**Batch: C - 3 Roll No.: 16010122096**

**Experiment No. 07**

**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:**

**ALGORITHM:**

**Main Program:**

1. **Start**
2. Prompt the user to enter the image name with extension.
3. Read the image.
4. If the image is colored (RGB), convert it to grayscale.
5. Convert the image to double precision.
6. Apply **low pass filter** to the image and store the result.
7. Apply **high pass filter** to the image and store the result.
8. Apply **median filter** to the image and store the result.
9. Display the original and all processed images in a 2x2 grid.
10. **End**

**Low Pass Filter:**

1. **Input:** Grayscale image
2. Define a 3x3 kernel with all elements equal to 1/91/91/9.
3. Call apply\_filter() with the image and kernel.
4. **Output:** Low pass filtered image

**High Pass Filter:**

1. **Input:** Grayscale image
2. Define a 3x3 kernel with values:

-1 -1 -1

-1 8 -1

-1 -1 -1

1. Call apply\_filter() with the image and kernel.
2. **Output:** High pass filtered image

**Median Filter:**

1. **Input:** Grayscale image, kernel size (default 3)
2. Define padding size as floor(ksize / 2).
3. Call zero\_pad() to add padding to the image.
4. For each pixel in the original image:
   * Extract the region of the image corresponding to the kernel size.
   * Compute the median of the region.
   * Assign the median value to the corresponding pixel in the output.
5. **Output:** Median filtered image

**apply\_filter():**

1. **Input:** Grayscale image, kernel
2. Get image dimensions h and w.
3. Get kernel dimensions kh and kw.
4. Define padding size as floor(kh / 2) and floor(kw / 2).
5. Call zero\_pad() to add padding to the image.
6. Create an empty matrix to store the result.
7. For each pixel in the original image:
   * Extract the region of the image corresponding to the kernel size.
   * Apply the kernel to the region and compute the sum.
   * Assign the result to the corresponding pixel in the output.
8. Clip pixel values between 0 and 255.
9. **Output:** Filtered image

**zero\_pad():**

1. **Input:** Grayscale image, pad\_h and pad\_w
2. Create a zero matrix of size (h + 2 \* pad\_h, w + 2 \* pad\_w).
3. Copy the original image to the center of the padded matrix.
4. **Output:** Padded image

**CODE:**

clc;

clear;

close all;

image\_path = input('Enter the image name with extension: ', 's');

img = imread(image\_path);

if size(img, 3) == 3

img = rgb2gray(img);

end

img = double(img);

low\_pass\_img = low\_pass\_filter(img);

high\_pass\_img = high\_pass\_filter(img);

median\_filtered\_img = median\_filter(img, 3);

figure;

subplot(2, 2, 1), imshow(uint8(img)), title('Original Image');

subplot(2, 2, 2), imshow(uint8(low\_pass\_img)), title('Low Pass Filtered');

subplot(2, 2, 3), imshow(uint8(high\_pass\_img)), title('High Pass Filtered');

subplot(2, 2, 4), imshow(uint8(median\_filtered\_img)), title('Median Filtered');

function result = apply\_filter(img, kernel)

[h, w] = size(img);

[kh, kw] = size(kernel);

pad\_h = floor(kh / 2);

pad\_w = floor(kw / 2);

padded\_img = zero\_pad(img, pad\_h, pad\_w);

result = zeros(h, w);

for i = 1:h

for j = 1:w

region = padded\_img(i:i+kh-1, j:j+kw-1);

result(i, j) = sum(sum(region .\* kernel));

end

end

result = max(0, min(result, 255));

end

function padded\_img = zero\_pad(img, pad\_h, pad\_w)

[h, w] = size(img);

padded\_img = zeros(h + 2 \* pad\_h, w + 2 \* pad\_w);

padded\_img(pad\_h+1:end-pad\_h, pad\_w+1:end-pad\_w) = img;

end

function result = low\_pass\_filter(img)

kernel = ones(3, 3) / 9;

result = apply\_filter(img, kernel);

end

function result = high\_pass\_filter(img)

kernel = [-1 -1 -1; -1 8 -1; -1 -1 -1];

result = apply\_filter(img, kernel);

end

function result = median\_filter(img, ksize)

[h, w] = size(img);

pad = floor(ksize / 2);

padded\_img = zero\_pad(img, pad, pad);

result = zeros(h, w);

for i = 1:h

for j = 1:w

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

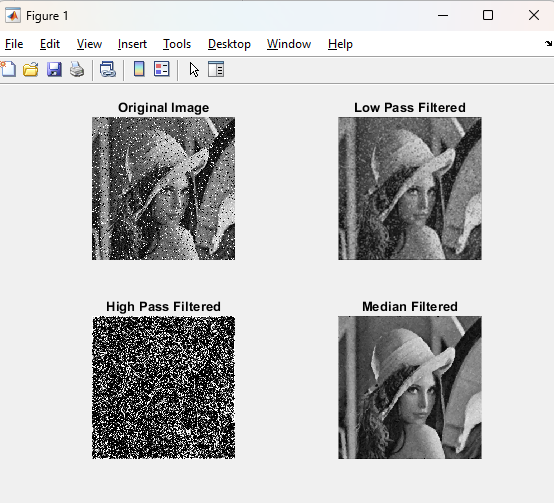
result(i, j) = median(region(:));

end

end

end

**OUTPUT:**



**Conclusion:-**

The implemented filters successfully enhance images by smoothing, sharpening, and noise reduction, demonstrating effective neighborhood processing in the spatial domain.

**Post Lab Descriptive Questions**

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

#### a) ****Gaussian Noise (Normal Noise)****

* **Description:** Statistical noise with a probability density function (PDF) following a Gaussian distribution.
* **Cause:** Thermal noise in electronic circuits.
* **Effect:** Affects all image pixels, introducing random brightness or color variations.

#### b) ****Salt and Pepper Noise (Impulse Noise)****

* **Description:** Random occurrences of black and white pixels (impulsive disturbances).
* **Cause:** Transmission errors, malfunctioning camera sensors.
* **Effect:** Pixels appear randomly white (salt) or black (pepper), degrading image quality.

#### c) ****Poisson Noise (Shot Noise)****

* **Description:** Noise arising due to statistical variations in photon detection.
* **Cause:** Low-light conditions during image acquisition.
* **Effect:** Variation in pixel intensity values, prominent in low-light or high-gain images.

#### d) ****Speckle Noise****

* **Description:** Multiplicative noise affecting radar and ultrasound images.
* **Cause:** Interference between reflected signals.
* **Effect:** Grainy texture in images, reducing clarity.

#### e) ****Periodic Noise****

* **Description:** Periodic pattern of noise due to electrical or mechanical interference.
* **Cause:** Scanner or camera hardware issues.
* **Effect:** Repetitive patterns or stripes degrading visual quality.

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

#### ****Example:****

* **Original Image:** Consider an image captured in a low-light environment with salt and pepper noise.
* **Problem:** The image contains random black and white dots affecting the overall clarity.

#### ****Solution Using Filtering:****

* **Apply Median Filter:**
  + Replaces each pixel value with the median of its neighborhood.
  + Removes salt and pepper noise effectively by preserving edge information.

#### ****Result:****

* **Enhanced Image:** The processed image appears smoother, with noise significantly reduced, improving the overall quality.