INDIAN INSTITUTE OF TECHNOLOGY

DELHI

Course: BML735

Biomedical Signal & Image Processing

Assignment-2 | 30 January 2022

Image Noise Filtering

Submitted by

Aniruddh Nagar - 2021EET2113

Table of Contents

1. PROBLEM STATEMENT:

Q1) Starting from the Gaussian noise (mean 0, $\sigma = 13$) corrupted image, compute both mean filter and Gaussian filter smoothing at various scales and compare each in terms of noise removal vs loss of detail. How would you select the correct σ for an image?

Q2) Add speckle noise to image and comment on which filter works best to remove this noise.

Q3) Demonstrate the effect of removing high-frequency values/coefficients on the quality of an image (Brain image).

Q4) Add motion blur and gaussian noise to image and restore the image by using Weiner filer. Comment on the image quality after image restoration.

2. READING & DISPLAYING OF 'T2W_BRAIN.DCM' IMAGE:

Code 1:

```
%% ========================================================================%%
%                          Course  : BML735 / Assignment2                   %
%                          Name    : Aniruddh Nagar                          %
%                          EntryNo : 2021EET2113                             %
% ========================================================================%

%% ====================== Workspace Initialisation ========================
close all;  % Close all figures (except those of imtool).
clear;      % Erase all existing variables.
clc;        % Clear the command window.

%% ============= Code for loading & displaying image  ======================
% Reading T2W_Brain.dcm image through given path ---
path = 'C:\Users\hp\MATLAB Drive\bml_practical\Assign_2\T2W_Brain.dcm';
img = dicomread(path);
img = double(img);

% Displaying Original T2W_Brain.dcm image ---
figure, imagesc(img), colormap(gray), colorbar, axis off, axis image;
                                                    title('Orignal Image');

% Normalising T2W_Brain.dcm image ---
norm_img = mat2gray(img);

% Displaying Normalised T2W_Brain.dcm image ---
figure, imagesc(norm_img), colormap(gray), colorbar, axis off, axis image;
                                                  title('Normalized Image');
```

Description:

***Path*:** is the variables, which stores the location of the input images.
dicomread(path): is the function to read the T2W_Brain.dcm image.
Image Normalization is done to scale the image in range [0-255]. Both, the original and Normalised T2W_Brain.dcm images have been displayed below in Gray colormap.

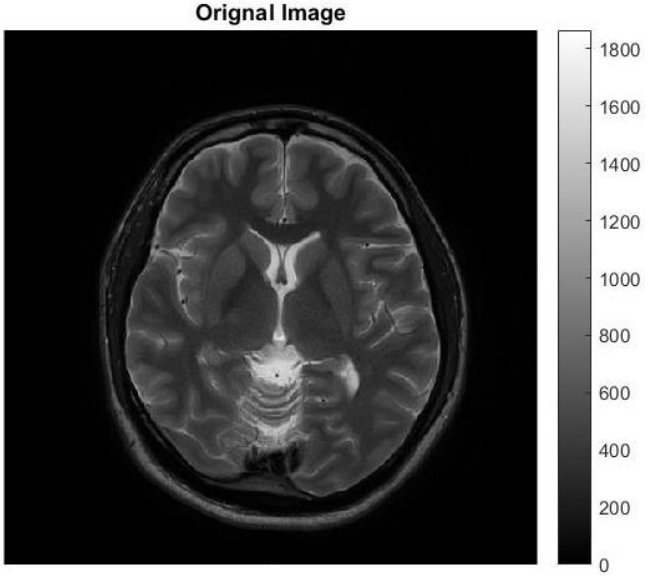


Figure 1: T2W_Brain original image

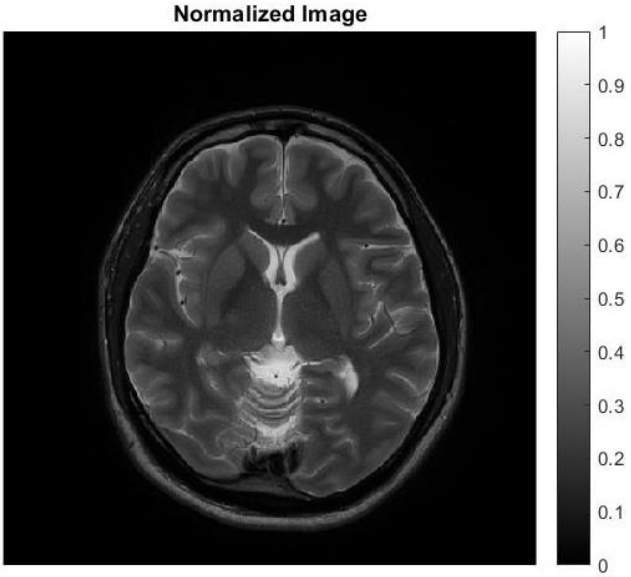


Figure 2: T2W_Brain Normalized image

3. CORRUPTING 'T2W_BRAIN.DCM' IMAGE USING GAUSSIAN NOISE:

Code 2:

```
29  %% ============  1a. Code for adding Gaussian noise to image  ==============
30  % Corrupting image using Gaussian noise (mean 0, σ = 0.13) ---
31  mean = 0;
32  variance = 0.13;
33  noisy_img = imnoise(norm_img,'gaussian',mean,variance);
34
35  % Displaying Gaussian-Noisy T2W_Brain.dcm image ---
36  figure, imagesc(noisy_img), colormap(gray), colorbar, axis off, axis image;
37                              title('Gaussian Noise with mean:0 & var: 0.13');
```

Description:

Noise result in pixel values that do not reflect the true intensities of the real scene. Here, inbuilt function of MATLAB is used to add Gaussian noise to normalised image.

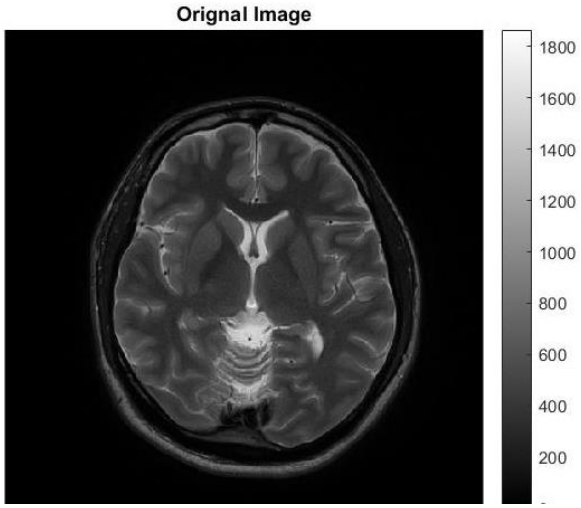


Figure 3: T2W_Brain original image

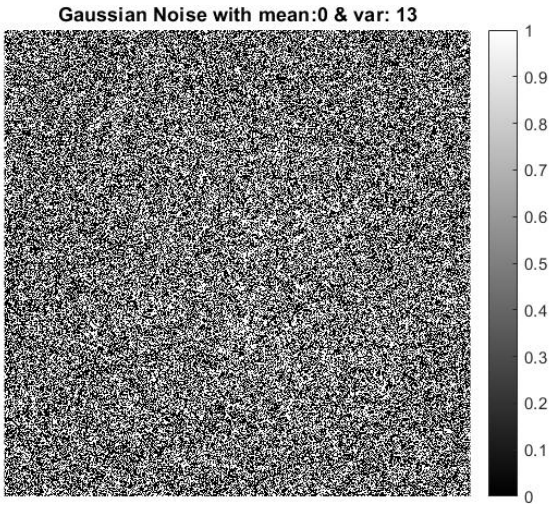


Figure 4: Gaussian Noise image, SD: 13

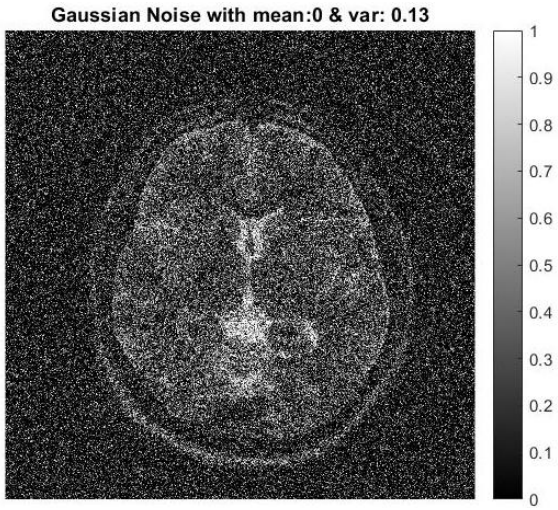


Figure 5: Gaussian Noise image, SD: 0.13

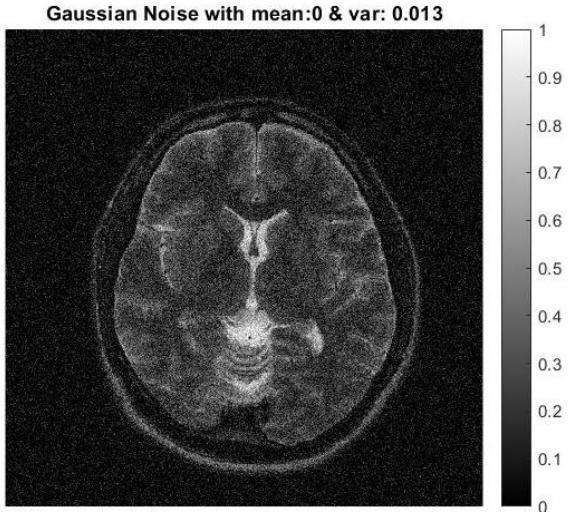


Figure 6: Gaussian Noise image, SD: 0.013

4. COMPARISON OF MEAN & GAUSSIAN FILTER :

Mean Filter: is a non-linear filter, calculates the mean value around a target pixel and replaces the target pixel with the filtered result. It gives the same weighting to all the neighbouring pixels; thus, a pixel can be largely affected by surrounding pixels.

Gaussian filter: is a linear filter, designed to reduce the effects of surrounding pixels. The Gaussian filter weigh pixels as bell-curve around the centre pixel and decreases the weight as a pixel moves away from the target pixel.

Code 3:

```
%% ============  1b. Smoothing Noisy image using Filters  =================
f = figure();
f.WindowState = 'maximized';

% Applying Mean filter using fspecial ---
mean_filt1 = fspecial('average', 9);
m_fil_img = imfilter(noisy_img, mean_filt1);
subplot(2,3,1), imagesc(m_fil_img), colormap(gray), colorbar, axis off, axis image;
                                                title('Mean filtered: 9x9');

mean_filt2 = fspecial('average', 5);
m_fil_img = imfilter(noisy_img, mean_filt2);
subplot(2,3,2), imagesc(m_fil_img), colormap(gray), colorbar, axis off, axis image;
                                                title('Mean filtered: 5x5');

mean_filt3 = fspecial('average', 3);
m_fil_img = imfilter(noisy_img, mean_filt3);
subplot(2,3,3), imagesc(m_fil_img), colormap(gray), colorbar, axis off, axis image;
                                                title('Mean filtered: 3x3');
```

Code 4:

```

% Applying Gaussian filter using fspecial ---
gaus_filt1 = fspecial('gaussian', 9, 5);
g_fil_img = imfilter(noisy_img, gaus_filt1);
subplot(2,3,4), imagesc(g_fil_img), colormap(gray), colorbar, axis off, axis image;
                                                title('Gaussian filtered: 9x9');
gaus_filt2 = fspecial('gaussian', 5, 5);
g_fil_img = imfilter(noisy_img, gaus_filt2);
% Displaying Gaussian-Filtered T2W_Brain.dcm image ---
subplot(2,3,5), imagesc(g_fil_img), colormap(gray), colorbar, axis off, axis image;
                                                title('Gaussian filtered: 5x5');
gaus_filt3 = fspecial('gaussian', 3, 5);
g_fil_img = imfilter(noisy_img, gaus_filt3);
% Displaying Gaussian-Filtered T2W_Brain.dcm image ---
subplot(2,3,6), imagesc(g_fil_img), colormap(gray), colorbar, axis off, axis image;
                                                title('Gaussian filtered: 3x3');
```

- ***Comparison of Mean and Gaussian filter of sizes 9x9, 5x5 & 3x3:-***

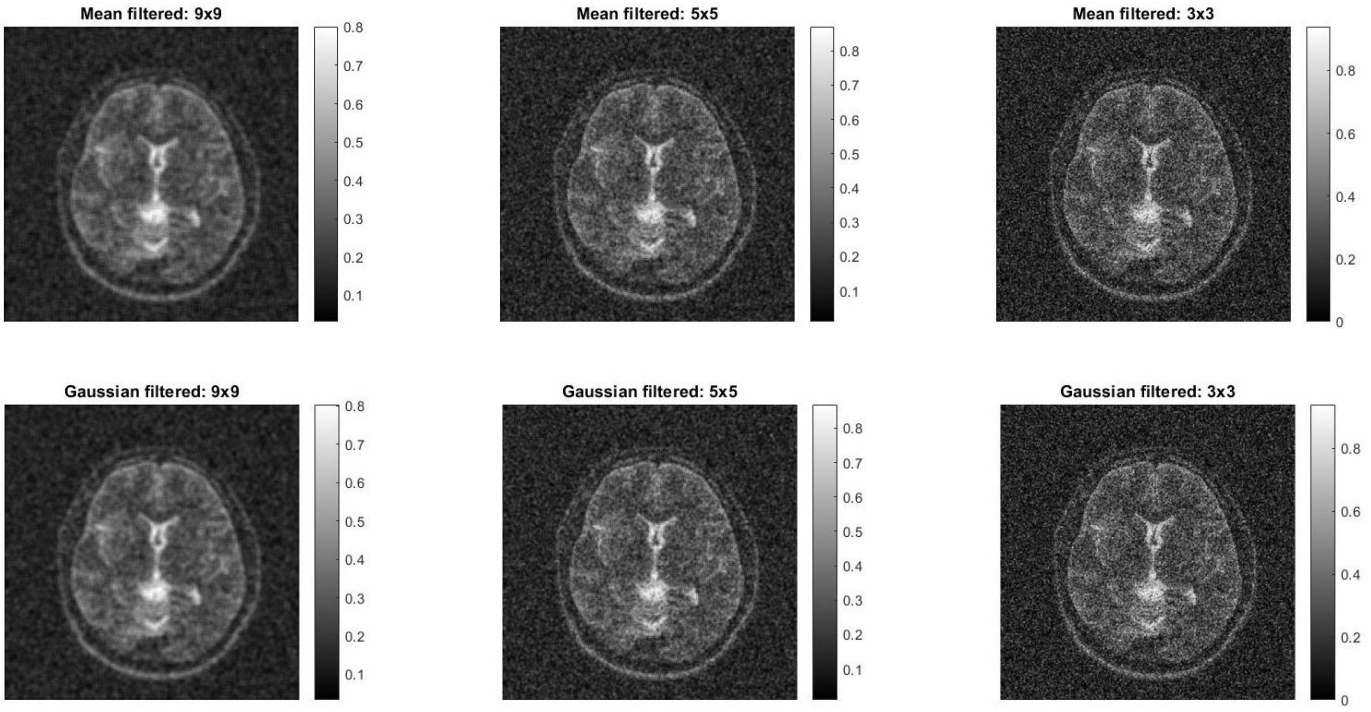


Figure7: Filtered images for noisy image of SD: 0.13

- ***Comparison of Mean and Gaussian filter of sizes 9x9, 5x5 & 3x3:-***

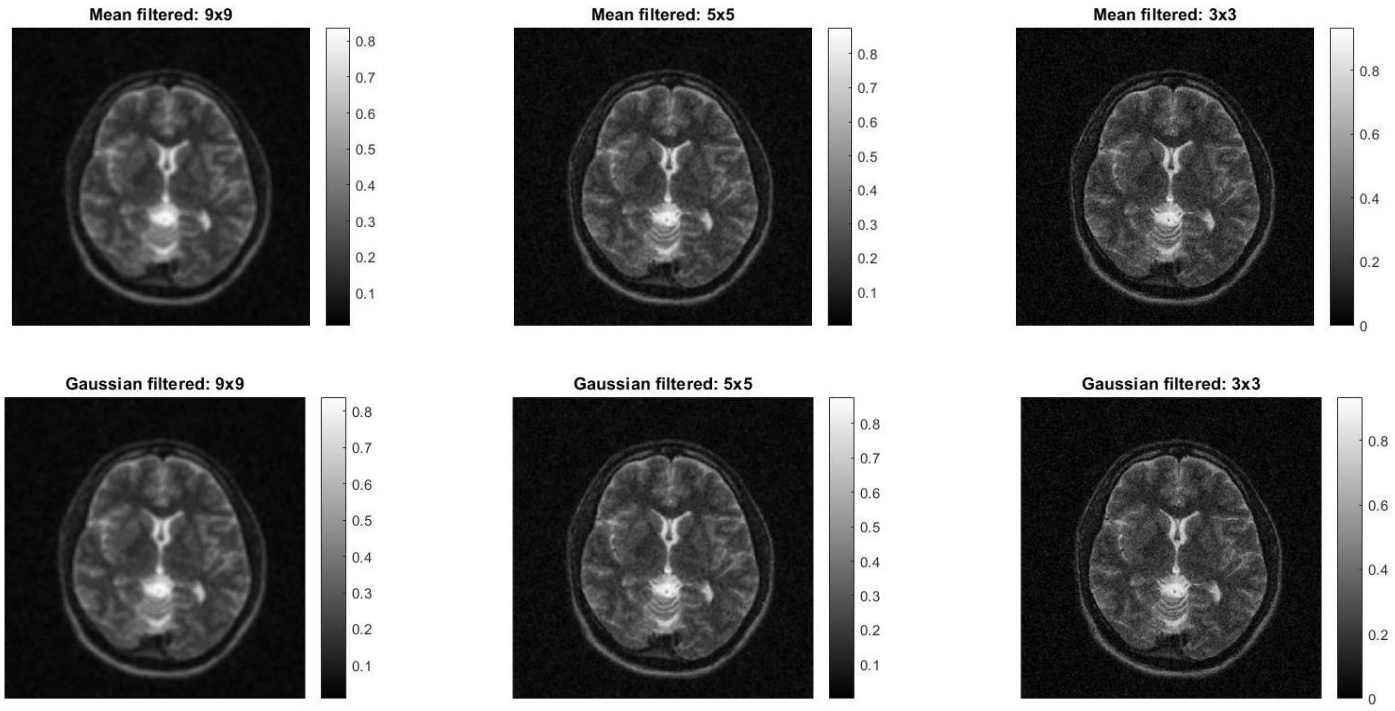


Figure8: Filtered images for noisy image of SD: 0.013

In terms of Noise:

Mean filter is the **worst filter for frequency domain**, with little ability to separate band of frequencies, but the noise of the entire image can be processed **quickly**.

Gaussian filter is much **better at separating frequencies**, produce clearer, smoother, more visually pleasing image, maintain original clarity to some degree. But it is **slow**.

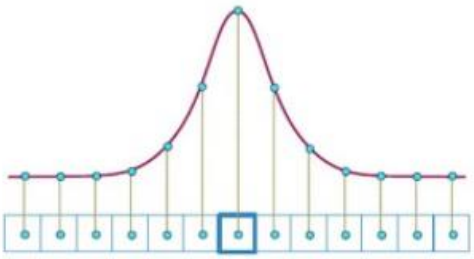
In terms of loss of detail:

Gaussian filter loses details as it reduces the effects of surrounding pixels by decreasing their weight.

Selection of correct Sigma (σ) for an image: -

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-x^2}{2\sigma^2}}$$

σ : controls the amount of smoothing



Where, σ is the standard deviation of the distribution. Assuming mean of 0.

The values located between +/- σ account for 68 % of the set, while +/- 2σ from the mean account for 95 % and +/- 3σ account for 99.7 %. This is very important when designing a Gaussian kernel of fixed length.

Therefore, choice of sigma is made in such a way that negligible Gaussian weights lies outside it, but also, we have to keep check of much smoothing, which can destroy image to irrecoverable state.

In my case, I have tried gaussian noise with σ = 13, 0.13 and 0.013.

I found better results on using gaussian filter with size 3x3.

5. ADDING OF SPECKLE NOISE TO IMAGE:

Speckle Noise:

The speckle noise results from the interference of many waves of the same frequency, having different phases and amplitudes, which add together to give a resultant wave whose amplitude and intensity varies randomly.

Code 5:

```
%% ============  2a. Code for adding speckle noise to image  ================
% Corrupting image using Speckle noise ---
speky_img = imnoise(norm_img, "speckle", 0.6);

% Displaying Original Normalised T2W_Brain.dcm image ---
figure, imagesc(norm_img), colormap(gray), colorbar, axis off, axis image;
                                        title('Original Normalized Image');

% Displaying Speckle-Noisy T2W_Brain.dcm image ---
figure, imagesc(speky_img), colormap(gray), colorbar, axis off, axis image;
                                         title('Image with Speckle noise');

```

Description:

Above code applies Speckle noise to Normalised image using inbuilt MATLAB function, **imnoise().** It is the function which is used to add noise to image.

J = imnoise(I, type) adds noise of a given type to the intensity image I. type is a string that specifies the types of noise .

J = imnoise(I, 'speckle', s) adds speckle noise to the image I, using the equation **J = I + n * I ,** where n is uniformly distributed random noise with mean 0 and variance s. The default value of s is 0.04

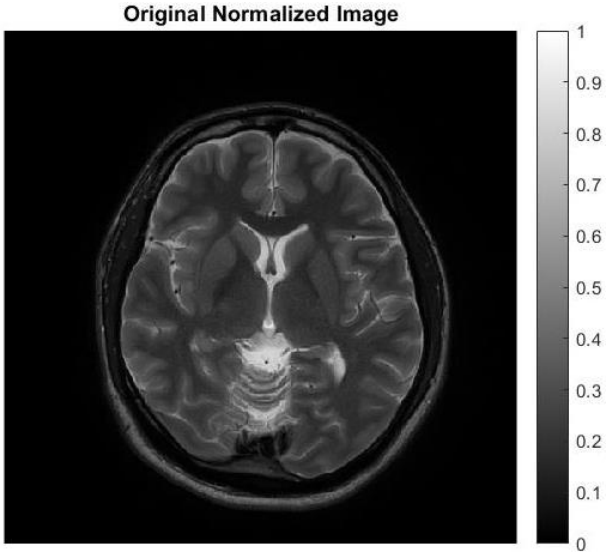


Figure 9: T2W_Brain Normalised image

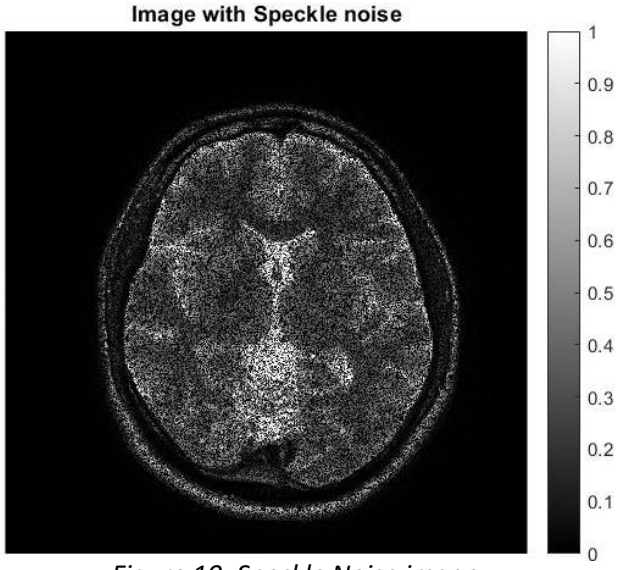


Figure 10: Speckle Noise image

6. FINDING BEST FILTER FOR SPECKLE NOISE:

Code 6:

```
%% =============  2b. Smoothing Noisy image using Filters  ================
% Applying Mean filter using fspecial ---
mean_filt = fspecial('average', 7);
mean_img = imfilter(speky_img, mean_filt);
figure, imagesc(mean_img), colormap(gray), colorbar, axis off, axis image;
                                              title('Mean filtered image');

% Applying Gaussian filter using fspecial ---
gaus_filt = fspecial('gaussian', 5, 5);
gaus_img = imfilter(speky_img, gaus_filt);
figure, imagesc(gaus_img), colormap(gray), colorbar, axis off, axis image;
                                           title('Gaussian filtered image');

% Applying Median filter ---
med_img = medfilt2(speky_img, [3 3]);
figure, imagesc(med_img), colormap(gray), colorbar, axis off, axis image;
                                            title('Median filtered image');

% Applying Wiener filter ---
wnr_img = wiener2(speky_img,[3 3]);
figure, imagesc(wnr_img), colormap(gray), colorbar, axis off, axis image;
                                            title('Wiener filtered image');

```

Description:

In the above code, I have applied various type of filters on the Speckle noise affected image. These filters include Mean, Gaussian, Median and Wiener filter.

Comparing: Mean, Gaussian, Median and Wiener filter: -

Mean Filter: is a non-linear filter. It removes noise while keeping edges relatively sharp. It can introduce patterns and maxima where previously there were none.

Gaussian filter: is a linear filter, can alone be able to blur edges and reduce contrast. The Gaussian filter produced a smoother, more visually pleasing result.

Median filter: is a non-linear filter, it preserves edges while removing the noise.

Wiener filter: adopts Noise-to-Signal-Power Ratio as a criterion for Image restoration. It produces satisfactory results in most cases. However, when the NSR is very small, the reconstructed result is unsatisfactory.

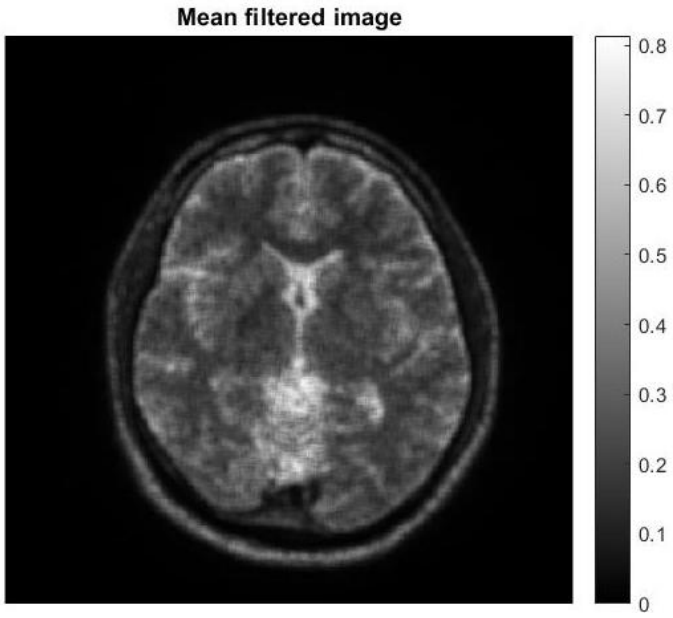


Figure 11: Mean Filtered image

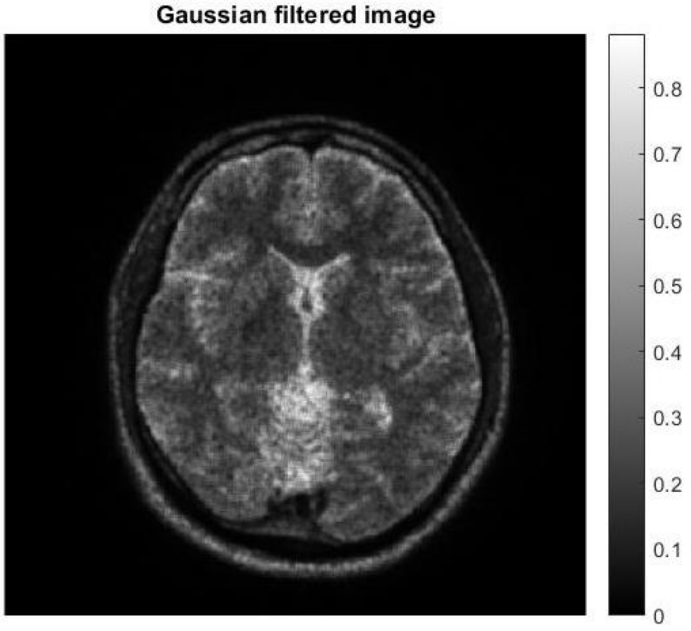


Figure 12: Gaussian Filtered image

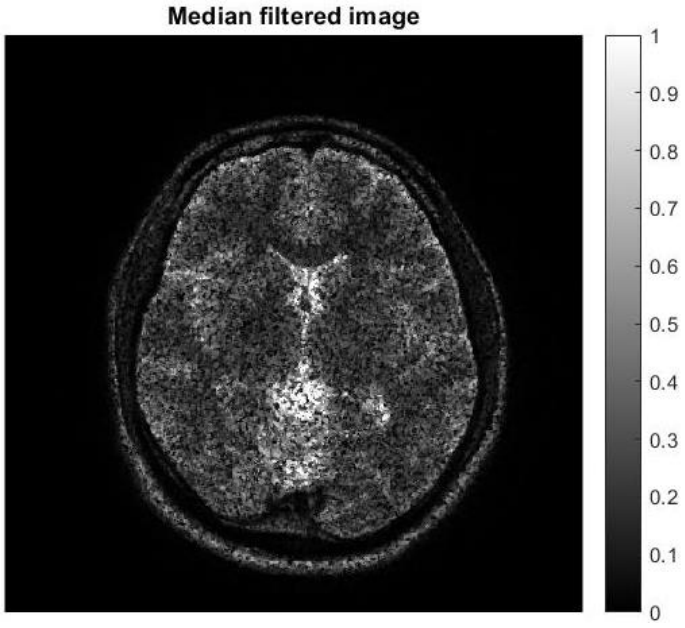


Figure 13: Median Filtered image

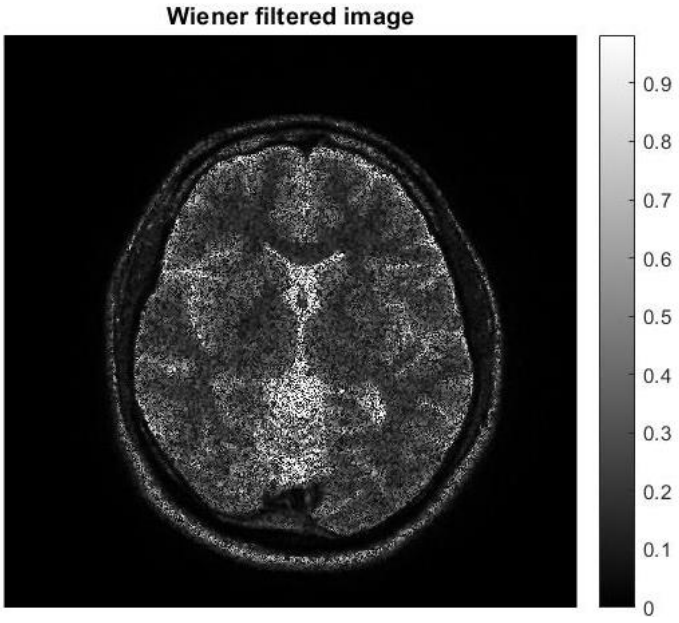


Figure 14: Wiener Filtered image

After de-noising for Speckle noise using various filters,

I have observed that the **performance of the Wiener Filter is much better** than Mean filter, Gaussian filter and Median filter.

7. EFFECT OF HIGH-FREQUENCY REMOVAL ON IMAGE QUALITY:

Code 7:

```
%% ========= 3. Effect of High Freq removal on image quality  ==============
f = figure();
f.WindowState = 'maximized';
i = 0; % iterator for subplot
% Displaying Original T2W_Brain.dcm image ---
subplot(2,3,i+1), imagesc(norm_img), colormap(gray), colorbar, axis off,
                                        axis image; title('Orignal Image');

for amp_Th = 11:-1:7
    i=i+1;
    % Compute the 2D fft.
    norm_fft = fftshift(fft2(norm_img));
    % Taking log magnitude for better display ---
    minVal = min(min(log(abs(norm_fft))));
    maxVal = max(max(log(abs(norm_fft))));

    % Find the location of the high frequencies ---
    high_freq = (log(abs(norm_fft))+1) > amp_Th; % Binary image.
```

Description:

Frequency Domain Filters basically focuses on the frequency of the images. They are used for smoothing and sharpening of image by removal of high or low frequency components. The Inbuilt Functions used are:

fft2( ) : inbuilt function used to apply forward Fourier transform on 2D signal.
ifft2( ) : inbuilt function used to apply inverse Fourier transform on 2D signal.
fftshift( ) : inbuilt function is used to shift corners to centre in FT image.
log( ) : inbuilt function is used to evaluate logarithm of FT complex signal.

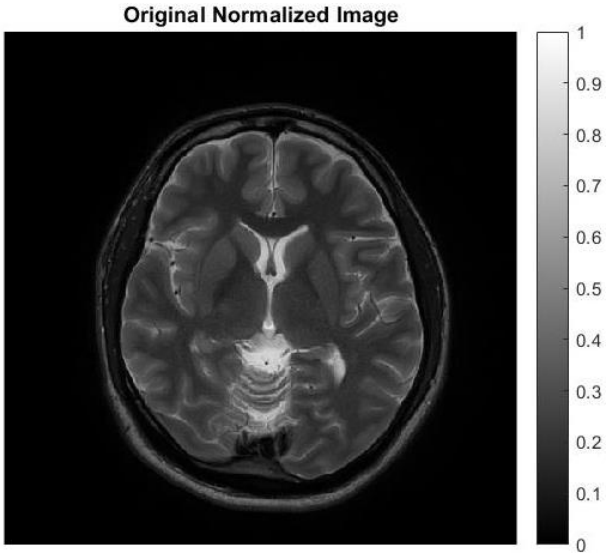


Figure 15: T2W_Brain Normalised image

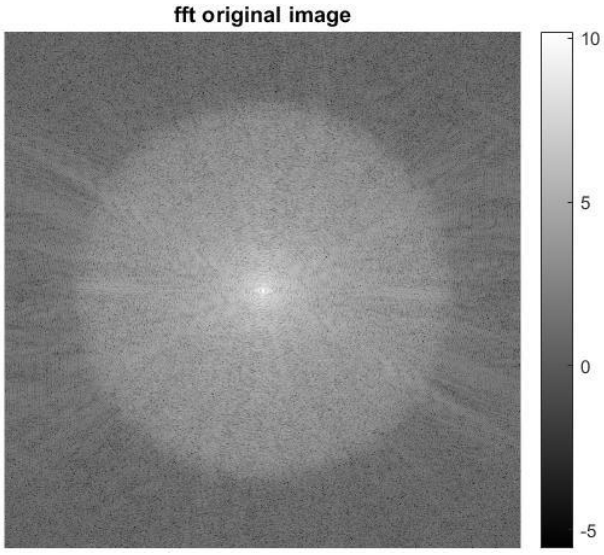


Figure 16: Fourier transformed image

Code 8:

```
110         % Removing the high freq spikes by filter/masking the spectrum ---
111         norm_fft(high_freq(:) == 1) = 0;
112         minVal = min(min(log(abs(norm_fft))));
113         maxVal = max(max(log(abs(norm_fft))));
114
115         % Final filtered image ---
116         filtered_img = ifft2(fftshift(norm_fft));
117         minVal = min(min(abs(filtered_img )));
118         maxVal = max(max(abs(filtered_img )));
119
120         subplot(2,3,i+1),imagesc(abs(filtered_img ), [minVal maxVal]),
121         colormap(gray), colorbar, axis off, axis image; title('Filtered image',i);
122     end
```

Description:

Lower frequency represents **the smooth part** of the image.
Higher frequency represents **the shape components** like edges of an image.

Lowpass Filter: is one which suppresses or attenuates the high frequency components of a spectrum while 'passing' the low frequencies within a specified range. In the above code, I have tried to apply lowpass filters with different thresholds, to remove high frequency components.

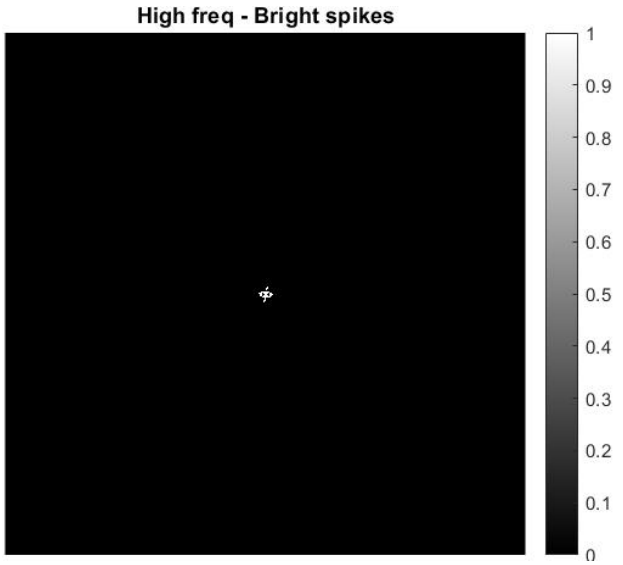


Figure 17: Bright spikes

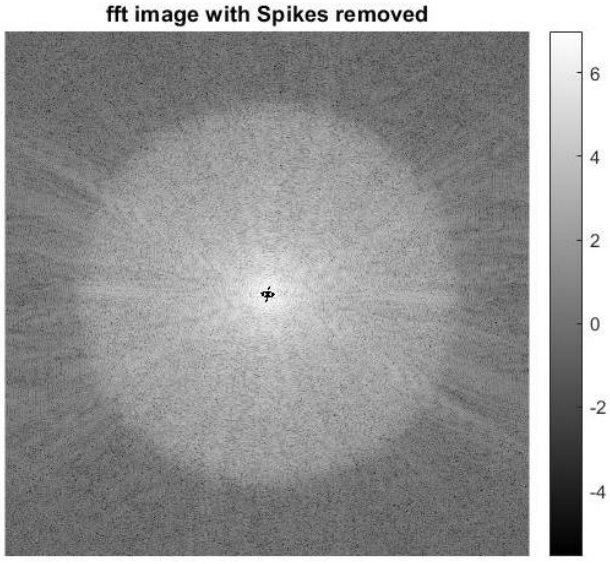


Figure 18: Image with spikes removed

If the high frequency value is removed from the frequency domain image, then the edges of spatial domain image will get degraded. If the low-frequency part is removed from the frequency domain image then the spatial domain image will get blurred.

The more the number of high frequency components are removed from the frequency domain image, the higher the edges of spatial domain image will get degraded. This can be observed in figure 19, which shows effect of removal of a greater number of high frequency components, shown from left to right.

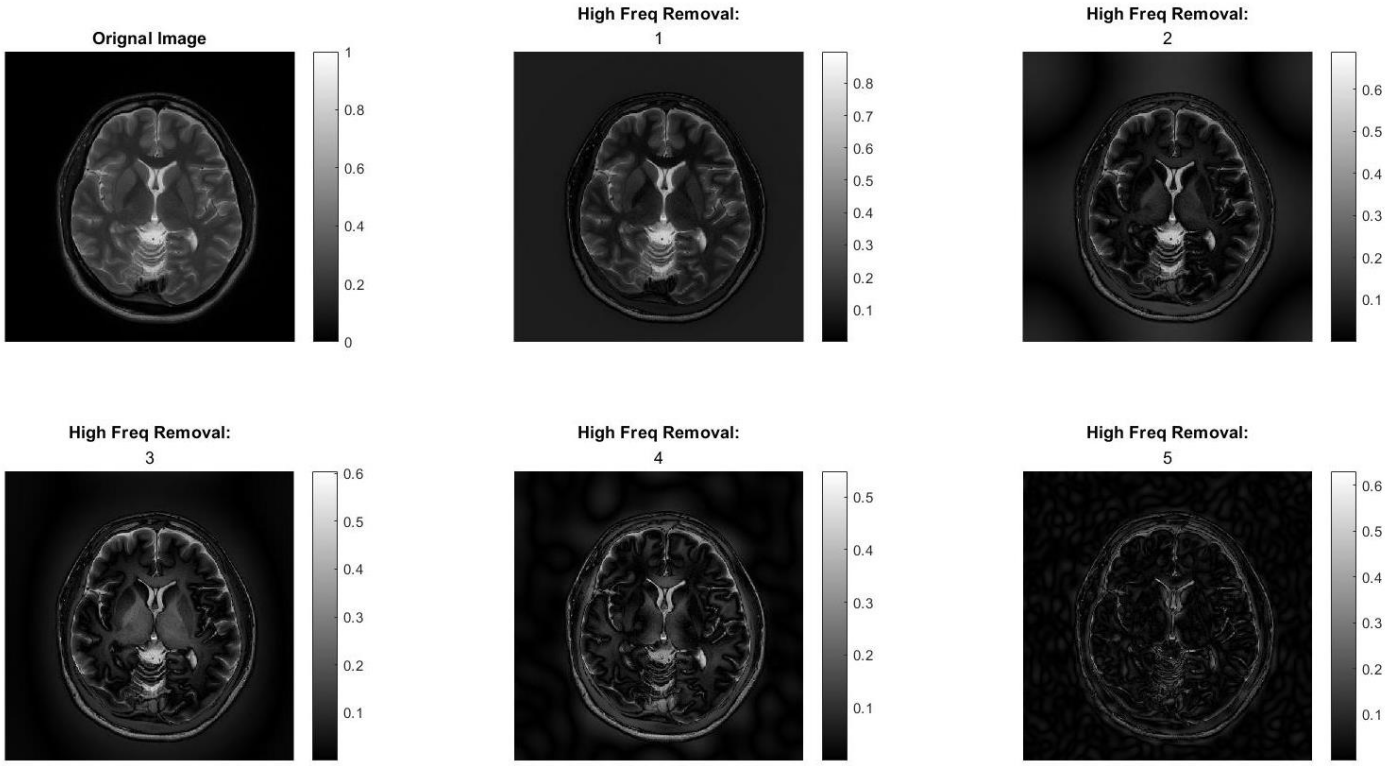


Figure 19: Effect of removal of a greater number of high frequency components
The more the number of high frequency components are removed from the frequency domain image, the higher the edges of spatial domain image will get degraded

8. ADDING BLUR & GAUSSIAN NOISE TO IMAGE:

Code 9:

```
132  %% ============  4a. Additive Effect of blur and gaussian noise  ============
133  % Adding blur and gaussian noise ---
134  PSF = fspecial('motion', 21, 11);
135  b_img = imfilter(norm_img, PSF,'conv','circular');
136  b_g_img = imnoise(b_img, 'gaussian', 0, 0.005);
137  figure, imshow(b_g_img);title('Blury and Gausy image')
```

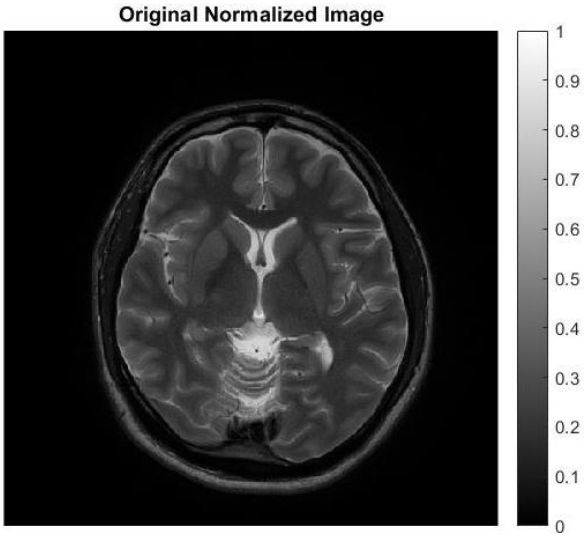


Figure 20: Normalised image

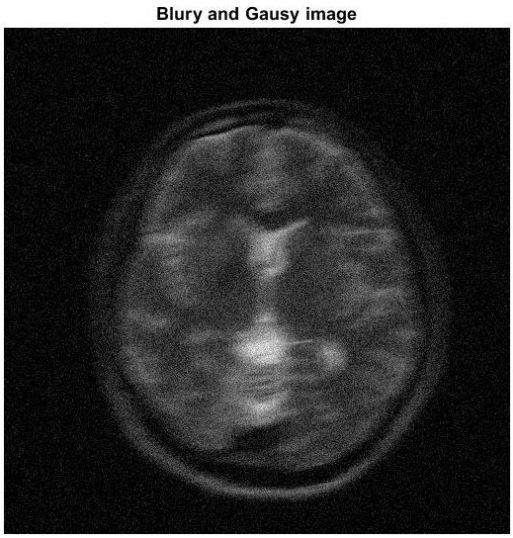


Figure 21: Blurry & Gaussian image

9. RESTORING BLUR & GAUSSIAN IMAGE USING WEINER FILTER:

Code 10:

```
139  %% ===============  4b. Restoration using Weiner Filter  ==================
140  % Restoration assuming no noise.
141  nsr = 0;
142  wnr_img1 = deconvwnr(b_g_img, PSF, nsr);
143  figure, imshow(wnr_img1); title('Restored Blury_Gausy Image with Zero NSR')
144
145  % Restoration using a better estimate of NSR ratio.
146  estim_nsr = 0.01 / var(norm_img(:));
147  wnr_img2 = deconvwnr(b_g_img, PSF, estim_nsr);
148  figure, imshow(wnr_img2); title('Restored Blury_Gausy Image with Estimated NSR');
```

Description:

Above code deals with the restoration of blurry and gaussian noise image using Weiner filter.

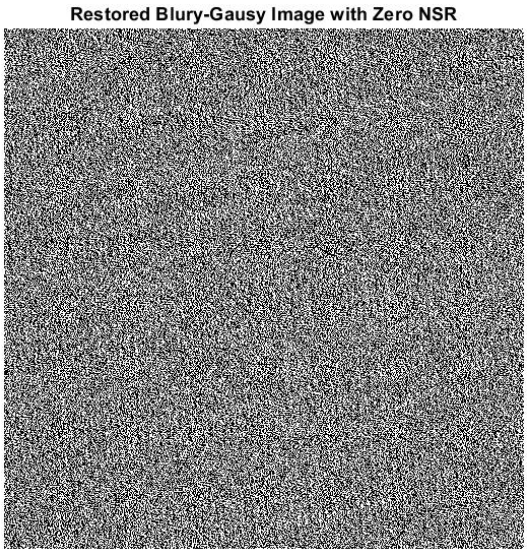


Figure 22: NSR=0, Restored image

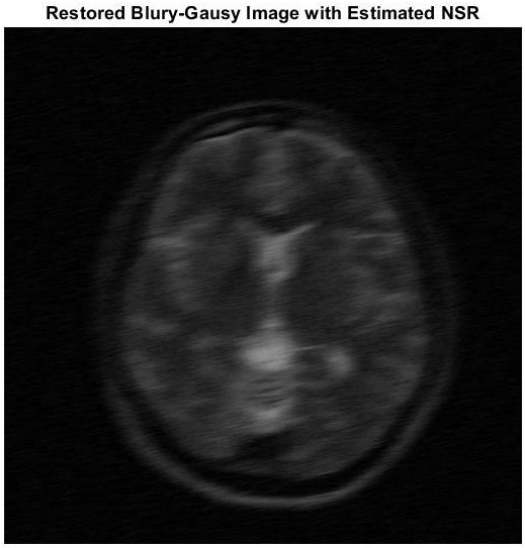


Figure 23: Estimated NSR, Restored image

Wiener filter adopts Noise-to-Signal-Power Ratio as a criterion for Image restoration. Wiener filter produces satisfactory results in most cases. However, when the NSR is very small, the reconstructed result is unsatisfactory. This can be observed from the above restored images with NSR = 0 and NSR = Estimated value.