CS-565 Computer Vision

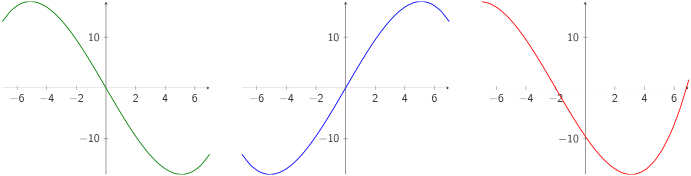
Nazar Khan

Department of Computer Science
University of the Punjab

3. Image Filtering

Convolution

- In 1D, a function $f(x)$ can be flipped as $f(-x)$.
- Similarly, $f(x+\tau)$ translates the function $f(x)$ by τ *to the left*.



$f(x)$ $f(-x)$ $f(x+2)$

- Can you plot $f(-x-2)$?

Convolution

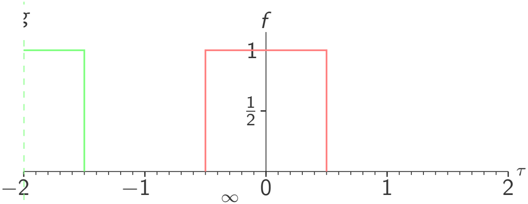
- For two functions $f(\tau)$ and $g(\tau)$ defined over $\mathbb{R}$, *continuous convolution* is defined as

$$(f * g)(t) := \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$$
$$= \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau$$

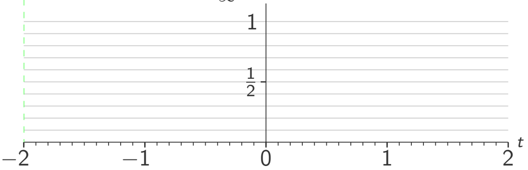
- For two functions $f[m]$ and $g[m]$ defined over $\mathbb{Z}$, *discrete convolution* is defined as

$$(f * g)[n] := \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$
$$= \sum_{m=-\infty}^{\infty} f[n-m]g[m]$$

1D Convolution
Example Continuous

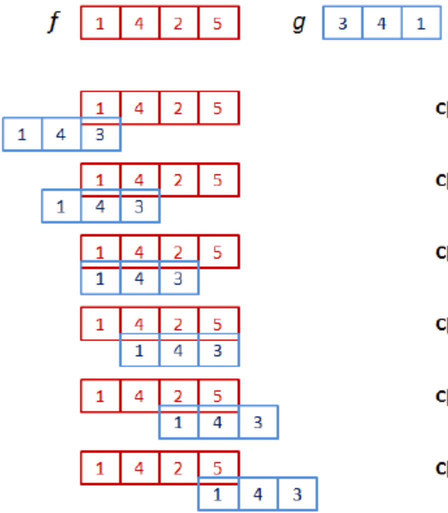


$$f * g = \int_{-\infty}^{\infty} f(\tau) g(t-\tau)\, d\tau$$



Source: `http://www.texample.net/tikz/examples/convolution-of-two-functions/`

1D Convolution
Example Discrete



$$c = f * g$$

C[0] = 1*3 = **3**

C[1] = 1*4 + 4*3 = **16**

C[2] = 1*1 + 4*4 + 2*3 = **23**

C[3] = 4*1 + 2*4 + 5*3 = **27**

C[4] = 2*1 + 5*4 = **22**

C[5] = 5*1 = **5**

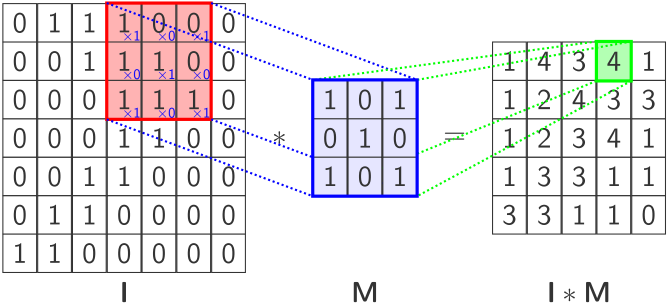
http://toto-share.com

2D Convolution

- Integral/sum of the product of two functions after one is reversed and shifted.
- Central role in image processing.
- For 2D images I and M, convolution is defined as

$$\begin{aligned}(I * M)[i,j] &:= \sum_{k\in\mathbb{Z}} \sum_{l\in\mathbb{Z}} I[k,l] M[i-k, j-l] \\ &= \sum_{k\in\mathbb{Z}} \sum_{l\in\mathbb{Z}} I[i-k, j-l] M[k,l]\end{aligned}$$

2D Convolution
Example



Modified from `https://github.com/PetarV-/TikZ/tree/master/2D%20Convolution`

M is usually called a *mask* or *kernel*.

2D Convolution

Applying a mask to an image

- In order to obtain $I_2 = I_1 * M$
  1. First flip the mask M in both dimensions.
  2. For each pixel p in I_1
     - 2.1 Place *mask origin* on top of the pixel.
     - 2.2 Multiply each mask weight with the pixel value under it.
     - 2.3 Sum the result and put in location of the pixel p in I_2.
- Any pixel can be defined as the origin of the mask. Usually it is the central pixel.

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & \mathbf{1} & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} \mathbf{1} & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & \mathbf{0} & -1 \end{bmatrix}$$

Dealing with boundaries

- What about edge and corner pixels where the mask goes outside the image boundaries?
  - Expand image I_1 with virtual pixels. Options are:
    1. Fill with a particular value, *e.g.* zeros.
    2. Replicating boundaries: fill with nearest pixel value.
    3. Reflecting boundaries: mirror the boundary
  - Fatalism: just ignore them. Not recommended since size of I_2 will shrink.

Dealing with boundaries
Expand by zeros

For a 5 $\times$ 5 image and 5 $\times$ 5 mask

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & a & b & c & d & e & 0 & 0 \\ 0 & 0 & f & g & h & i & j & 0 & 0 \\ 0 & 0 & k & l & m & n & o & 0 & 0 \\ 0 & 0 & p & q & r & s & t & 0 & 0 \\ 0 & 0 & u & v & w & x & y & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Dealing with boundaries
Replicating boundaries

For a 5×5 image and 5×5 mask

$$\begin{bmatrix} a & a & a & b & c & d & e & e & e \\ a & a & a & b & c & d & e & e & e \\ a & a & a & b & c & d & e & e & e \\ f & f & f & g & h & i & j & j & j \\ k & k & k & l & m & n & o & o & o \\ p & p & p & q & r & s & t & t & t \\ u & u & u & v & w & x & y & y & y \\ u & u & u & v & w & x & y & y & y \\ u & u & u & v & w & x & y & y & y \end{bmatrix}$$

Dealing with boundaries
Reflecting boundaries

For a 5 × 5 image and 5 × 5 mask

$$\begin{bmatrix} m & l & k & l & m & n & o & n & m \\ h & g & f & g & h & i & j & i & h \\ c & b & a & b & c & d & e & d & c \\ h & g & f & g & h & i & j & i & h \\ m & l & k & l & m & n & o & n & m \\ r & q & p & q & r & s & t & s & r \\ w & v & u & v & w & x & y & x & w \\ r & q & p & q & r & s & t & s & r \\ m & l & k & l & m & n & o & n & m \end{bmatrix}$$

Convolution Masks

- Different masks lead to different effects.

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} \quad \frac{1}{2}\begin{bmatrix}1 & 1 & 1\\ 0 & 0 & 0\\ -1 & -1 & -1\end{bmatrix} \quad \frac{1}{18}\begin{bmatrix}2 & 3 & 2\\ 0 & 0 & 0\\ -2 & -3 & -2\end{bmatrix}$$

Averaging x-Derivative x-Derivative & averaging

- Interactive demo at `https://phiresky.github.io/convolution-demo/`
- Cost of convolving an $m \times n$ image with a $k \times k$ kernel is k^2 multiplications plus $k^2 - 1$ additions repeated for $m \times n$ locations. That is $mn(2k^2 - 1)$ operations.

Properties of Convolution

- Commutativity: $I * M = M * I$
  - Signal and kernel play an equal role.
- Associativity: $(I * M_1) * M_2 = I * (M_1 * M_2)$
  - Successive convolutions with kernels M_1 and M_2 is equivalent to a single convolution with kernel $M_1 * M_2$ *which is computationally much cheaper* since kernels are usually smaller than images.
- Shift Invariance: $\text{Translation}(I * M) = \text{Translation}(I) * M$
  - Translation of convolved signal is equivalent to convolution of translated signal.
- Linearity: $(aI + bJ) * M = a(I * M) + b(J * M)\ \forall\ a, b \in \mathbb{R}$
  - Single convolution of a linear combination of signals is equivalent to a linear combination of multiple convolutions.
  - Obviously, the single convolution is *computationally much cheaper*.

Because of the last two properties, convolution is called *linear shift invariant (LSI)* filtering.

Gaussian Kernel

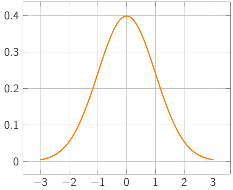
- A widely used mask for smoothing is the *Gaussian* kernel.

$$1D: \; g(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

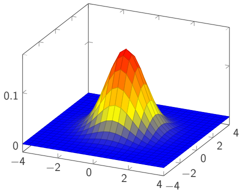
$$2D: \; G(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(x-\mu_1)^2 + (y-\mu_2)^2}{2\sigma^2}\right)$$

where μ is the 1D mean, (μ_1, μ_2) is the 2D mean and σ^2 is the variance.

- Most commonly $\mu = 0$ and $\sigma = 1$.

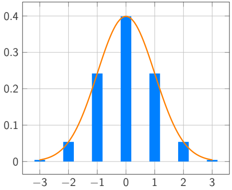


$\mu = 0, \sigma = 1$



$(\mu_1, \mu_2) = (0, 0), \sigma = 1$

Gaussian Kernel
1D Discrete approximation



$$\begin{bmatrix} 0.0044 & 0.054 & 0.242 & 0.399 & 0.242 & 0.054 & 0.0044 \end{bmatrix}$$

Gaussian Kernel
2D Discrete Approximation

$$\begin{bmatrix} 0.0000 & 0.0002 & 0.0011 & 0.0018 & 0.0011 & 0.0002 & 0.0000 \\ 0.0002 & 0.0029 & 0.0131 & 0.0215 & 0.0131 & 0.0029 & 0.0002 \\ 0.0011 & 0.0131 & 0.0585 & 0.0965 & 0.0585 & 0.0131 & 0.0011 \\ 0.0018 & 0.0215 & 0.0965 & 0.1592 & 0.0965 & 0.0215 & 0.0018 \\ 0.0011 & 0.0131 & 0.0585 & 0.0965 & 0.0585 & 0.0131 & 0.0011 \\ 0.0002 & 0.0029 & 0.0131 & 0.0215 & 0.0131 & 0.0029 & 0.0002 \\ 0.0000 & 0.0002 & 0.0011 & 0.0018 & 0.0011 & 0.0002 & 0.0000 \end{bmatrix}$$

Gaussian Kernel

2D Discrete approximation

$$\begin{bmatrix} 0.0044 & 0.054 & 0.242 & 0.399 & 0.242 & 0.054 & 0.0044 \end{bmatrix} * \begin{bmatrix} 0.0044 \\ 0.054 \\ 0.242 \\ 0.399 \\ 0.242 \\ 0.054 \\ 0.0044 \end{bmatrix}$$

Separability of Gaussian Kernels: Convolution with 2D Gaussian can be performed via two successive convolutions with 1D Gaussians which are computationally much cheaper.

Gaussian Kernel

Discrete integer approximation

$$1D: \frac{1}{9997}\begin{bmatrix}44 & 540 & 2420 & 3989 & 2420 & 540 & 44\end{bmatrix}$$

$$2D: \frac{1}{9997^2}\begin{bmatrix}44 & 540 & 2420 & 3989 & 2420 & 540 & 44\end{bmatrix} * \begin{bmatrix}44\\540\\2420\\3989\\2420\\540\\44\end{bmatrix}$$

For images stored in unsigned 8-bit integer format (uint8), integer operations are faster than floating point operations.

Figure: Effect of convolving with averaging masks of increasing size. Author: N. Khan (2018)

Figure: Effect of convolving with Gaussian masks of increasing size. Author: N. Khan (2018)

Non-linear Filtering

- Any filtering performed via convolution is *linear filtering*.
- *Non-linear filtering* yields additional benefits.
  - Median filtering
  - Bilateral filtering
  - Non-local means

Figure: Effect of median filtering with masks of increasing size. Author: N. Khan (2018)

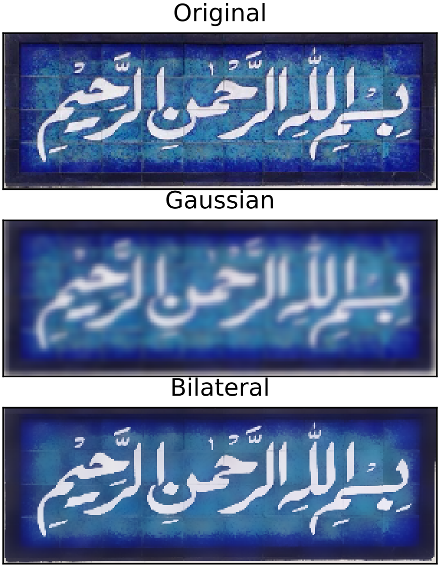


Figure: Comparison of 35×35 Gaussian smoothing with bilateral filter of diameter 35. Notice how bilateral filtering preserves edges. Author: N. Khan (2018)