Department of Electronics Engineering

Image Enhancement-Neighborhood processing

Aim: - To implement low pass & high pass filtering operation in spatial domain on noisy color images using matlab/python.

Theory:-

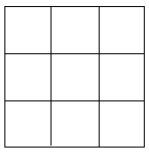
Image Enhancement -Spatial Domain
•Smoothing filter
•Median filter
•Sharpening filter

Neighborhood Averaging

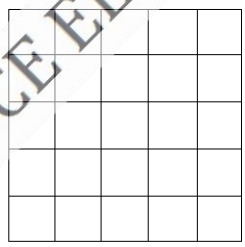
S = neighborhood of pixel (x,y)
M = number of pixels in neighborhood S

$$g(x,y) = (1/M) \sum_{(n,m) \in S} f(n,m)$$

Neighborhoods:



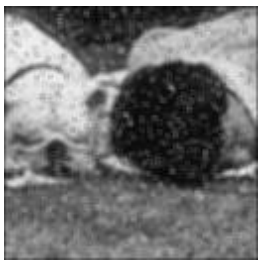
3 x 3



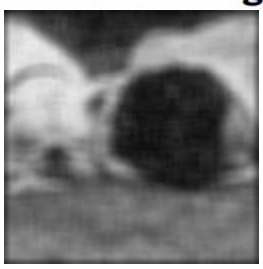
5 x 5

Neighborhood Averaging - Example

Image + Salt & Pepper Noise


3 X 3 Average


5 X 5 Average


7 X 7 Average


Median

Neighborhood processing Operation:

A, B = images

B is typically smaller than A and is called the **mask**.

Convolution - 2 Dimensions

A * B

$$(A * B)(x,y) = \sum_i \sum_j A(i,j)\, B(x-i,y-j)$$

Grayscale Smoothing

Grayscale averaging = convolution with:

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

3 X 3

1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25

5 X 5

"Soft" Averaging: Convolution with a Gaussian

$$\frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

Discrete case:

(1/8) x

0	1	0
1	2	1
0	1	0

(1/81) x

1	2	3	2	1
2	4	6	4	2
3	6	9	6	3
2	4	6	4	2
1	2	3	2	1

Median Filtering

S = neighborhood of pixel (x,y)

New value at (x,y) = $\underset{(x,y)\in S}{\text{median}}\{f(x,y)\}$

30	10	20
10	250	25
20	25	30

10, 10, 20, 20, 25, 25, 30, 30, 250

median

Median + Average: average the k central values.

10, 10, 20, 20, 25, 25, 30, 30, 250

24

Sharpening Spatial Filters

• To highlight fine detail in an image or to enhance detail that has been blurred, either in error or as a natural effect of a particular method of image acquisition.

Derivative operator

• The strength of the response of a derivative operator is proportional to the degree of discontinuity of the image at the point at which the operator is applied.

First and Second-order derivative of 2D
• when we consider an image function of two variables, f(x, y), at which time we will dealing with partial derivatives along the two spatial axes.

Gradient operator (linear operator)

$$\nabla f = \frac{\partial f(x,y)}{\partial x \partial y} = \frac{\partial f(x,y)}{\partial x} + \frac{\partial f(x,y)}{\partial y}$$

Laplacian operator (non-linear)

$$\nabla^2 f = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$$

Discrete form of Laplacian

from

$$\frac{\partial^2 f}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y)$$

$$\frac{\partial^2 f}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y)$$

$$\nabla^2 f = [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)]$$

High Pass filter Mask

0	1	0	1	1	1
1	−4	1	1	−8	1
0	1	0	1	1	1

0	−1	0	−1	−1	−1
−1	4	−1	−1	8	−1
0	−1	0	−1	−1	−1

High pass filter output:-

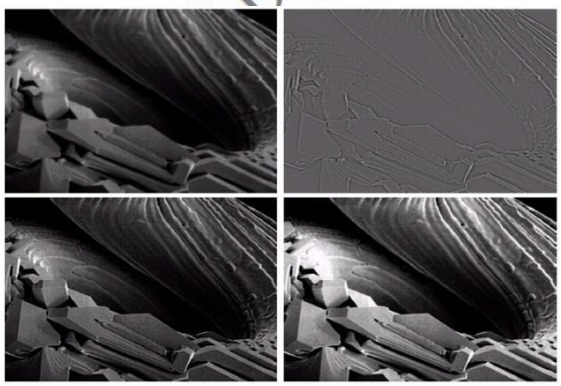
Department of Electronics Engineering

High-Boost (High-Frequency Emphasis) Filtering

- Result of High-Pass is not always desirable.
- H-P may be expressed as (see exVI):
  High-Pass = Original Image − Low-Pass
- High-Boost: Multiply original image by amplification factor A (= 1 for H-P):
  High-Boost = A(Original) − Low-Pass
  = (A-1)(Original) + Original − Low-Pass
  = (A-1)(Original) + High-Pass
- When A>1, part of original is added back to result, partially restoring low-frequency components lost to high-pass filtering.

0	−1	0	−1	−1	−1
−1	$A+4$	−1	−1	$A+8$	−1
0	−1	0	−1	−1	−1

(a)-Image
(b)-with A=0
(c)-with A=1
(d)-with A=1.7



Implementation Instructions:-

1) For averaging Low pass filter-Add gaussian noise to an image (using IMNOISE function in matlab). Now process it with averaging mask & observe the result.
2) For Median Low pass filter- Add salt & pepper noise to an image (using IMNOISE function in matlab). Now process it with median mask & observe the result.
3) For High pass filter- Process the image with high pass filter mask & high boost filter mask & observe the result.

Code

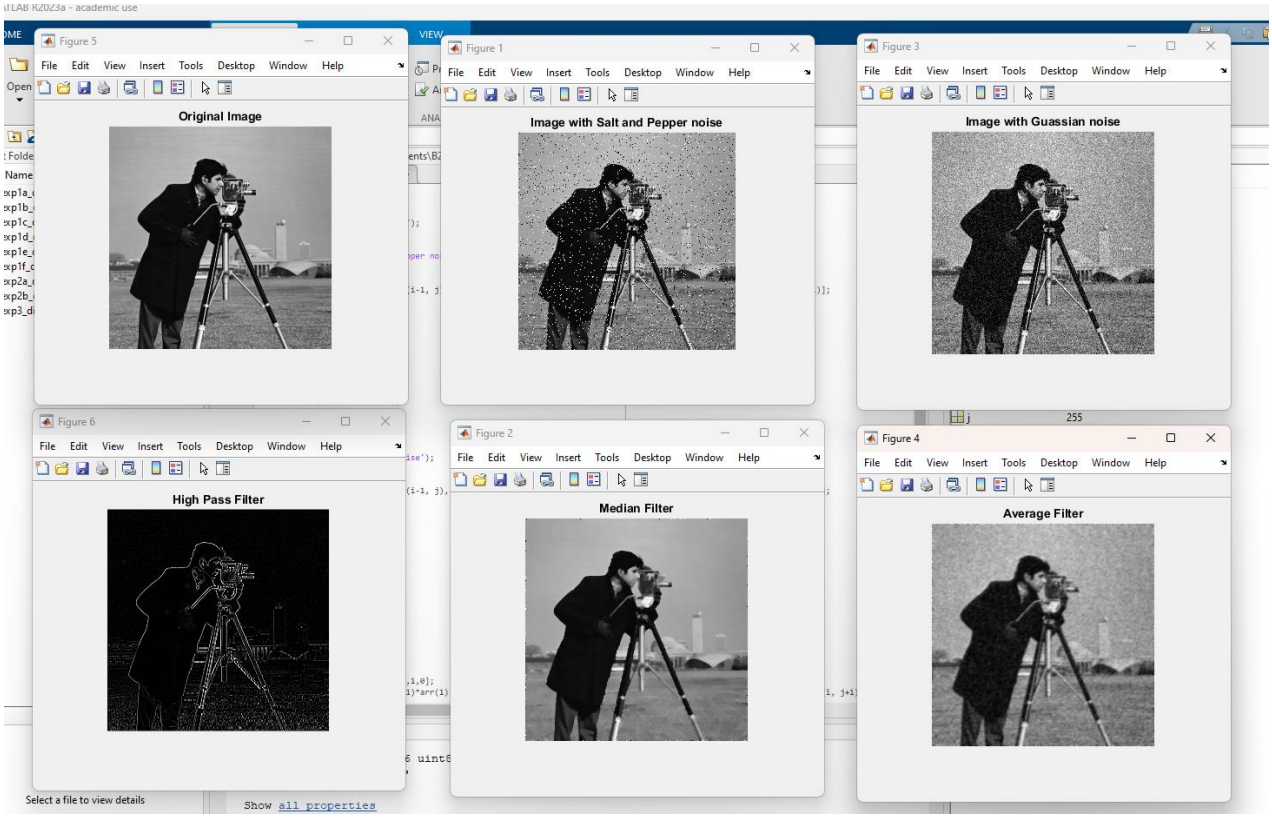
```
%Experiment 3 - Image Enhancement Neighbourhood Processing

%Krisha Lakhani - 60001200097
%Salt and Pepper
a = imread('cameraman.tif');
size(a);
b = imnoise(a, "salt & pepper");
figure(1);
imshow(b);
title('Image with Salt and Pepper noise');
for i = 2:255
    for j = 2:255
        arr = [b(i-1, j-1), b(i-1, j), b(i-1, j+1), b(i, j-1), b(i, j), b(i, j+1), b(i+1, j-1),b(i+1, j),b(i+1, j+1)];
        M = median(arr);
        b(i,j) = M;
    end
end
figure(2);
imshow(b);
title('Median Filter');
```

```
%Krisha Lakhani - 60001200097
%Gaussian
c = imread('cameraman.tif');
size(c);
d = imnoise(c, "gaussian");
figure(3);
imshow(d);
title('Image with Guassian noise');
for i = 2:255
    for j = 2:255
        arr = [d(i-1, j-1), d(i-1, j), d(i-1, j+1), d(i, j-1), d(i, j), d(i, j+1), d(i+1, j-1),d(i+1, j),d(i+1, j+1)];
        A = mean(arr);
        d(i,j) = A;
    end
end
figure(4);
imshow(d);
title('Average Filter');
```

```
%Krisha Lakhani - 60001200097
%HPF
e = imread('cameraman.tif');
f = double(e);
figure(5);
imshow(e);
title('Original Image');
for i = 2:255
    for j = 2:255
        arr = [0,1,0,1,-4,1,0,1,0];
        newi(i,j) = f(i-1, j-1)*arr(1) + f(i-1, j)*arr(2) + f(i-1, j+1)*arr(3)+f(i, j-1)*arr(4) + f(i, j)*arr(5) + f(i, j+1)*arr(6) + f(i+1, j-1)*arr(7)
    end
end
figure(6);
imshow(uint8(newi));
title('High Pass Filter');
```

Output



Conclusion:

We have successfully implemented neighborhood processing by inducing salt and pepper and Gaussian noise and removing them with median and averaging filters respectively, as well as the image has been passed through a high pass filter to remove low-frequency pixel values.