Spatial Filters

Digital Signal and Image Processing

Francesca Odone

Outline

Convolution and spatial filtering

Smoothing filters

Convolution

Convolution is defined as the the integral of the product of the two functions after one is reversed and shifted:

$$\begin{aligned}(f * h)(t) &= \int_{-\infty}^{+\infty} f(\tau)h(t-\tau)d\tau \\ &= \int_{-\infty}^{+\infty} f(t-\tau)h(\tau)d\tau \quad \text{(by commutativity)}\end{aligned}$$

https://phiresky.github.io/convolution-demo/

Discrete convolution

Let us consider two discrete 1D signals $f[\,]$ and $h[\,]$ defined on $\mathbb{Z}$

$$
\begin{aligned}
(f * h)[n] &= \sum_{m=-\infty}^{+\infty} f[m]h[n-m] \\
&= \sum_{m=-\infty}^{+\infty} f[n-m]h[m]
\end{aligned}
$$

Discrete (circular) convolution

- If the two discrete 1D signals $f[\,]$ and $h[\,]$ have a finite support in $\mathbb{Z}$, $[0, N-1]$, we can consider their periodic extension

$$
\begin{aligned}
(f * h)[n] &= \sum_{m=0}^{N-1} f[m]h[n-m] \\
&= \sum_{m=0}^{N-1} f[n-m]h[m]
\end{aligned}
$$

Università di Genova | MaLGa

Discrete circular convolution

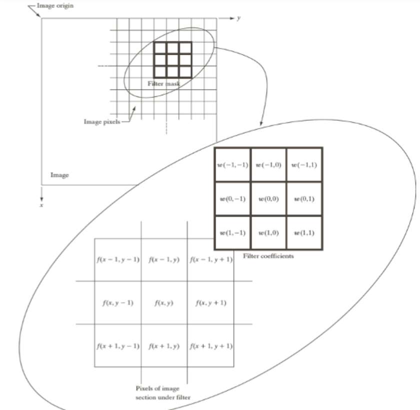
- For computational reasons we may consider them to have a different length: $f[n]$ of size N and $h[m]$ of size M
- In practice we often assume the filter h to be interpreted as a mask, smaller than the signal f, $M << N$, where each convolutional step acts on a signal neighbourhood
- The following notation is quite common (we assume the support of h to be in $\mathbb{Z}$ $[-M/2, M/2]$, so that the convolution acts on a central element

$$(f * h)[n] = \sum_{m=-M/2}^{+M/2} h[m]f[n-m]$$

2D Discrete convolution

We consider an image f and a 2D filter or kernel h of size $M \times L$. We obtain g, the filtered version of f by applying a 2D discrete convolution as follows:

$$g[x,y] = (f * h)[x,y] = \sum_{m=-M/2}^{M/2} \sum_{l=-L/2}^{L/2} h[m,l] f[x-m, y-l]$$



2D Discrete convolution

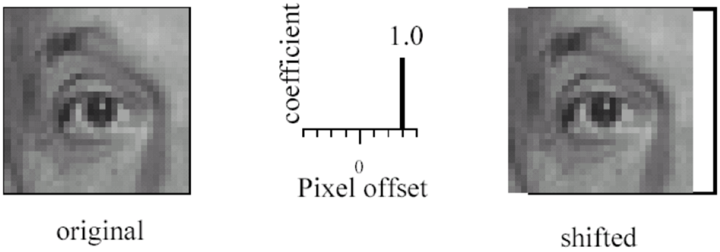
We consider an image f and a 2D filter or kernel h of size $M \times L$. We obtain g, the filtered version of f by applying a 2D discrete convolution as follows:

$$g[x, y] = (f * h)[x, y] = \sum_{m=-M/2}^{M/2} \sum_{l=-L/2}^{L/2} h[m, l] f[x - m, y - l]$$

$$\underbrace{\begin{array}{|c|c|c|c|c|c|c|c|}\hline 45 & 60 & 98 & 127 & 132 & 133 & 137 & 133 \\ \hline 46 & 65 & 98 & 123 & 126 & 128 & 131 & 133 \\ \hline 47 & 65 & 96 & 115 & 119 & 123 & 135 & 137 \\ \hline 47 & 63 & 91 & 107 & 113 & 122 & 138 & 134 \\ \hline 50 & 59 & 80 & 97 & 110 & 123 & 133 & 134 \\ \hline 49 & 53 & 68 & 83 & 97 & 113 & 128 & 133 \\ \hline 50 & 50 & 58 & 70 & 84 & 102 & 116 & 126 \\ \hline 50 & 50 & 52 & 58 & 69 & 86 & 101 & 120 \\ \hline \end{array}}_{f(x,y)} * \underbrace{\begin{array}{|c|c|c|}\hline 0.1 & 0.1 & 0.1 \\ \hline 0.1 & 0.2 & 0.1 \\ \hline 0.1 & 0.1 & 0.1 \\ \hline \end{array}}_{h(x,y)} = \underbrace{\begin{array}{|c|c|c|c|c|c|}\hline 69 & 95 & 116 & 125 & 129 & 132 \\ \hline 68 & 92 & 110 & 120 & 126 & 132 \\ \hline 66 & 86 & 104 & 114 & 124 & 132 \\ \hline 62 & 78 & 94 & 108 & 120 & 129 \\ \hline 57 & 69 & 83 & 98 & 112 & 124 \\ \hline 53 & 60 & 71 & 85 & 100 & 114 \\ \hline \end{array}}_{g(x,y)}$$

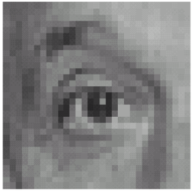
Convolution and Filtering: examples

Each convolutional step acts on an image neighbourhood

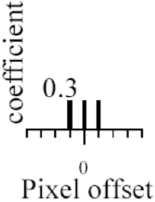


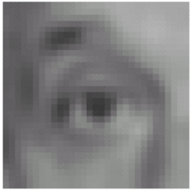
Convolution and Filtering: examples

Each convolutional step acts on an image neighbourhood



original





Blurred (filter applied in both dimensions).

Image filtering and convolution theorem

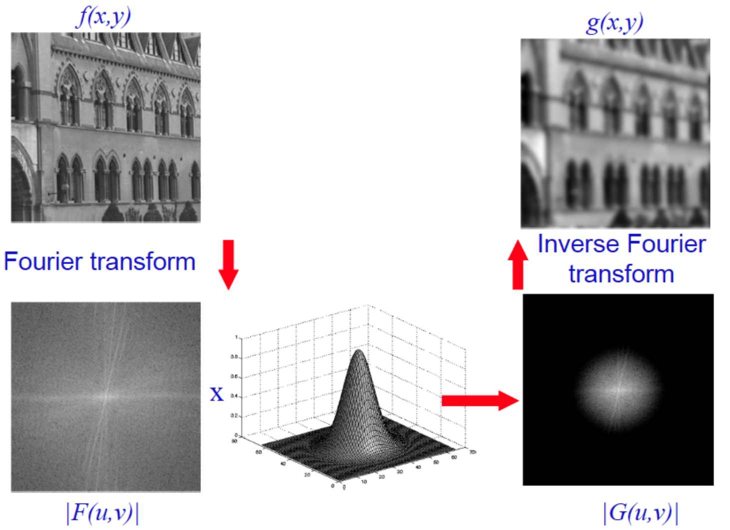
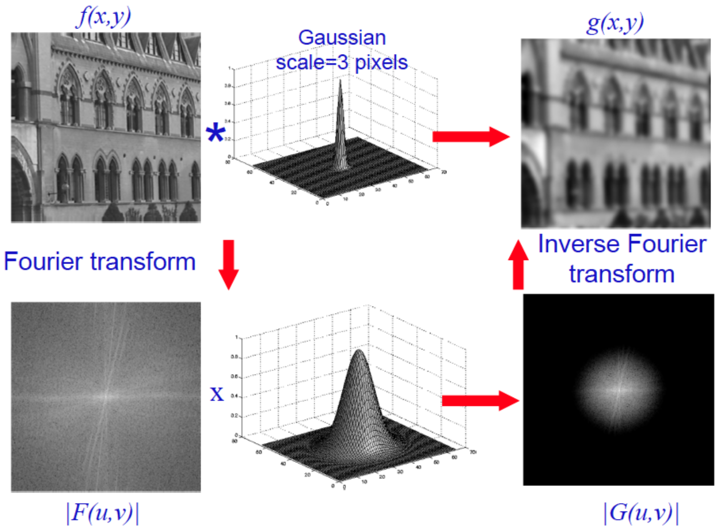


Image filtering and convolution theorem



Outline

Convolution and spatial filtering

Smoothing filters

Low Pass / smoothing filters

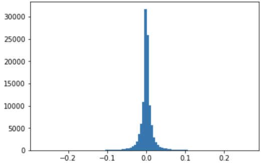
- Applying a smoothing filter in space corresponds to applying a low pass filter in Fourier (how are the two operations related? remember the Convolution theorem!)
- The main application of low pass filtering is in noise reduction, as noise is often hidden in the high frequencies

Noise

We only briefly mention the fact real images are affected by different sources of noise.
An empirical evidence of *acquisition noise*

I1

I2



Histogram of differences

Noise models

Different models of noise can be found in the literature
Pictorial images are often assumed to be affected by some amount of additive noise

$$f_r(x, y) = f_i(x, y) + \eta(x, y)$$

where

- f_i is the ideal (unknown) image
- f_r is the real (observed) image
- η is the noise term

A rather common and effective model for noise is the Gaussian distribution

Noise reduction: smoothing filters

- The simplest choice are the so-called *average filters* (that are in fact derived by a rectangle function ...)
- With an average filter we replace each pixel by the average of its neighbours and itself.
- It assumes that neighbouring pixels are similar, and the noise to be independent from pixel to pixel.
- Average can be represented by an appropriate kernel (consider also the weighted version)

$$\frac{1}{9} \times \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \qquad \frac{1}{16} \times \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

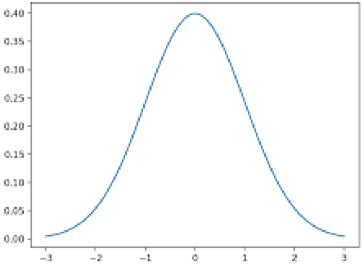
The larger the size of the kernel the more severe the smoothing

Noise reduction: Gaussian filter

We have seen the Gaussian filter in Fourier, its Fourier transform which is also a Gaussian can be used as a filter in space The Gaussian (zero-mean) distribution in 1D has the form

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

with σ the standard deviation

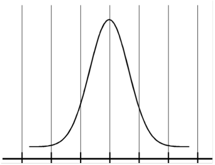


In 2D (isotropic case) $G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$

Noise reduction: Gaussian filter

A few details more on how to build a "small" discrete Gaussian kernel mask k_G in space. Let's see the 1D case

- Mask finite support: cut out the tails to keep "most" of the area
- Sample W odd points (including $x = 0$) and collect the corresponding values into the gaussian kernel $[G(-W/2), ..., G(0), ...G(W/2)]$
- Check the sum of the values (should be close to 1) and normalize the values of the kernel to 1



m=3
σ=0.6

m=5
σ=1

m=7
σ=1.4

m=9
σ=1.8

A parenthesis on efficiency: separable kernels

- A single 2D convolution costs $O(K^2)$ is $K \times K$ is the size of the kernel mask
- Two consecutive 1D filtering operations may be more efficient $O(2K)$
- Separable filters $k = vh^\top$ are more efficient
- If the filter is separable, instead than one 2D convolution we obtain the same effect with 2 consecutive 1D convolutions
  - $f_R = f * v$
  - $g = f_R * h$
- The Gaussian filter and the average filters are separable