EXPERIMENT NO.4

To perform image filtering in spatial domain.

Date of Performance	
Date of Checking	

Title: To perform image filtering in spatial domain.

Aim:

1. To study spatial domain filtering.
2. To filter given image using different masks.

Software: MATLAB/SciLab

Theory:-

In this experiment, the convolution operator is presented. This operator is used in the linear image filtering process applied in the spatial domain (in the image plane by directly manipulating the pixels) or in the frequency domain (applying a Fourier transform, filtering and then applying the inverse Fourier transform. Examples of such filters are: low pass filters (for smoothing) and high pass filters (for edge enhancement).

Convolution process in the spatial domain :

The convolution process implies the usage of a convolution mask/kernel H (usually with symmetric shape and size w*w, with w=2k+1) which is applied on the source image.

$$I_D = H * I_S$$

$$I_D(x,y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} H(i,j).I_s\,(x+i, y+j)$$

$$for\ x = 0\ to\ M-1\ and\ y = 0\ to\ N-1$$

This implies the scanning of the source image I_s pixel by pixel, ignoring the first and last k rows and columns(Fig.1) and the computation of the intensity value in the current position (x, y) of the destination image I_D. The convolution mask is positioned spatially with its central element over the current position(x,y).

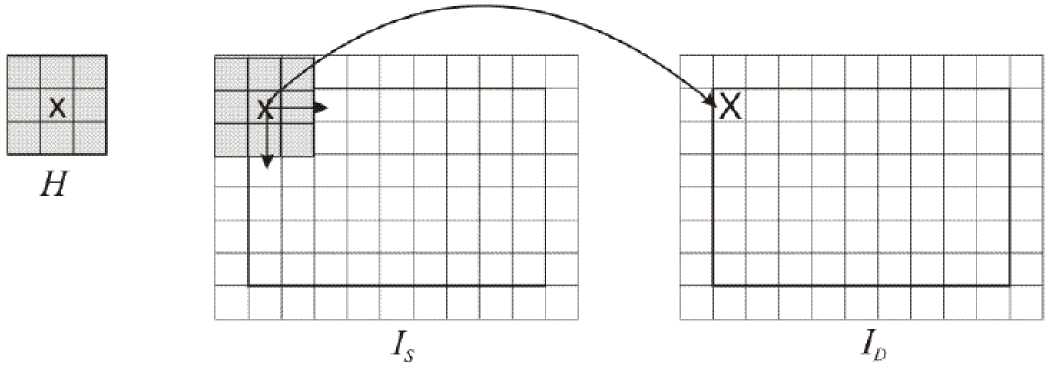

Fig. 1 Illustration of the convolution process.

The convolution kernels can have also non-symmetrical shapes (the central/reference element is not positioned in the center of symmetry).

Low-pass filters

Low-pass filters are used for image smoothing and noise reduction. Their effect is an averaging of the current pixel with the values of its neighbors, observable as a "blurring" of the output image (they allow to pass only the low frequencies of the image). All elements of the kernels used for low-pass filtering have positive values. Therefore, a common practice used to scale the result in the intensity domain of the output image is to divide the result of the convolution with the sum of the elements of the kernel:

$$I_D(x,y) = \frac{1}{C}\sum_{i=-k}^{k}\sum_{j=-k}^{k} H(i,j).I_s\,(x+i,y+j)$$

Where $C = \sum_{i=-k}^{k}\sum_{j=-k}^{k} H(i,j)$

Examples:

1) Averaging Filter(3X3):

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

2) Gaussian Filter(3X3):

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

High-pass filters

These filters will highlight regions with step intensity variations, such as edges (will allow to pass the high frequencies). The kernels used for edge detection have the sum of their elements equal to 0:

Examples:

1) Laplacian filters (edge detection) (3x3):

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

2)

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Algorithm:

1) Start
2) Read the gray scale image.
3) Enter the mask for smoothing operation.
4) Enter the mask for sharpening operation.
5) Obtain the size of original image and store it.
6) Scale the image by performing double operation.
7) Create four nested loops, outer two loops are in9tialized with respect to image size MXN and inner loops are with respect to 3X3 masks for smoothing and sharpening.
8) Perform convolution with the mask and the image and store the results in two variables.
9) Perform absolute operation i. e. (abs ()) to remove the negative values.
10) Rescale the image from 'double' to 'uint8'.
11) Display the final output image.
12) Compare the smoothened and sharpened image with the original image.
13) End.

Conclusion: