LAB Manual

PART A

(PART A : TO BE REFFERED BY STUDENTS)

**Experiment No.04**

**A.1 Aim:**

To write a program to enhance the quality of an image (on your own photograph) by noise removal.

**A.2 Prerequisite:**

1 Matlab programming syntax (Refer the Matlab manual).

2. Knowledge of filtering techniques.

2. Availability of Soft copy of your Photograph for experiment.

**A.3 Outcome:**

**After successful completion of this experiment students will be able to**

1. Understand the fundamentals of Noise and its effects in digital images
2. Design HPF, LPF and meadian filters.
3. Remove noise from given image using following Filtering methods.
4. Histogram Equalization
5. Low Pass Filtering
6. High Pass Filtering
7. Median Filtering.
8. Identify applications of filtering techniques studied.

**A.4 Theory:**

**Filtering**

**Noise:**

Images are corrupted by random variations in intensity values called noise due to non-perfect camera acquisition or environmental conditions.

**Common Types of Noise**

* **Salt and pepper noise:** random occurrences of both black and white intensity values
* **Impulse noise:** random occurrences of white intensity values
* **Gaussian noise:** impulse noise but its intensity values are drawn from a Gaussian distribution. Noise Intensity Value is given by following equation:

…… Equation (1)

k: random intensity value

**Effects of noise on digital images:**

Presence of particular type of noise in an image may deteriorate image quality to certain level. The level of deterioration depends on the density of noise

**Noise Filtering:**

**Basic Idea**: replace each pixel intensity value with an new value taken over a neighborhood of fixed size

**The size of the filter controls degree of smoothing**

* large filter » large neighborhood » intensive smoothing

**Low Pass Filter:**

Low pass filtering removes the high frequency contents from the image. It is used to remove noise present in the image. Noise, is normally a highn frequency signal abd low pass filtering eliminates the noise.

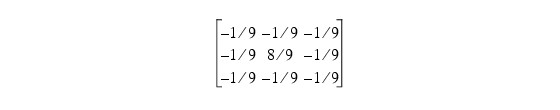
A low pass filter is the basis for most smoothing methods. An image is smoothed by decreasing the disparity between pixel values by averaging nearby pixels.

Using a low pass filter tends to retain the low frequency information within an image while reducing the high frequency information. An example is an array of ones divided by the number of elements within the kernel, such as the following 3 by 3 kernel: 

**High Pass Filter:**

High pass filtering eliminates the low frequency regions while retaining or enhancing the hogh frequency components. An image, which is high passed, would have no background and would have enhanced edges.

A high pass filter is the basis for most sharpening methods. An image is sharpened when contrast is enhanced between adjoining areas with little variation in brightness or darkness. A high pass filter tends to retain the high frequency information within an image while reducing the low frequency information. The kernel of the high pass filter is designed to increase the brightness of the center pixel relative to neighboring pixels. The kernel array usually contains a single positive value at its center, which is completely surrounded by negative values. The following array is an example of a 3 by 3 kernel for a high pass filter:



**Median Filter**

Replace each pixel value with the median of the gray values in the region of the pixel:

1. Take a 3 x 3 (or 5 x 5 etc.) region centered around pixel (i,j)

2. Sort the intensity values of the pixels in the region into ascending order

3. Seletc the middle value as the new value of pixel (i,j)

Computation of Median Values:



**Figure 1. Computation of Median values**

**A.5 Procedure/Algorithm:**

**A.5.1 TASK 1:**

**Image Filtering**

1. Read the input image.

2. Add noise to the input image. (Make use of Use imnoise( ) ).

3. Design your own HPF, LPF and Meadian filters

4. Apply LPF, HPF and Median Filter to the image.

4. Display the original and the output image.

5. Observe the output and complete PART B of lab manual.

6. Add salt and pepper noise to the input image.

7. Apply LPF, HPF and Median Filter to the image obtained in step 6.

8. Display the original and the output images.

9. Observe the output and complete PART B of lab manual.

10. Save and close the file and name it as **EX4\_Task1\_your Roll no.m**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PART B

(PART B : TO BE COMPLETED BY STUDENTS)

***(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no Black board access available)***

|  |  |
| --- | --- |
| Roll No. B032 | Name: Naman Garg |
| Class : Btech CS B | Batch : B2 |
| Date of Experiment: | Date of Submission |
| Grade : |  |

**B.1 Software Code written by student:**

**FILTERING**

img = imread("gg.jpg");

img1 = rgb2gray(img);

img2 = imresize(img1, [256,256]);

image3 = imnoise(img2, "gaussian");

img3 = imnoise(img2, "salt & pepper");

img4 = zeros(258, 258);

img4(1:256, 1:256) = img2;

img5 = zeros(258, 258);

img5(1:256, 1:256) = img3;

image4 = zeros(258, 258);

image4(1:256, 1:256) = image3;

img6 = lpf(img4);

img7 = hpf(img4);

img8 = mf(img4);

img9 = lpf(img5);

img10 = hpf(img5);

img11 = mf(img5);

img12 = lpf(image4);

img13 = hpf(image4);

img14 = mf(image4);

img4 = uint8(img4(2:255,2:255));

img5 = uint8(img5(2:255,2:255));

image4 = uint8(image4(2:255,2:255));

subplot(3,4,1), imshow(img4), title("Input Image w/o Noise");

subplot(3,4,2), imshow(img6), title("Low Pass Filter");

subplot(3,4,3), imshow(img7), title("High Pass Filter");

subplot(3,4,4), imshow(img8), title("Median Filter");

subplot(3,4,5), imshow(img5), title("Input Image w Salt & Pepper");

subplot(3,4,6), imshow(img9), title("Low Pass Filter");

subplot(3,4,7), imshow(img10), title("High Pass Filter");

subplot(3,4,8), imshow(img11), title("Median Filter");

subplot(3,4,9), imshow(image4), title("Input Image w Gaussian");

subplot(3,4,10), imshow(img12), title("Low Pass Filter");

subplot(3,4,11), imshow(img13), title("High Pass Filter");

subplot(3,4,12), imshow(img14), title("Median Filter");

function img3 = lpf(img)

img2 = zeros(256, 256);

for i = 2:256

for j = 2:256

img2(i,j) = double((img(i-1,j-1)+img(i-1,j)+img(i-1,j+1)+img(i,j-1)+img(i,j)+img(i,j+1)+img(i+1,j-1)+img(i+1,j)+img(i+1,j+1))\*1/9);

end

end

img3 = uint8(img2(2:255,2:255));

end

function img3 = hpf(img)

img2 = zeros(256, 256);

for i = 2:256

for j = 2:256

img2(i,j) = double((img(i-1,j-1)+img(i-1,j)+img(i-1,j+1)+img(i,j-1)+img(i,j+1)+img(i+1,j-1)+img(i+1,j)+img(i+1,j+1))\*(-2/9)+(img(i,j)\*12/9));

end

end

img3 = uint8(img2(2:255,2:255));

end

function img3 = mf(img)

img2 = zeros(256, 256);

for i = 2:256

for j = 2:256

l1 = [img(i-1,j-1),img(i-1,j),img(i-1,j+1),img(i,j-1),img(i,j+1),img(i+1,j-1),img(i+1,j),img(i+1,j+1),img(i,j)];

l2 = sort(l1);

img2(i,j) = l2(5);

end

end

img3 = uint8(img2(2:255,2:255));

end

**HISTOGRAM EQUALIZATION**

i = imread("gg.jpg");

i = rgb2gray(i)

i2=i;

GIm = i;

i = imread("gg.jpg");

i = rgb2gray(i)

ih = histeq(i,255);

numofpixels=size(GIm,1)\*size(GIm,2);

HIm=uint8(zeros(size(GIm,1),size(GIm,2)));

freq=zeros(256,1);

probf=zeros(256,1);

probc=zeros(256,1);

cum=zeros(256,1);

output=zeros(256,1);

for i=1:size(GIm,1)

for j=1:size(GIm,2)

value=GIm(i,j);

freq(value+1)=freq(value+1)+1;

probf(value+1)=freq(value+1)/numofpixels;

end

end

sum=0;

no\_bins=255;

for i=1:size(probf)

sum=sum+freq(i);

cum(i)=sum;

probc(i)=cum(i)/numofpixels;

output(i)=round(probc(i)\*no\_bins);

end

for i=1:size(GIm,1)

for j=1:size(GIm,2)

HIm(i,j)=output(GIm(i,j)+1);

end

end

subplot(2,4,1),imshow(i2), title('Original Image');

subplot(2,4,2),imshow(HIm), title('Histogram Equalized Image self code');

subplot(2,4,3),imhist(i2), title('Histogram of Original Image self code');

subplot(2,4,4),imhist(HIm), title('histogram of Equalized Image self code');

subplot(2,4,5),imshow(i2), title('Original Image');

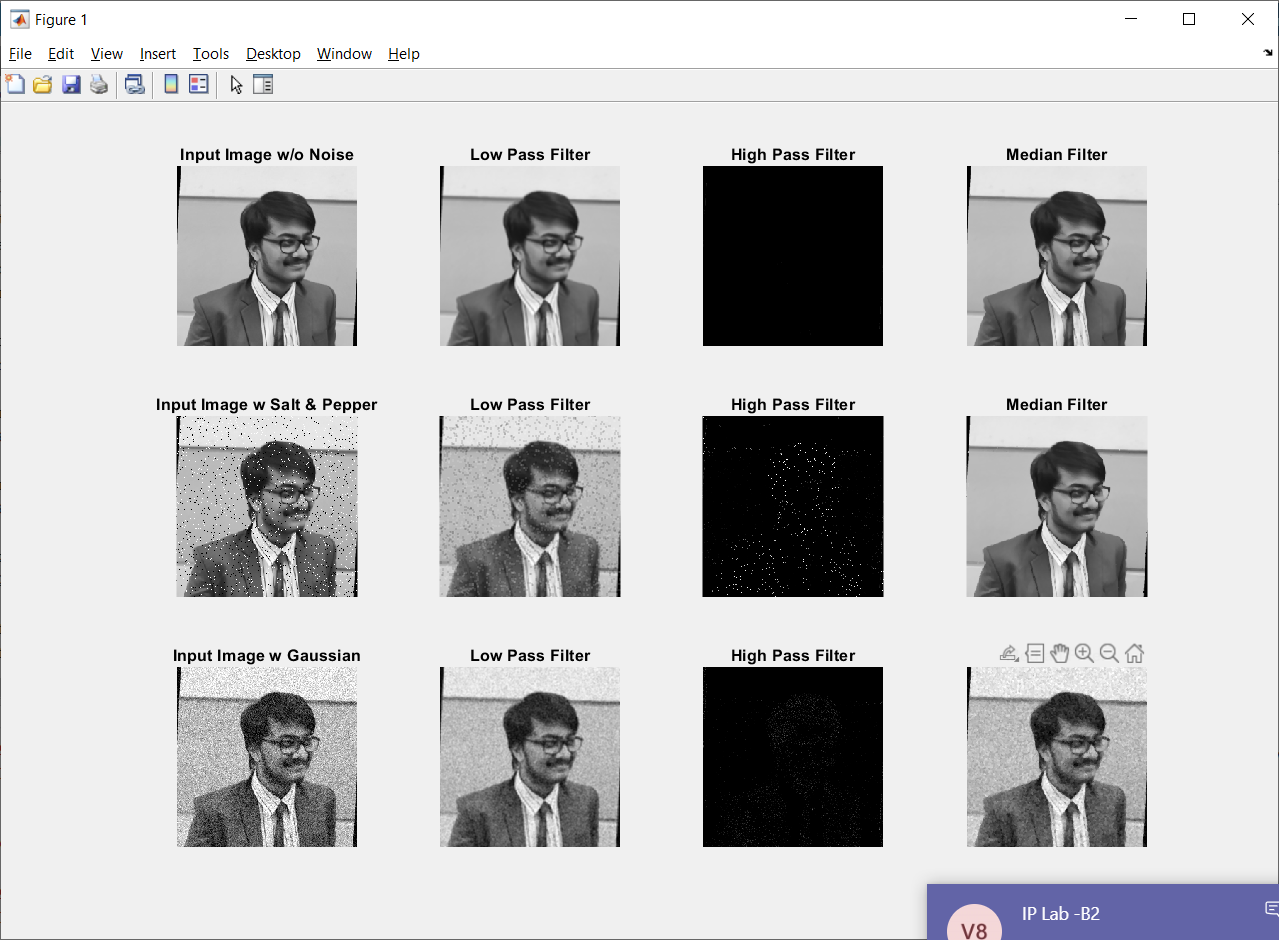
subplot(2,4,6),imshow(ih), title('Histogram Equalized Image inbuillt');

subplot(2,4,7),imhist(i2), title('Histogram of Original Image inbuillt');

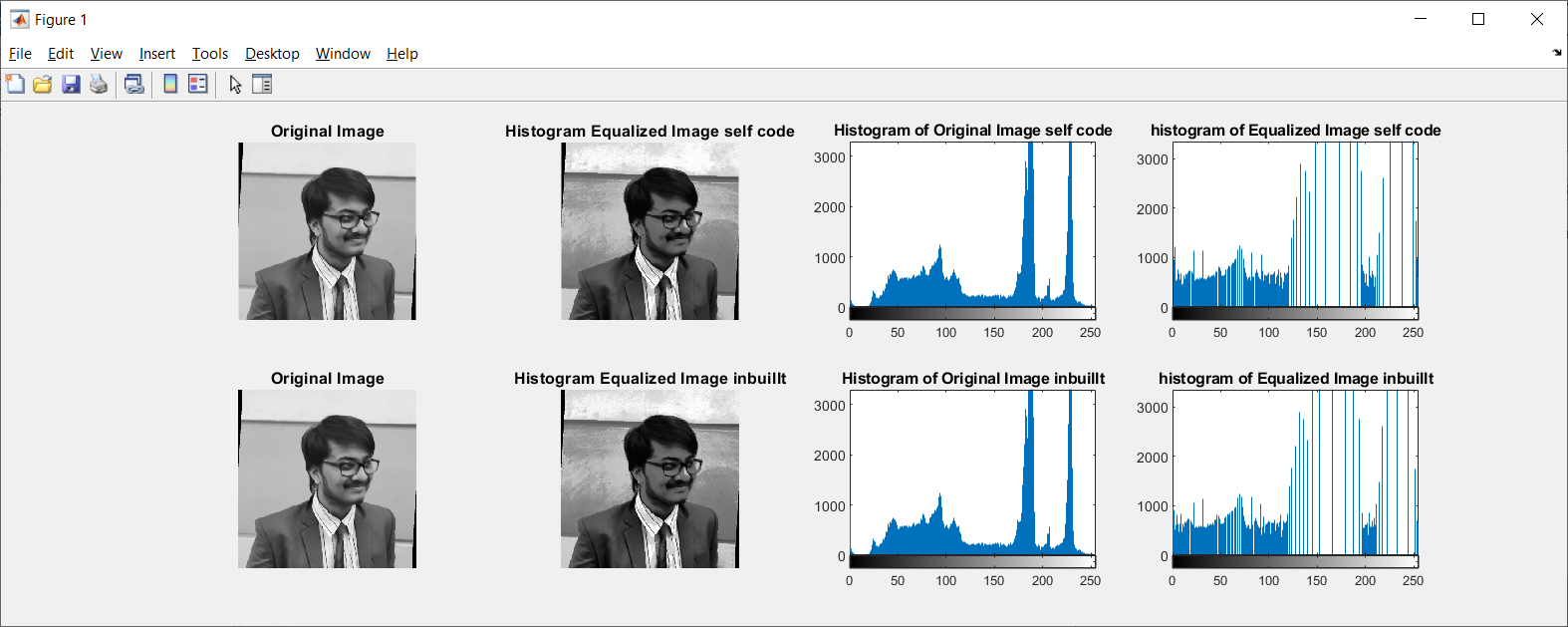
subplot(2,4,8),imhist(ih,255), title('histogram of Equalized Image inbuillt');

**B.2 Input and Output:**

**FILTERING**



**HISTOGRAM EQUALIZAION**



**B.3 Observations and learning:**

* **FILTERING**
* adding the noise:
  + I added salt and pepper noise to the image, this made the image grainy by randomly changing some pixels to be lighter or darker Gaussian noise is statistical noise having a probability density function (PDF) equal to that of the normal distribution, which is also known as the Gaussian distribution. In other words, the values that the noise can take on are Gaussian-distributed.
* Low Pass Filter
  + The low pass filter removed the high frequency contents used to remove noise,
  + We used 3x3 filter each pixel having value 1/9 of the original image.
  + It removed some noise and had a blurring effect
* High Pass Filter
  + The high pass filter removed the low frequency contents and enhances the high frequency contents
  + Thus we used a 3x3 filter with the central pixel having 8/9 value and the surrounding pixels have 1/9 most of the observed outputs were black we need a light image for this filter to work, as the darkest pixel becomes the lightest one here
* Median Filter
  + Median filtering replaces each pixel value with the median of the gray values in the region of the pixel (a 3 x 3 region centered around pixel in our case). It removes noise while preserving the edges at the same time.
  + Among all the three filtering techniques used, median filtering works the best in removing noise. In the case of salt and pepper noise, the result was almost completely resembling the original image
* **HISTOGRAM EQUALIZATION:**
* The main takeaway for me was that HE improved the contrast in the image
* It accomplishes this by effectively spreading out the most frequent intensity values

**B.4 Conclusion:**

* Understood the fundamentals of Noise and its effects in digital images.
* Designed HPF, LPF and median filters.
* Removed noise from given image using following Filtering methods.

Low Pass Filtering

High Pass Filtering

Median Filtering.

* Identified applications of filtering techniques studied.

**B.5 Question of Curiosity**

Q1. Which are the other methods of noise removal?

Q2.What are different types of noises possible to get in the image? Is there any specific methods/filters etc. to get rid of these noises

A1.) The other methods of noise removal are:

1. **Mean Filter:**

The mean filter is a simple sliding window spatial filter that replaces the center value in the window with the average of all the neighboring pixel values including itself. This process is repeated for all pixel values in the image. By doing this, it replaces pixels that are unrepresentative of their surroundings. The window is usually square but it can be of any shape.

1. **Adaptive Filter:**

Adaptive filter is performed on the degraded image that contains original image and noise. The mean and variance are the two statistical measures that a local adaptive filter depends with a defined m x n window region. The adaptive filter is more selective than a comparable linear filter, preserving edges and other high-frequency parts of an image

1. **Wiener Filter:**

The main aim of this technique is to filter out noise that has corrupted the signal. It is kind of statistical approach. For the designing of this filter one should know the spectral properties of the original signal ,the noise and linear time-variant filter whose output should be as close as to the original as possible. The Wiener filter minimizes the mean square error between the estimated random process and the desired process

1. **Max and Min Filter:**

Minimum and maximum filters, also known as erosion and dilation filters, respectively, are morphological filters that work by considering a neighborhood around each pixel. From the list of neighbor pixels, the minimum or maximum value is found and stored as the corresponding resulting value. Finally, each pixel in the image is replaced by the resulting value generated for its associated neighborhood. If we apply max and min filters alternately they can remove certain kind of noise, such as salt-and-pepper noise very efficiently.

1. **Bilateral Filter:**

A bilateral filter is a [non-linear](https://en.wikipedia.org/wiki/Non-linear), [edge-preserving](https://en.wikipedia.org/wiki/Edge-preserving_smoothing), and [noise-reducing](https://en.wikipedia.org/wiki/Noise_reduction) [smoothing](https://en.wikipedia.org/wiki/Smoothing) [filter for images](https://en.wikipedia.org/wiki/Digital_image_processing). It replaces the intensity of each pixel with a weighted average of intensity values from nearby pixels. This weight can be based on a Gaussian distribution. Crucially, the weights depend not only on Euclidean distance of pixels, but also on the radiometric differences (e.g., range differences, such as color intensity, depth distance, etc.). This preserves sharp edges.

1. **Midpoint Filter:**

The midpoint filter simply computes the midpoint between the maximum and minimum values in the area encompassed by the filter.

1. **Alpha- trimmed mean Filter:**

Alpha-trimmed mean filter is windowed filter of nonlinear class, its nature is hybrid of the mean and median filters. The basic idea behind filter is for any element of the signal (image) look at its neighborhood, discard the most atypical elements and calculate mean value using the rest of them.

A2.) The different types of noises and the filters that can remove them effectively:

1. Salt & Pepper Noise - Max and Min Filter
2. Gaussian Noise - Weiner Filter
3. Poisson Noise - Bilateral Filter
4. Speckle Noise - Median Filter

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*