

```
% Image filtering using Wiener2 Filter (using MATLAB inbuilt function)

clear

close all

clc

% Read the input image

I = imread('Cameraman.tif');

% Convert to grayscale if not already

%I = rgb2gray(I); % (Optional, in case it's already grayscale, no effect)

% Add Gaussian noise to the image

I_noisy = imnoise(I, 'gaussian',0, 0.01); % mean=0, variance=0.01

% Apply Wiener filter

I_filtered = wiener2(I_noisy, [5 5]); % [m n] neighborhood size

% Display results

figure;

subplot(1,3,1);

imshow(I);

title('Original Image');

subplot(1,3,2);

imshow(I_noisy);

title('Noisy Image (Gaussian)');

subplot(1,3,3);

imshow(I_filtered);

title('Filtered Image (Wiener2)');
```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Adaptive Mean Filtering

% Read the original image

I = imread('cameraman.tif');

I = im2double(I); % Convert to double for calculations

% Add Gaussian noise

I\_noisy = imnoise(I, 'gaussian', 0, 0.01); % mean=0, variance=0.01

% Adaptive Mean Filter implementation

[m, n] = size(I\_noisy);

I\_filtered = zeros(m, n);

% Define window size

win\_size = 5; % 5x5 neighborhood

pad\_size = floor(win\_size/2);

I\_padded = padarray(I\_noisy, [pad\_size pad\_size], 'symmetric');

% Noise variance (estimated from the noisy image)

noise\_var = 0.01;

for i = 1:m

for j = 1:n

% Extract local window

local\_window = I\_padded(i:i+win\_size-1, j:j+win\_size-1);

% Local mean and variance

local\_mean = mean(local\_window(:));

local\_var = var(local\_window(:));

```

% Adaptive mean filter formula

if local_var > 0
    I_filtered(i,j) = I_noisy(i,j) - (noise_var/local_var) * (I_noisy(i,j) - local_mean);
else
    I_filtered(i,j) = local_mean;
end
end
end

% Display results

figure;
subplot(1,3,1); imshow(I); title('Original Image');
subplot(1,3,2); imshow(I_noisy); title('Noisy Image');
subplot(1,3,3); imshow(I_filtered); title('Filtered Image (Adaptive Mean)');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Adaptive Median Filter demo for 'cameraman.tif'

clear; close all; clc;

% Read image and convert to double
I = im2double(imread('cameraman.tif'));

% Add salt & pepper noise
noiseDensity = 0.20;      % 20% noise
I_noisy = imnoise(I, 'salt & pepper', noiseDensity);

```

```

% Apply adaptive median filter

S_max = 7; % maximum allowed window size

I_filtered = adaptiveMedianFilter(I_noisy, S_max);


% Show results

figure;

subplot(1,3,1); imshow(I); title('Original');

subplot(1,3,2); imshow(I_noisy); title('Noisy (Salt & Pepper)');

subplot(1,3,3); imshow(I_filtered); title('Filtered (Adaptive Median)');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Copy this following code in a separate file and save as a function

function J = adaptiveMedianFilter(I, S_max)

    [m, n] = size(I);

    J = zeros(m,n);

    pad = floor(S_max/2);

    P = padarray(I, [pad pad], 'symmetric');

    for r = 1:m

        for c = 1:n

            ip = r + pad; jp = c + pad;

            z_xy = P(ip, jp);

            windowSize = 3;

            done = false;

            while ~done

                h = floor(windowSize/2);

                local = P(ip-h:ip+h, jp-h:jp+h);

                z_min = min(local(:));

                z_max = max(local(:));

```

```

z_med = median(local(:));

% Stage A
A1 = z_med - z_min;
A2 = z_med - z_max;
if (A1 > 0) && (A2 < 0)
    % Stage B
    B1 = z_xy - z_min;
    B2 = z_xy - z_max;
    if (B1 > 0) && (B2 < 0)
        J(r,c) = z_xy;
    else
        J(r,c) = z_med;
    end
    done = true;
else
    windowSize = windowSize + 2;
    if windowSize <= S_max
        % repeat Stage A
    else
        J(r,c) = z_med;
        done = true;
    end
end
end
end
end
end

```