dev. kernel is the same as convolving with a kernel with larger std. dev.
    - We can approximate heavy smoothing by repeatedly smoothing using a small kernel (e.g., smooth, subsample, smooth, subsample, etc.), which is more efficient
- Separability: 2D Gaussian factors into product of two 1D Gaussians —> separable kernel
    - We can perform 2D convolution by performing two 1D convolutions (one over rows and one over columns)

Separable filters

- The process of performing a convolution involves K^2 operations per pixel, where K is the width or height of the kernel
- Often, the process can be made more efficient by first performing a 1D horizontal convolution followed by a 1D vertical convolution, requiring $2K$ operations
- In this case, the kernel is said to be separable

Gaussian kernels

$\sigma = 1$

Gaussian Kernels

$$G_\sigma[n_x, n_y] \propto \exp\left(-\frac{n_x^2 + n_y^2}{2\sigma^2}\right)$$

$$\sum_{n_x, n_y} G_\sigma[n_x, n_y] = 1 \qquad n_x, n_y = [-S, -(S-1), \ldots, -1, 0, 1, \ldots, (S-1), S]$$

Gaussian kernels

$\sigma = 3$

Gaussian Kernels

$$G_\sigma[n_x, n_y] \propto \exp\left(-\frac{n_x^2 + n_y^2}{2\sigma^2}\right)$$

$$\sum_{n_x, n_y} G_\sigma[n_x, n_y] = 1 \qquad n_x, n_y = [-S, -(S-1), \ldots, -1, 0, 1, \ldots, (S-1), S]$$

Gaussian kernels

$\sigma = 4$

Gaussian Kernels

$$G_\sigma[n_x, n_y] \propto \exp\left(-\frac{n_x^2 + n_y^2}{2\sigma^2}\right)$$

$$\sum_{n_x, n_y} G_\sigma[n_x, n_y] = 1 \qquad n_x, n_y = [-S, -(S-1), \ldots, -1, 0, 1, \ldots, (S-1), S]$$