**Batch: B2 Roll No.:16010122221**

**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:



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:



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:**

**Commands used:**

|  |  |
| --- | --- |
| imread() | Read an image |

|  |  |
| --- | --- |
| rgb2gray() | Convert to grayscale |

|  |  |
| --- | --- |
| conv2() | Apply 2D convolution (filtering) |

|  |  |
| --- | --- |
| ones() | Create a matrix of ones (for averaging filter) |

|  |  |
| --- | --- |
| zeros() | Create a zero matrix (for padding) |

|  |  |
| --- | --- |
| uint8() | Convert to 8-bit image format |

|  |  |
| --- | --- |
| abs() | Take absolute value (for high-pass filtering) |

|  |  |
| --- | --- |
| median() | Compute median (for manual median filtering) |

|  |  |
| --- | --- |
| figure | Create a new figure window |

|  |  |
| --- | --- |
| subplot() | Divide figure into subplots |

|  |  |
| --- | --- |
| imshow() | Display an image |

|  |  |
| --- | --- |
| title() | Add a title to a subplot |

**ALGO:**

input\_real = imread('rock.jpg');

gray\_image = rgb2gray(input\_real);

lowpass\_filter = ones(3,3) / 9;

highpass\_filter = [-1 -1 -1; -1 8 -1; -1 -1 -1];

lowpass\_result = conv2(double(gray\_image), lowpass\_filter, 'same');

highpass\_result = abs(conv2(double(gray\_image), highpass\_filter, 'same'));

lowpass\_result = uint8(lowpass\_result);

highpass\_result = uint8(highpass\_result);

[rows, cols] = size(gray\_image);

padded\_lowpass = zeros(rows+2, cols+2, 'uint8');

padded\_highpass = zeros(rows+2, cols+2, 'uint8');

padded\_lowpass(2:end-1, 2:end-1) = lowpass\_result;

padded\_highpass(2:end-1, 2:end-1) = highpass\_result;

padded\_lowpass(1, 2:end-1) = lowpass\_result(1, :);

padded\_lowpass(end, 2:end-1) = lowpass\_result(end, :);

padded\_lowpass(:, 1) = padded\_lowpass(:, 2);

padded\_lowpass(:, end) = padded\_lowpass(:, end-1);

padded\_highpass(1, 2:end-1) = highpass\_result(1, :);

padded\_highpass(end, 2:end-1) = highpass\_result(end, :);

padded\_highpass(:, 1) = padded\_highpass(:, 2);

padded\_highpass(:, end) = padded\_highpass(:, end-1);

filtered\_lowpass = zeros(rows, cols, 'uint8');

filtered\_highpass = zeros(rows, cols, 'uint8');

for i = 2:rows+1

for j = 2:cols+1

region\_lowpass = padded\_lowpass(i-1:i+1, j-1:j+1);

region\_highpass = padded\_highpass(i-1:i+1, j-1:j+1);

filtered\_lowpass(i-1, j-1) = median(region\_lowpass(:));

filtered\_highpass(i-1, j-1) = median(region\_highpass(:));

end

end

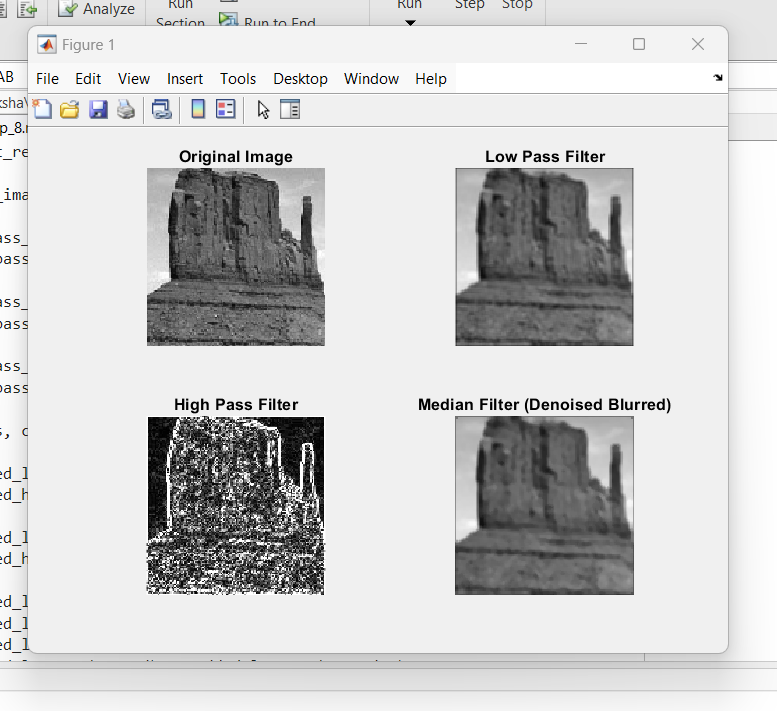
figure;

subplot(2,2,1), imshow(gray\_image), title('Original Image');

subplot(2,2,2), imshow(lowpass\_result), title('Low Pass Filter');

subplot(2,2,3), imshow(highpass\_result), title('High Pass Filter');

subplot(2,2,4), imshow(filtered\_lowpass), title('Median Filter (Denoised Blurred)');



**Conclusion:-**

This MATLAB implementation applies low-pass, high-pass, and median filtering to enhance and denoise a grayscale image without requiring additional toolboxes

**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 degrade signal quality. Common types include:

1. **Gaussian Noise (White Noise)**
   * Follows a normal distribution and affects all pixel values randomly.
   * Example: Sensor noise in low-light images.
2. **Salt and Pepper Noise (Impulse Noise)**
   * Appears as random white and black pixels due to transmission errors or sensor faults.
   * Example: Image corruption due to bit errors in data transmission.
3. **Poisson Noise (Shot Noise)**
   * Caused by variations in photon detection in optical sensors.
   * Example: Low-light photography results in grainy images.
4. **Speckle Noise**
   * Occurs due to interference in coherent imaging systems like radar and ultrasound.
   * Example: Ultrasound medical images often appear grainy due to speckle noise.
5. **Quantization Noise**
   * Arises from rounding errors during digitization of an analog signal.
   * Example: Low-bit depth images lose smooth gradients, appearing blocky.
6. **Explain with the help of an example how filtering helps in enhancing the quality of an image.**

Filtering improves image quality by removing noise and enhancing important features.

**Example: Removing Salt & Pepper Noise using a Median Filter**

* **Original Image:** Suppose an image is corrupted with salt and pepper noise, where random black and white pixels appear.
* **Filtering:** A **3×3 median filter** replaces each pixel with the median of its surrounding pixels, effectively removing outlier noise while preserving edges.
* **Enhanced Image:** The filtered image appears cleaner with noise removed, making details more visible.