**EXPERIMENT NO. 3**

**PERFORM AND COMPARE FREQUENCY DOMAIN FILTERS**

**EXPERIMENT NO. 3**

**AIM:** To implement ideal, butterworth, Gaussian low pass filters in frequency domain and compare their performances

**OBJECTIVES:**

1. To apply and compare performance averaging filters of various size and cutoff frequency
2. To understand to convert image from spatial domain to frequency domain.
3. To see the frequency spectrum of the image.
4. To understand the concept of frequency domain filtering.

**EQUIPMENTS/SOFTWARE:** SCILAB 6.0.0

**THEORY:** Using low pass filters -

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  | | --- | --- | --- | | 1 | 1 | 1 | | 1 | 1 | 1 | | 1 | 1 | 1 | |  | |  |  |  | | --- | --- | --- | | 1 | 2 | 1 | | 2 | 4 | 2 | | 1 | 2 | 1 | |
|  |  |  |  |

The second mask, shown in Figure, yields a so-called weight average, thus giving more importance (weight) to some pixels at the expense of others.

Frequency domain filtering-

The reason for doing the filtering in the frequency domain is generally because it is computationally faster to perform two 2D Fourier transforms and filter multiplication in this domain than to perform convolution in the image (spatial) domain. Also convolution becomes more complex in spatial domain as filter size increases.

The transfer function of a Butterworth LPF of order ‘n’, and with cutoff frequency at a distance Do from its origin is defined as

The transfer function of a Gaussian LPF with cutoff frequency at a distance Do from its origin is defined as

**ALGORITHM:**

**Spatial domain filtering-**

1. Read the image.
2. Define LPF masks
3. Run the mask on the image.
4. See result of the filtering

**Frequency domain filtering-**

1. Read the input image and its size.
2. Obtain the padding parameters P and Q. Typically, we select P=2M and Q=2N
3. Form a padded image, fp(x,y) of size PxQ by appending the necessary number of zeroes to f(x,y)
4. Multiply fp(x,y) by (-1)x+y to center its transform
5. Obtain the Fourier transform of the image
6. Generate a Butterworth and Gaussian filter function, H1 and H2, the same size as the image (PxQ)
7. Multiply the transformed image by the filter:  
   G1=H1.\*F; G2=H2.\*F;
8. Obtain the real part of the inverse FFT of G.

**FUNCTIONS USED (MATLAB / SCILAB):**

1. imread
2. double
3. fft2
4. ifft2
5. real
6. imshow

**CONCLUSION:**