Chapter 3

Fundamental of Image Processing

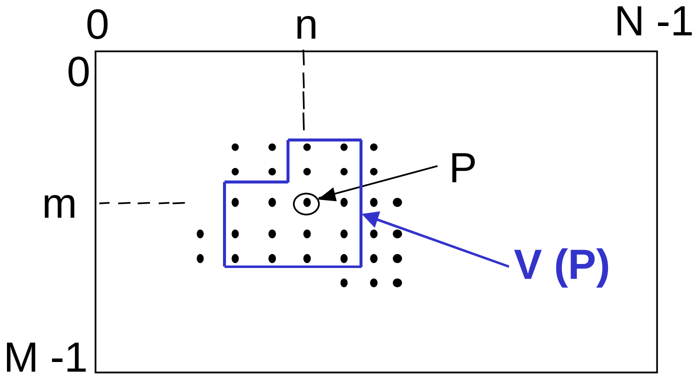
Linear Filtering

The Concept of Neighborhoods

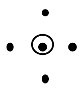
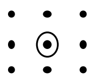
The Concept of Neighborhoods

- Point P: affix p = (m, n)

- Neighborhood: V(P) = {P’ connected to P}



Discrete image domain

Examples: 4-N  8-N 

4-connected neighborhood 8- connected neighborhood

Image processing is fundamentally based on techniques using neighborhoods. An image processing which is performed at the affix p of the pixel P depends not only on this pixel P but also on pixels in its neighboring area. Let us consider a pixel P whose location in the image is defined by the coordinates (m, n). Its affix is thus p = (m, n). A neighborhood V(P) of the pixel P can be defined by a set of pixels P' that are connected to P.

The pixel P (circled in the figure) belongs to its own neighborhood V(P).

We must here define the concept of connectivity: the criteria that describe how pixels within a discrete image form a connected group. Rather than developing this concept, we show here in the top figure the two most ”y qLRWycqLvmmon examples:

- a “4-connected” neighborhood: the surrounded pixel has only four neighboring pixels. The distance between the surrounded pixel and any pixel of its neighborhood is d

Example – Convolution – Correlation

Example : filter support of size (3×5) (3 in vertical and 5 in horizontal)

Convolution kernel : $$h = \begin{bmatrix} h_{-1,-2} & h_{-1,-1} & h_{-1,0} & h_{-1,1} & h_{-1,2} \\ h_{0,-2} & h_{0,-1} & h_{0,0} & h_{0,1} & h_{0,2} \\ h_{1,-2} & h_{1,-1} & h_{1,0} & h_{1,1} & h_{1,2} \end{bmatrix}$$

Convolution

$$I_s(m,n) = \sum_{j=-2}^{+2} \sum_{i=-1}^{+1} h(i,j).I_e(m-i, n-j)$$

Correlation Let $h^*(i,j) = h(-i,-j)$

(=> symmetric of h with respect to (0,0))

$$I_s(m,n) = \sum_{j=-2}^{+2} \sum_{i=-1}^{+1} h^*(i,j).I_e(m+i, n+j)$$

$$h^* = \begin{bmatrix} h_{1,2} & h_{1,1} & h_{1,0} & h_{1,-1} & h_{1,-2} \\ h_{0,2} & h_{0,1} & h_{0,0} & h_{0,-1} & h_{0,-2} \\ h_{-1,2} & h_{-1,1} & h_{-1,0} & h_{-1,-1} & h_{-1,-2} \end{bmatrix}$$

In the case of linear filtering using convolution, the filter is completely characterized by the coefficients { h(i,j) } (which can also be written { $h_{i,j}$ } to remind us that "i" and "j" are discrete variables). These coefficients define the convolution "kernel" of the filter.

This kernel defines:

- the neighborhood $V(P_e)$ to use (in the example it is a (3×5) neighborhood where the position (0,0) must be centered on P_e);
- the respective weightings h(i, j) of each neighboring pixel needed to calculate the new value P_S.

When the size of the rectangular neighborhood support (I, J) and the weightings are known, we can calculate all the pixels of the image Is:

$$I_S(m,n) = \sum_{i\in I} \sum_{j\in J} h(i,j).I_e(m-i, n-j)$$

Note: When computing an output pixel at the boundary of an image, a portion of the convolution kernel is usually off the edge of the image I_e. It is thus necessary to specify "boundary conditions" to process border distortions (simple solution: to consider only the pixels in the image). Some ways of dealing with the boundaries are described in the exercise: "Linear filtering" in this chapter.

For example, let us consider a linear filter « h ». Its convolution kernel is:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

In fact this filter is the sum of a "Laplacian" filter (contour detection) and an Identity filter. Together they form a new filter "h", an "enhancement" filter.
Let the 3×3 image I_e be defined by:

$$\begin{bmatrix} 4 & 125 & 255 \\ 7 & 0 & 45 \\ 9 & 56 & 13 \end{bmatrix}$$

This image I_e is filtered by h.
Here are all the steps to compute the output value of the central pixel $I_S(1, 1)$:

$$I_S(1,1) = \sum_{i=-1}^{+1} \sum_{j=-1}^{+1} h(i,j) \,.\, I_e(1-i,1-j)$$

$$= \sum_{i=-1}^{+1} h(i,-1) \,.\, I_e(1-i,1-(-1)) + h(i,0) \,.\, I_e(1-i,1-0) + h(i,1) \,.\, I_e(1-i,1-1)$$

$$= \Big[h(-1,-1).I_e(1+1,2) + h(-1,0).I_e(1+1,1) + h(-1,1).I_e(1+1,0)$$

$$+ h(0,-1).I_e(1,2) + h(0,0).I_e(1,1) + h(0,1).I_e(1,0)$$

$$+ h(1,-1).I_e(1-1,2) + h(1,0).I_e(1-1,1) + h(1,1).I_e(1-1,0) \Big]$$

$$= (0\times13) + (-1\times56) + (0\times9) + (-1\times45) + (5\times0) + (-1\times7) + (0\times255) + (-1\times125) + (0\times4)$$

$$I_S(1,1) = -233$$

Warning: row and column indices of the input image are not the same as the indices of the convolution kernel. In the input image, indices run from 0 to M-1 (rows) and from 0 to N-1 (columns). In convolution kernel, the element $h_{0,0}$ is centered and the indices run from –I to +I (rows) and from –J to J (columns).

Note that the bi-dimensional function « h^* », which is the symmetric of h with respect to (0, 0), can be used to compute the output values of the pixel $I_S(m, n)$. The output values are thus defined as the correlation between I_e and h^* :

$$I_S(m,n) = \sum_{i\in I} \sum_{j\in J} h^*(i,j).I_e(m+i,n+j)$$

Typical examples of convolution masks

$$\begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} \qquad \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix} \qquad \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix}$$

3×3 Mask 3×5 Mask 5×5 Mask

Convolution masks:

(~ circular: 45 pixels ; rectangular: 11×11)

- DC filter Gain: $\sum_i \sum_j h(i, j) = H(\nu_X = 0, \nu_Y = 0)$
- Symmetric filter : $h(-i,-j) = h(i,j)$ then: convolution ≡ correlation

=> $H(\nu_X, \nu_Y)$ is a real transfer function (no phase term)

The shape of a convolution mask can be one of many kinds. The size of the convolution mask also defines the size of the neighborhood. Some filter supports are presented here: 3×3, 3×5, and 5×5.
As in case of 1D signals, we define the frequency response $H(\nu_X, \nu_Y)$ of the filter. It depends on the horizontal (ν_X) and vertical (ν_Y) spatial frequencies. $H(\nu_X, \nu_Y)$ is the 2D Fourier transform of the impulse response:

$$\mathbf{H(\nu_X, \nu_Y) = \sum_n \sum_m h(m, n) \exp\left[-2j\pi (n\nu_X + m\nu_Y)\right]}$$

The DC component $H(\nu_X = 0, \nu_Y = 0)$ can be computed as the sum of all the convolution kernel coefficients $h(i, j)$: Gain_DC=$\sum_i \sum_j h(i,j)$ (DC stands for direct current, an electrical engineering term).

The exercise "FFT" details the methods to compute and display the spectrum image and the frequency response of a filter. The notion of spatial frequency is developed there. In the special case, when the filter is symmetrical, $H(\nu_X, \nu_Y)$ is a real frequency response.

Note: in the following chapters we will use the term "transfer function" instead of "frequency response" (it is a misuse of language).