K. J. Somaiya College of Engineering, Mumbai-77
Somaiya Vidyavihar University

Batch: A-4 **Roll No.:** 16010122151

Experiment No. 8

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

Title: Apply neighbourhood processing techniques: low pass, high pass and median filtering in spatial domain on a digital image

Objective: To learn and understand the effects of filtering in spatial and frequency domain on images using Matlab.

Expected Outcome of Experiment:

CO	Outcome
CO4	Design & implement algorithms for digital image enhancement, segmentation & restoration.

Books/ Journals/ Websites referred:

1. http://www.mathworks.com/support/
2. www.math.mtu.edu/~msgocken/intro/intro.html.
3. R. C.Gonsales R.E.Woods, “Digital Image Processing”, Second edition, Pearson Education
4. S.Jayaraman, S Esakkirajan, T Veerakumar “Digital Image Processing “Mc Graw Hill.
5. S.Sridhar,”Digital Image processing”, oxford university press, 1st edition."

Pre Lab/ Prior Concepts:

Filtering in Spatial Domain:

Low pass filtering as the name suggests removes the high frequency content from the image. It is used to remove noise present in the image. Mask for the low pass filter is:

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

One important thing to note from the spatial response is that all the coefficients are positive. We could also use 5 x 5 or 7 x 7 mask as per our requirement. We place a 3 x 3 mask on the image. We start from the left hand top corner. We cannot work with the borders and hence are normally left as they are. We then multiply each component of the image with the corresponding value of the mask. Add these values to get the response. Replace the centre pixel of the o/p image with these responses. We now shift the mask towards the right till we reach the end of the line and then move it downwards.

High pass filtering as the name suggests removes the low frequency content from the image. It is used to highlight fine detail in an image or to enhance detail that has been blurred. Mask for the high pass filter is:

-1/9	-1/9	-1/9
-1/9	8/9	-1/9
-1/9	-1/9	-1/9

One important thing to note from the spatial response is that sum of all the coefficients is zero. We could also use 5 x 5 or 7 x 7 mask as per our requirement. We place a 3 x 3 mask on the image. We start from the left hand top corner. We cannot work with the borders and hence are normally left as they are. We then multiply each component of the image with the corresponding value of the mask. Add these values to get the response. Replace the centre pixel of the o/p image with these responses. We now shift

the mask towards the right till we reach the end of the line and then move it downwards.

Median filtering is a signal processing technique developed by tukey that is useful for noise suppression in images. Here the input pixel is replaced by the median of the pixels contained in the window around the pixel. The median filter disregards extreme values and does not allow them to influence the selection of a pixel value which is truly representative of the neighbourhood.

Implementation Details:

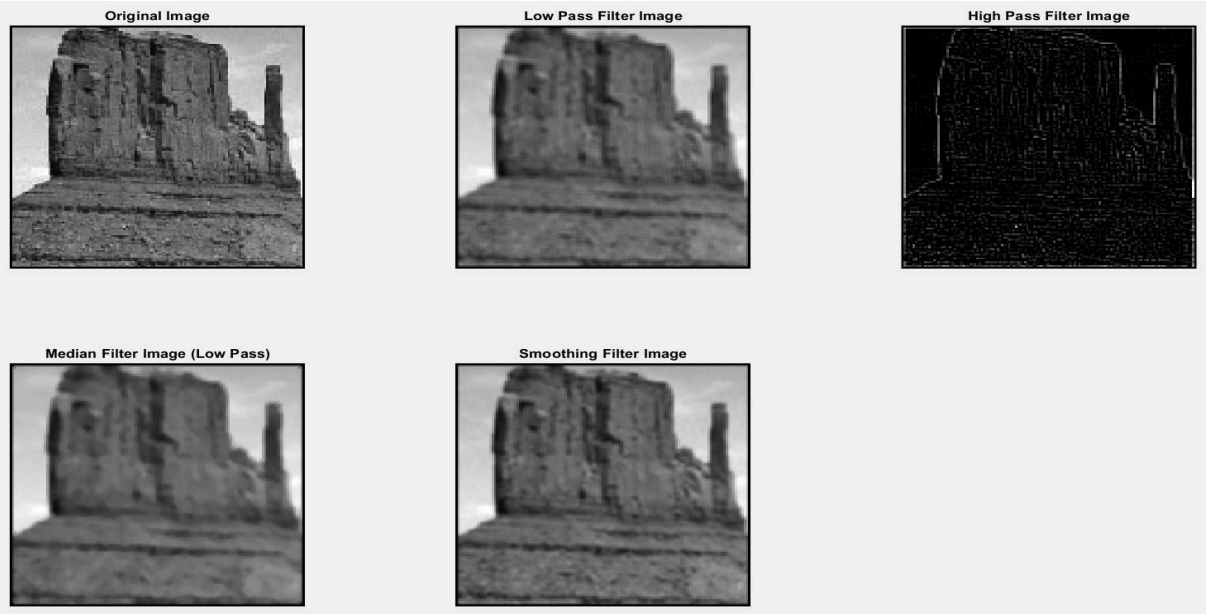
Write Algorithm and Matlab commands used:

CODE:

```
input_real = imread('rockbmp.jpg');
input = rgb2gray(input_real);
padSize = [1, 1];
inputPadded = zeros(size(input) + 2 * padSize);
inputPadded(2:end-1, 2:end-1) = in`          put;
input = inputPadded;
[rows, columns] = size(input);

lowpass = [1 1 1; 1 1 1; 1 1 1] / 9;
result_img_lowpass = zeros(rows, columns);
for integer1 = 2:rows-1
  for integer2 = 2:columns-1
     new_matrix = input(integer1-1:integer1+1, integer2-1:integer2+1);
     result_img_lowpass(integer1,integer2) = sum(sum(lowpass .* new_matrix));
  end
  end

highpass = [-1 -1 -1; -1 8 -1; -1 -1 -1] / 9;
result_img_highpass = zeros(rows, columns);
for integer1 = 2:rows-1
  for integer2 = 2:columns-1
     new_matrix = input(integer1-1:integer1+1, integer2-1:integer2+1);
     temp_sum = sum(sum(highpass .* new_matrix));
     result_img_highpass(integer1,integer2) = max(temp_sum, 0);
  end
end

smoothing = [1 2 1; 2 4 2; 1 2 1] / 16;
result_img_smoothing = zeros(rows, columns);
for integer1 = 2:rows-1
  for integer2 = 2:columns-1
     new_matrix = input(integer1-1:integer1+1, integer2-1:integer2+1);
     result_img_smoothing(integer1,integer2) = sum(sum(smoothing .* new_matrix));
  end
```

```
end

result_img_lowpass = uint8(255 * normalizeImage(result_img_lowpass));
result_img_highpass = uint8(255 * normalizeImage(result_img_highpass));
result_img_smoothing = uint8(255 * normalizeImage(result_img_smoothing));
result_img_lowpass_median = customMedfilt2(result_img_lowpass);

figure;

subplot(2,3,1), imshow(input, []); title('Original Image');
subplot(2,3,2), imshow(result_img_lowpass, []); title('Low Pass Filter Image');
subplot(2,3,3), imshow(result_img_highpass, []); title('High Pass Filter Image');
subplot(2,3,4), imshow(result_img_lowpass_median, []); title('Median Filter Image (Low Pass)');
subplot(2,3,5), imshow(result_img_smoothing, []); title('Smoothing Filter Image');

set(gcf, 'Position', [100, 100, 800, 500]);
set(gca, 'LooseInset', max(get(gca,'TightInset'), 0));

function output = customMedfilt2(input)
  [rows, columns] = size(input);
  output = uint8(zeros(rows, columns));
  for integer1 = 2:rows-1
     for integer2 = 2:columns-1
        window = input(integer1-1:integer1+1, integer2-1:integer2+1);
        output(integer1,integer2) = median(window(:));
     end
     end
     end

function output = normalizeImage(input)
  minimum_value = min(input(:));
  maximum_value = max(input(:));
  output = (input - minimum_value) / (maximum_value - minimum_value);
end
```

OUTPUT:



Conclusion:- We learnt about different filters associated with low pass, high pass, smoothing and median filter, studied them formulating and through a problem solving demonstration, and then implemented them using MATLAB.

Post Lab Descriptive Questions

1. List & explain different types of noise associated with a digital signal.

Noise in a digital signal refers to unwanted variations or distortions that can degrade the quality of the signal. Here are some common types:

a) Gaussian Noise (White Noise)

- **Description**: Random variations in intensity due to electronic circuit interference or sensor noise.
- **Source**: Thermal noise in sensors or electronic components.
- **Effect**: Blurs image details and adds graininess.
- **Example**: A digital photograph taken in low light may appear grainy due to sensor noise.

b) Salt-and-Pepper Noise (Impulse Noise)

- **Description**: Appears as random black and white pixels in an image.
- **Source**: Transmission errors, bit flips, or faulty sensor pixels.
- **Effect**: Causes sharp intensity changes that can make object boundaries unclear.**Example**: A corrupted scanned document with white and black specks.

c) Speckle Noise

- **Description**: Grainy texture affecting coherent imaging systems like ultrasound or radar.
- **Source**: Interference of reflected waves in imaging devices.
- **Effect**: Reduces contrast and clarity, making edges difficult to detect.
- **Example**: Ultrasound scans often have a speckled pattern due to wave interference.

d) Poisson Noise (Shot Noise)

- **Description**: Occurs due to random fluctuations in the number of photons hitting the sensor.
- **Source**: Low-light conditions in imaging sensors.
- **Effect**: Creates pixel intensity variations.
- **Example**: Night photography with long exposures may show brightness variations.

e) Periodic Noise

- **Description**: Regular noise patterns, often appearing as repetitive artifacts in images.
- **Source**: Electrical interference from power sources or scanner defects.
- **Effect**: Creates unwanted bands or waves across the image.
- **Example**: A scanned image with unwanted horizontal or vertical lines.

2. Explain with the help of an example how filtering helps in enhancing the quality of an image.

Example: Removing Salt-and-Pepper Noise Using a Median Filter

Salt-and-pepper noise can degrade image quality by introducing unwanted black and white specks. A **median filter** is an effective method to remove such noise while preserving image edges.

Process of Filtering

1. **Original Image**: A grayscale image contains noise, making object boundaries unclear.
2. **Applying a Median Filter**: A sliding window (e.g., 3×3 kernel) moves across the image.
3. **Pixel Replacement**: Each pixel is replaced by the median value of the surrounding pixels.
4. **Enhanced Image**: The noise is removed, but the edges and details remain intact.Comparison Before & After Filtering
5. **Before Filtering**: The image appears corrupted with random black and white specks.
6. **After Median Filtering**: The specks are removed, and the image is smooth but retains sharp edges