# What you should know about convolutions

The convolution of the two discrete functions $f(x,y), g(x,y)$ is the discrete function $h(u,v)$ given by:

$$h(u,v) = f(x,y) * g(x,y) = \sum_{x,y} f(u-x, v-y) \cdot g(x,y)$$

The cross correlation of these functions is a discrete function $h(u,v)$ given by:

$$h(u,v) = f(x,y) \otimes g(x,y) = \sum_{x,y} f(x-u, y-v) \cdot g(x,y)$$

The functions are defined for all (positive and negative) integers $x, y, u, v$. We use the convention that unspecified values are zero.

## Linearity

Both convolution and cross correlation are linear operations:

$$f * (g_1 + g_2) = f * g_1 + f * g_2$$
$$f * (a \cdot g) = (a \cdot f) * g$$

$$f \otimes (g_1 + g_2) = f \otimes g_1 + f \otimes g_2$$
$$f \otimes (a \cdot g) = (a \cdot f) \otimes g$$

## Important properties of convolution

The following properties hold for convolution but not necessarily for cross-correlation:

$$f * g = g * f$$
$$f * (g_1 * g_2) = (f * g_1) * g_2$$

## Computing convolutions and cross-correlations

In may help to think of $f$ as being a mask, or template, and of $g$ as the picture. The cross-correlation value at the point $u, v$ can be computed by shifting the function $f$ and positioning it such that its $0, 0$ is "above" the coordinate $u, v$ of the function $g$. The value $h(u,v)$ is the sum of the products of the values of $g$ by the corresponding values of $f$.

Two dimensional convolutions can be computed by first reversing the order of the elements in each row of $f$, then reversing the order of the rows in $f$, and then computing cross correlation between the twice reversed $f$ and $g$.

**Example:**

$$g = \begin{array}{|ccccc|}\hline a_1 & a_2 & a_3 & a_4 & a_5 \\ b_1 & b_2 & b_3 & b_4 & b_5 \\ c_1 & c_2 & c_3 & c_4 & c_5 \\ \hline \end{array}$$

$$f = \begin{array}{|cc|}\hline 1 & 0 \\ 0 & -2 \\ \hline \end{array}$$

$$f \otimes g = \begin{array}{|cccccc|}\hline -2a_1 & -2a_2 & -2a_3 & -2a_4 & -2a_5 & 0 \\ -2b_1 & a_1 - 2b_2 & a_2 - 2b_3 & a_3 - 2b_4 & a_4 - 2b_5 & a_5 \\ -2c_1 & b_1 - 2c_2 & b_2 - 2c_3 & b_3 - 2c_4 & b_4 - 2c_5 & b_5 \\ 0 & c_1 & c_2 & c_3 & c_4 & c_5 \\ \hline \end{array}$$

The convolution of $f$ and $g$ is the same as the cross correlation of $f'$ and $g$, where:

$$f' = \begin{array}{|cc|}\hline -2 & 0 \\ 0 & 1 \\ \hline \end{array}$$

$$f * g = \begin{array}{|cccccc|}\hline a_1 & a_2 & a_3 & a_4 & a_5 & 0 \\ b_1 & -2a_1 + b_2 & -2a_2 + b_3 & -2a_3 + b_4 & -2a_4 + b_5 & -2a_5 \\ c_1 & -2b_1 + c_2 & -2b_2 + c_3 & -2b_3 + c_4 & -2b_4 + c_5 & -2b_5 \\ 0 & -2c_1 & -2c_2 & -2c_3 & -2c_4 & -2c_5 \\ \hline \end{array}$$

## Smoothing

We discussed two methods for cleaning "salt and pepper" noise. The first method was averaging by convolutions with masks such as:

$$\frac{1}{4}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \quad \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

The second method was Median Filtering, where each pixel is replaced with the median of its neighbors. The median is computed by first sorting the neighbors and then taking the middle value in the sorted list. Usually a neighborhood includes the pixel and its 4 closest neighbors or its 8 closest neighbors.

For example, if we consider as a neighborhood of a pixel its 8 neighbors, then a median filter would not change the value of the central pixel in $\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix}$ or in $\begin{matrix} 4 & 4 & 4 \\ 5 & 5 & 6 \\ 9 & 9 & 9 \end{matrix}$. Median filter changes the value of the central pixel from 5 to 7 in the following case: $\begin{matrix} 7 & 4 & 4 \\ 5 & 5 & 8 \\ 9 & 9 & 9 \end{matrix}$.

By considering the behavior of averaging and median filtering for flat, step, and slope surfaces we concluded that the median filter appears to perform better than averaging because it does not blur edges.

## Edge detection

The derivative of a picture in the horizontal direction can be obtained by a cross correlation with the mask $\Delta_1 = \begin{bmatrix} -1 & 1 \end{bmatrix}$. The derivative in the vertical direction can be obtained by a cross correlation with the mask $\Delta_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$. The norm of the Gradient can be used to detect edges. It can be computed in several ways, e.g., $\sqrt{\Delta_1^2 + \Delta_2^2}$ , $|\Delta_1| + |\Delta_2|$ , $\max(|\Delta_1|, |\Delta_2|)$.

Experience shows that better edges are obtained by other choices of $\Delta_1$ and $\Delta_2$. The Roberts operator requires that we choose:

$$\Delta_1 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \quad \Delta_2 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

Here we must choose the same center in both $\Delta_1$ and $\Delta_2$.
The Sobel operator requires that we choose:

$$\Delta_1 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \Delta_2 = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

The Laplacian of a picture corresponds intuitively to its second derivative. It can be computed

by a cross correlation of the picture with the mask:

$$\begin{array}{|ccc|}\hline 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \\ \hline \end{array}$$

## Template matching

In order to detect a template $f$ in a picture $g$ we can compute the cross correlation of $f$ and $g$. The values of the result can be taken as a first approximation to the match, where large values correspond to a probable match, and small values to a probable mismatch.

A better match measure is obtained when each value is normalized. The normalization can be computed by the following three steps: 1- square each pixel in the picture $g$. 2- compute the cross correlation of the result with a mask of the same shape as $f$ but where all mask values are 1. 3- compute the square root of each element in the result to obtain the normalization factor. The normalized match measure is the ratio between the cross correlation of $f$ with $g$ and the normalization factor.