ASSIGNMENT-06(A)

Exercise 1:

Mean Filter or uniform filter

One of the simplest linear filters is implemented by a local averaging operation where the value of each pixel is replaced by the average of all the values in the local neighborhood:

$$h[i,j] = \frac{1}{M} \sum_{(k,l) \in N} f[k,l)$$

where Af is the total number of pixels in the neighborhood *N.* For example, taking a 3 X 3 neighborhood about |i, >| yields:

$$h[i,j] = \frac{1}{9} \sum_{k=i-1}^{i+1} \sum_{l=j-1}^{j+1} f[k,l].$$

Now if *g[i,j]* = 1/9 for every [•',» in the convolution mask, the convolution operation: reduces to the

local averaging operation This result shows that a mean filter

can be implemented as a convolution operation with equal weights in the convolution mask.

The size of the neighborhood JV controls the amount of filtering. A larger neighborhood, corresponding to a larger convolution mask, will result in a greater degree of filtering. As a trade-off for greater amounts of noise reduction. larser filters also result in a loss of image detail. '

When designing linear smoothing filters, the filter weights should be chosen so that the filter has a single peak, called the main lobe, and symmetry in the vertical and horizontal directions. A typical pattern of weights for a 3x3 smoothing filter *is*

Linear smoothing filters remove high-frequency components, and the sharp detail in the image is lost. For example, step changes will be blurred into gradual changes, and the ability to accurately localize a change will be sacrificed. A spatially varying filter can adjust the weights so that more smoothing is done in a relatively uniform area of the image, and little smoothing is done across sharp changes in the image.

It is used as a smoothing filter. A plot of this function is shown in Figure Gaussian functions have five properties that make them particularly useful in early vision processing. These properties indicate that the Gaussian smoothing filters are effective low-pass filters from the perspective of both the spatial and frequency domains, are efficient to implement, and can be

Mean Filter or uniform filter

One of the simplest linear filtere is implemented by a local averaging operation where the value of each pixel is replaced by the average of all the values in the local neighborhood:

where Af is the total number of pixels in the neighborhood *N.* For example, taking a 3 X 3 neighborhood about [i,j] yields:

Mbd-ijcE/M-

Now if *g[i,j]* = 1/9 for every [•',» in the convolution mask, the convolution operation: reduces to the

local averaging operation This result shows that a mean filter

can be implemented as a convolution operation with equal weights in the convolution mask.

The size of the neighborhood, Y controls the amount of filtering. A larger neighborhood, corresponding to a larger convolution mask, will result in a greater degree of filtering. As a trade-off for greater amounts of noise reduction. laser filters also result in a loss of image detail. '

When designing linear smoothing filters, the filter weights should be chosen so that the filter has a single peak, called the main lobe, and symmetry in the vertical and horizontal directions. A typical pattern of weights for a 3x3 smoothing filter is

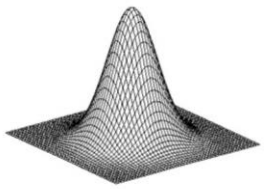
Exercise 2: The first derivative of a signal is the rate of change of y with x, that is, dy/dx, which is interpreted as the *slope* of the tangent to the signal at each point,

Important Gaussian Derivative Properties:

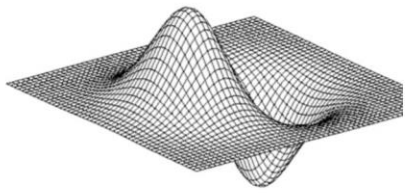
- Image Differentiation d/dx is a convolution on image I.
- Smoothing by Gaussian kernel g is a convolution on image I.
- 2D Gaussian Kernel is separable g = g^(1) * g^(2).
- Convolution is
    - Commutative I * g = g * f
    - Associative (I * g) * h = I * (g * h)

So,

$$\frac{d}{dx}(I * g) = I * \frac{d}{dx}g = (I * (\frac{d}{dx}g^{(1)})) * g^{(2)}$$



g



$\frac{d}{dx}g$

Exercise 4:

The shadow removal is done by multiplying the shadow region by a constant. Shadow edge correction is done to reduce the errors due to diffusion in the shadow boundary.

Method 1:

Step 1: a = readim('images\shading.ics','')

Step 2: b = gaussf(a,30)

Step 3: c = a – b

The final result is: