Nikeet Pandit

214118806

Assignment 3

Harmonic Analysis and Image Processing

Introductory Remarks

In this assignment, periodic noise reduction, Wiener, constrained least-squares (CLS), and inverse filtering are to be tested. All functions are appended at the end of this report.

Periodic Noise Reduction Using Frequency Domain Filtering

In the first filter comparison, we will look at periodic noise reduction using frequency domain filtering. Specifically, we will look at notch filtering in the frequency domain. First, the image will be read in, and periodic noise is added to the image. Periodic noise is added to the image in the frequency domain for simplicity and the image is transformed back to the spatial domain with the periodic noise added. Thereafter, a notch reject filter is constructed to reject the added periodic noise. Codes for constructing the notch reject kernels have been written and are pasted to the appendix. Metrics to test the efficacy of the notch filter are the mean-square error (MSE) and structural similarity index (SSI) are employed to compare the original and filtered image. Other types of noise are also tested to see the resultant spectrum.

As will be seen, when the noise is not periodic, the image could likely be improved with a lowpass filter to smooth out high frequency variations. However, other spatial algorithms (for example adaptive spatial filtering likely may perform better) and preserve more edge details that the low-pass frequency domain filtering is not capable of due to its very nature of smoothing high frequency components. More importantly, the effect of different sorts of noise in the spectrum is evident in the visual change of the noise added image spectrum compared against the original spectrum; however, it is impossible to exactly detect where in the spectrum the noise is exactly contained because it is not exactly periodic at one frequency where it can easily be removed. When the noise is periodic, it can easily be removed with some investigation of the spectrum as will be seen.

Nikeet Pandit

214118806

Assignment 3

Harmonic Analysis and Image Processing

Reading Image

```
Im = imread("rose512.tif");
```

Adding Periodic Noise

```
%--- FFT of image and centering
Im_DFT = fftshift(fft2((double(Im))));

%--- Calculating distance matrix from centre of frequency domain rectangle
[M,N] = size(Im);
u = 0:M-1; %frequency components
v = 0:N-1;
[U, V] = meshgrid(u,v);
D = hypot(U-M/2, V-N/2); %calculating distances

%--- Extracting top left and bottom right index from distance matrix equal to 348
[row,col] = find(D == 348);
TopLeft = [row(1), col(1)]; %TopLeft and BotRight and on opposite corners of D matrix
BotRight = [row(end), col(end)];

%--- Adding noise at these locations
Noise = 2e7;
Im_DFT(TopLeft(1),TopLeft(2)) = Im_DFT(TopLeft(1),TopLeft(2)) + Noise;
Im_DFT(BotRight(1),BotRight(2)) = Im_DFT(BotRight(1),BotRight(2)) + Noise;

%--- Inverting by IFFT2
Im_Noisy = ifft2(fftshift(Im_DFT));

%--- Plotting original and noise added spectrum
fh = figure(1); fh.WindowState = 'maximized';
subplot_tight(1,2,1)
imagesc((abs(real(fftshift(fft2((double(Im))))))));
colorbar; title('Original Spectrum');
subplot_tight(1,2,2); imagesc((abs(real(Im_DFT)))); colorbar; title('Noise Added Spectrum');
```

Where the lines and circles are denotes where the periodic noise has been added in the frequency domain.

Nikeet Pandit

214118806

Assignment 3

Harmonic Analysis and Image Processing

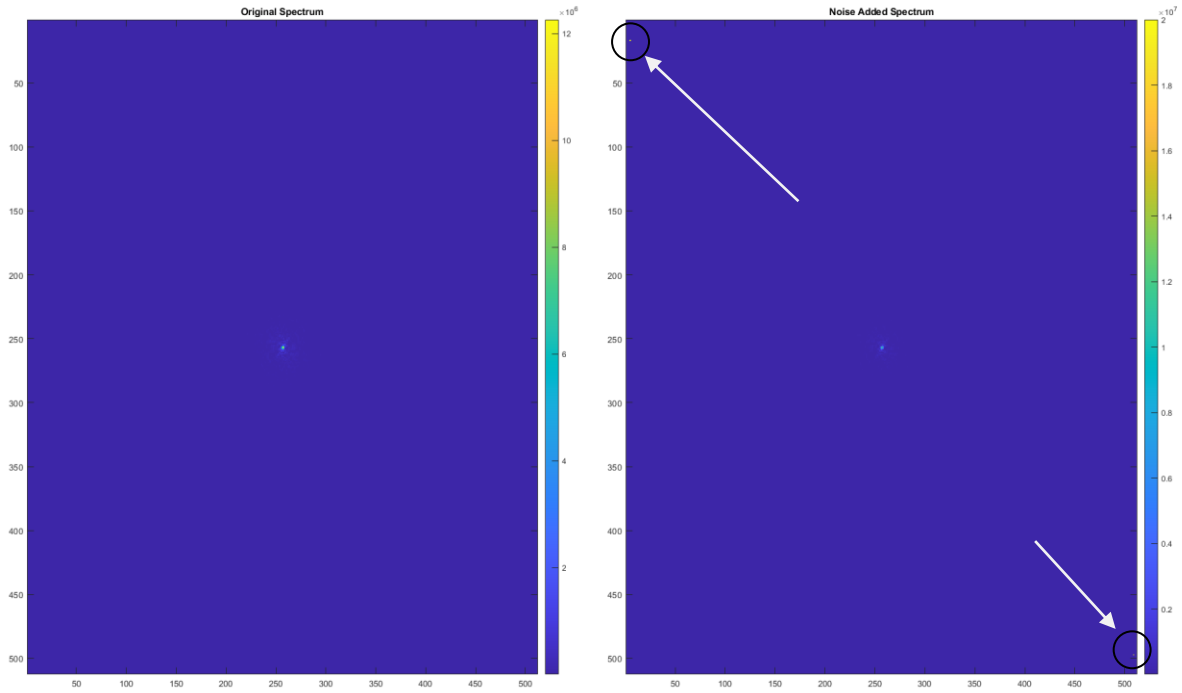

Figure 1

```
%--- Setting cut off frequency and location of notch filter offset
%offset is with respect to centre of frequency centre
D0 = 5;
Vk = 3; Uk = 16; %determined as function of where periodic noise and visually from spectrum
kernel = kernel_construct_notch(D0, Uk, Vk, M, N, 'gaussian');
fh = figure(2); fh.WindowState = 'maximized';
h = surf(fftshift(kernel)); title('Notch Filter Gaussian Kernel');

%--- Filtering Periodic Noisy Image with Notch Filter and recovering image %in spacetial domain
Im_Filtered = ifft2(fftshift((fftshift(fft2(Im_Noisy)).*kernel)));

%--- Plotting All Results
fh = figure(3); fh.WindowState = 'maximized';
subplot_tight(1,3,1); imshow(uint8(Im)); title('Original Image');
subplot_tight(1,3,2); imshow(uint8(Im_Noisy)); title('Periodic Noise Added');
subplot_tight(1,3,3); imshow(uint8(Im_Filtered)); title('Notch Filtered');
```

Nikeet Pandit

214118806

Assignment 3

Harmonic Analysis and Image Processing

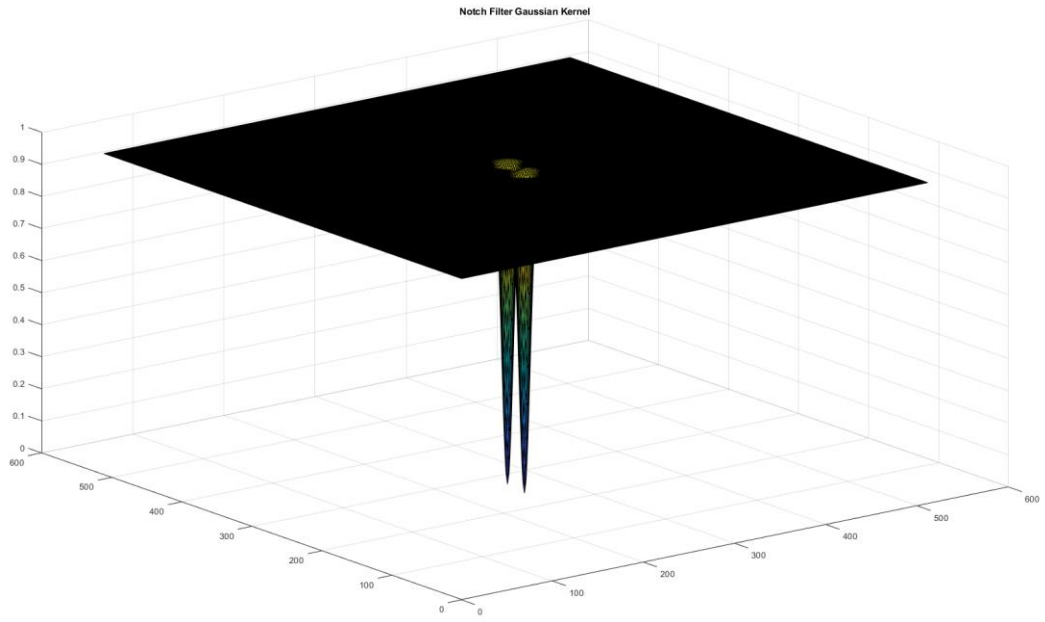

Figure 2

In the above photo, the kernel represents the centered notch kernel. Centering the noise added spectrum and multiplying this kernel with the noise added spectrum results in the attenuation of where the noise has been added. The location of where the centre frequency of the notches was determined through visual inspection of the spectrum and knowledge a-priori of where I added the noise.

Nikeet Pandit

214118806

Assignment 3

Harmonic Analysis and Image Processing

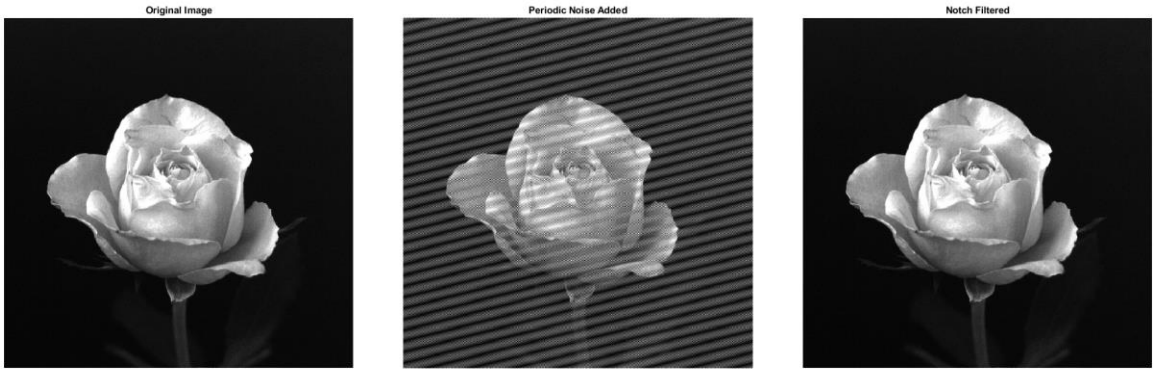

Figure 3

As we can see, the image on the left is the original image. The centre image has the periodic noise added to it in the image spatial representation. The image on the right-hand side has been filtered using the notch filter based on the filter kernel shown above which resembles visually very well the original image.

```
%--- Testing Notch Filter Periodic Noise Removal in Frequency Domain with Other noise sources
fh = figure(4); fh.WindowState = 'maximized';
Im_gaus = imnoise(Im,'gaussian',0,0.1); %mean 0 var 0.1
Im_sp = imnoise(Im,'salt & pepper',0.05); %salt and pepper noise density 0.05
Im_uform = double(Im)+10*rand(M,N); %uniform noise a = 0, b = 10

subplot_tight(2,2,1); imshow(uint8(Im)); title('Original Image');
subplot_tight(2,2,2); imshow(uint8(Im_gaus)); title('Gaussian Noise Added');
subplot_tight(2,2,3); imshow(uint8(Im_sp)); title('Salt & Pepper');
subplot_tight(2,2,4); imshow(uint8(Im_uform)); title('Uniform Noise');
```

Nikeet Pandit

214118806

Assignment 3

Harmonic Analysis and Image Processing

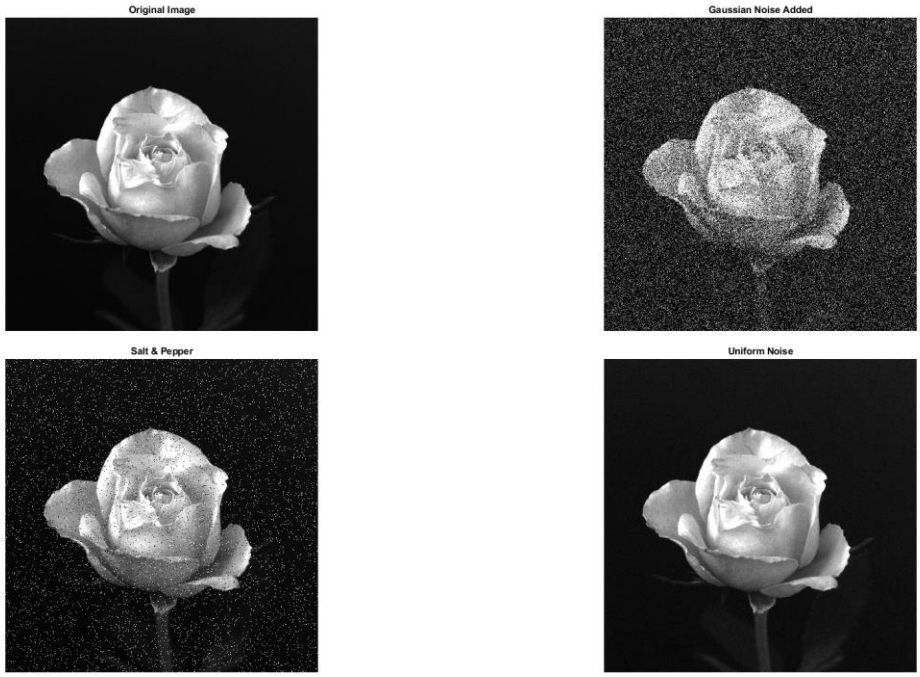

Figure 4

```
%--- Plotting Amp Spectrums
fh = figure(5); fh.WindowState = 'maximized';
%--- Calc. amplitude spec in dB
ImSpec = 20*log10(abs(real(fftshift(fft2(Im))))); %isolating amplitude component and taking abs
value
Im_gausSpec = 20*log10(abs(real(fftshift(fft2(Im_gaus)))));
Im_spSpec = 20*log10(abs(real(fftshift(fft2(Im_sp)))));
Im_uformSpec = 20*log10(abs(real(fftshift(fft2(Im_uform)))));

subplot_tight(2,2,1); imagesc(ImSpec); colorbar; title('Original Image Spec');
subplot_tight(2,2,2); imagesc(Im_gausSpec); colorbar; title('Gaussian Noise Added Spec');
subplot_tight(2,2,3); imagesc(Im_spSpec); colorbar; title('Salt & Pepper Spec');
subplot_tight(2,2,4); imagesc(Im_uformSpec);  colorbar; title('Uniform Noise Spec');
```

Nikeet Pandit

214118806

Assignment 3

Harmonic Analysis and Image Processing

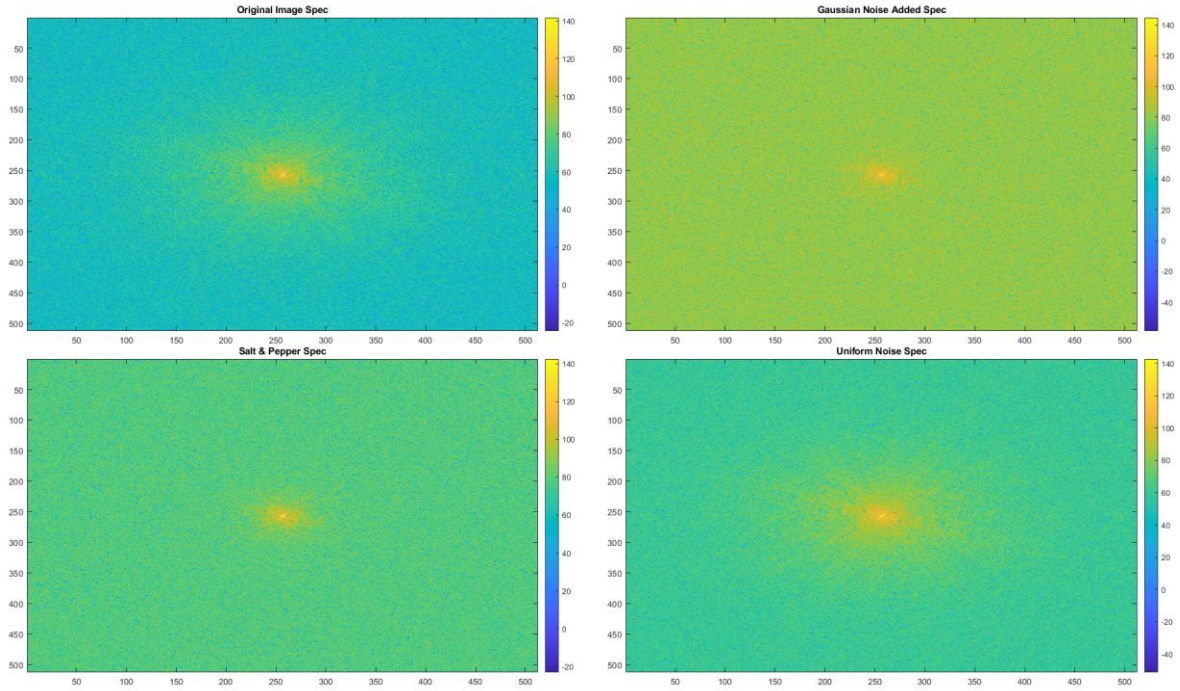

Figure 5

As we have seen, the periodic noise was added to the image which in the spatial domain the effect of this added noise completely corrupted the image seen in figure (3). Comparing the spectrum with the original image and the periodic noise added image shown in figure (1), the added periodic noise manifests itself in spectrally as two bursts of high amplitude energy. Thereafter, a notch filter, whose kernel is shown in figure (2), can be constructed whose centre frequency is merely shifted to the location of these high frequency bursts.

Some inspection of the spectrum was required to see where to shift the frequencies of the notch filter because the shift Uk and Vk is horizontal and vertical shifting with respect to the centre of the frequency rectangle. A gaussian kernel was used with a very sharp cut off because it is smooth, so no ringing artefacts are introduced in the filtering process.

The MSE for the corrupted image (without filtering) is 11642 which is very high. The filtered MSE has a result of 4.5 which is significantly less by a factor of 2500. The structurally similar index, another measure of image quality, shows a value of 0.8 (closer to 1 is better) for the filtered result and a value of

Nikeet Pandit

214118806

Assignment 3

Harmonic Analysis and Image Processing

0 for the corrupted image. Clearly, the effect of the removal of periodic noise is very efficient using notch filters.

When other types of noise are added, namely salt and pepper, gaussian, and uniform, there is a change in the resultant spectrum compared to the original, as showed by the introduction of additional higher frequency component amplitude power in figure (5). Although the uniform noise spectrum looks very similar to the original, this is likely due to a relatively small b value for the addition of uniform noise.

This example is to show is that that while a low pass filter would likely improve image quality and may remove some of the added noise (at the cost of edge detail), the effect of the noise that is not completely periodic is not as obvious to spot or pinpoint exactly in the spectrum what the noise is. Therefore, it is not easy (or impossible) to be completely removed using frequency domain filtering alone. In these cases, perhaps spatial filters such as the adaptive mean filters would be a better option. Choosing variable amounts of noise yields similar results as the test shown in this test.

Wiener & CLS & Inverse Filtering

In this test, two degradation or point spread functions (PSF) will be tested. Namely, a blurring function and a motion function is used. For both the blurring function and the motion function, there will be tests with salt and pepper noise, gaussian noise, and uniform noise. For the motion degradation function, there will be more noise inputted then the blurring function. Differences will be examined. Estimates for the noise function will be made and compared to the actual noise inputted. The filters tested for this series of experiments will be the inverse filter, constrained LS filter, and the Wiener filter. Functions used in these tests are appended at the appendix.

Reading Image

```
Im = im2double(imread("rose512.tif"));
```

Simulate motion and blur PSF

```
%--- Blurring degredation function
hsize = 10; sigma = 5;
H_blur = fspecial('gaussian', hsize, sigma);

%--- Applying blur degredation function to image
Im_blur = imfilter(Im, H_blur,'conv','circular');

%--- Motion degredation function
len = 25; theta =  60;
H_motion = fspecial('motion', len, theta);
```

Nikeet Pandit

214118806

Assignment 3

Harmonic Analysis and Image Processing

```
%--- Applying motion degredation function to image
Im_motion = imfilter(Im, H_motion,'conv','circular');

%--- Plotting degraded images alongside original
fh = figure(1); fh.WindowState = 'maximized';; subplot_tight(1,3,1); imshow(Im);
title('Original');
subplot_tight(1,3,2); imshow(Im_blur); title('Blurred');
subplot_tight(1,3,3); imshow(Im_motion); title('Motion');
```

Figure 6

Add noise to degraded images

S&P, Gaussian, Uniform Noise are added

```
ab_gauss = [0, 0.01]; %mean 0, variance 0.01 gaussian noise
d_sp = 0.05; %density 0.05 S&P noise (equal probability);
ab_uform = [0, 0.1]; %a = 0, b = 0.1 uniform noise

%--- adding noise to blurred degraded image
[~, Im_SP_blur, Im_Gaus_blur, Im_Uform_blur] = add_noise_func(Im_blur, ab_gauss, d_sp, ab_uform);

fh = figure(2); fh.WindowState = 'maximized';
subplot_tight(1,3,1); imshow(Im_SP_blur); title('S&P Noise + Blur');
```

Nikeet Pandit

214118806

Assignment 3

Harmonic Analysis and Image Processing

```
subplot_tight(1,3,2); imshow(Im_Gaus_blur); title('Gaussian Noise + Blur');
subplot_tight(1,3,3); imshow(Im_Uform_blur); title('Uniform Noise + Blur');
```

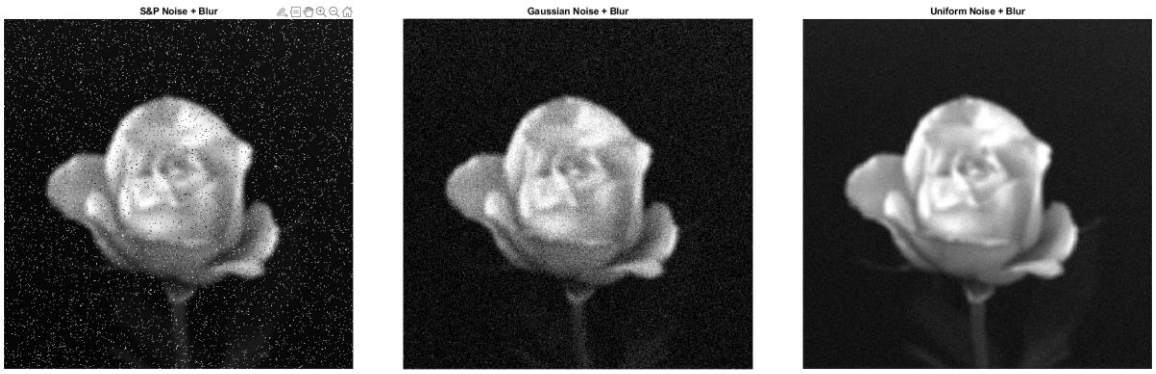

Figure 7

```
%--- S&P, Gaussian, Uniform Noise are added
%--- Please note there is much more noise added for the motion PSF then blur PSF
ab_gauss1 = [0, 0.01]*10; %mean 0, variance 0.1 gaus noise
d_sp1 = 0.1; %density 0.1 S&P noise (equal probability);
ab_uform1 = [0, 0.15]; %a = 0, b = 0.15 uniform noise

%--- adding noise to motion degraded image
[~, Im_SP_motion, Im_Gaus_motion, Im_Uform_motion] = add_noise_func(Im_motion, ab_gauss1, d_sp1,
ab_uform1);

fh = figure(3); fh.WindowState = 'maximized';
subplot_tight(1,3,1); imshow(Im_SP_motion); title('S&P Noise + Motion');
subplot_tight(1,3,2); imshow(Im_Gaus_motion); title('Gaussian Noise + Motion');
subplot_tight(1,3,3); imshow(Im_Uform_motion); title('Uniform Noise + Motion');
```

Nikeet Pandit

214118806

Assignment 3

Harmonic Analysis and Image Processing

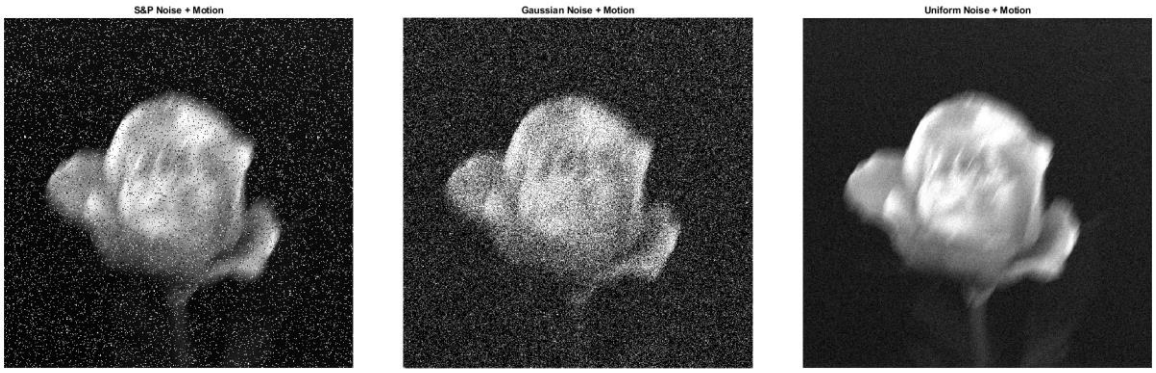

Figure 8

Figures (1) through (3) show the original image with the blurring and motion PSF with the different kinds of noise tested, namely salt and pepper noise, gaussian noise and uniform noise.

In the next segment of code, I perform tests to estimate the noise variance from the degraded noisy image. The method employed is documented in the comments of the code segments. Although the estimates are quite close to the real noise variances, some interesting conclusions will be made. Specifically, it will be seen that the Wiener filter is requires very accurate knowledge of the noise and signal variance and if this condition is not met, the restoration will be very much less than optimal.

Estimate noise variance from degraded noisy image

```
%--- I use a low pass filter on the degraded / noisy images
% After, I use a difference filter with the original degraded (with noise)
% imaged and the low pass filter to highlight high frequency differences
% between the two.

% Since degredation functions are introducing blurs we would expect any
% high frequency difference between the degredated image and the smoothed
% degredation image would attributed to the noise. Therefore, I use this
% as noise estimate and compared to the actual value of the noise variances to
% see its efficacy as a noise variance estimate.

%--- Smoothing blur degredation image with noise added
```

Nikeet Pandit

214118806

Assignment 3

Harmonic Analysis and Image Processing

```
%--- And returning difference filter (high pass)
%--- ... = original(degraded  %noise) - smoothed(degraded + noise)

DO = 120; %cut off frequency
[Im_Filter_Gaus_blur, Im_Diff_Gaus_blur, ~, ~, ~, ~] = lowpass_im(Im_Gaus_blur, DO, 'gaussian',
'FFT'); %Smoothing gaussian noise added image
[Im_Filter_SP_blur, Im_Diff_SP_blur, ~, ~, ~, ~] = lowpass_im(Im_SP_blur, DO, 'gaussian', 'FFT');
%Smoothing S&P noise added image
[Im_Filter_Uform_blur, Im_Diff_Uform_blur, ~, ~, ~, ~] = lowpass_im(Im_Uform_blur, DO,
'gaussian', 'FFT'); %Smoothing uniform noise added image

%--- Extracting quadrants of image (lower right, lower left, upper right, upper left)
%... then averaging to take noise estimate where there is only background

Im_Diff_Gaus_blur = quadrant_avg_func(Im_Diff_Gaus_blur);
Im_Diff_Uform_blur = quadrant_avg_func(Im_Diff_Uform_blur);

%--- For S&P taking only upper left quadrant
Im_Diff_SP_blur = rescale(uint8((Im_Diff_SP_blur(1:100,1:100))),0,255);

%--- Plotting Noise Patches Where Noise Variance is Est from
fh = figure(4); fh.WindowState = 'maximized';
subplot_tight(1,3,1); imshow(Im_Diff_SP_blur); title('Salt Noise + Blur');
subplot_tight(1,3,2); imshow(Im_Diff_Gaus_blur); title('Gaussian Noise + Blur');
subplot_tight(1,3,3); imshow(Im_Diff_Uform_blur); title('Uniform Noise + Blur');
suptitle('Noise Patch Plots (noise var is est. from here)')
```

Nikeet Pandit

214118806

Assignment 3

Harmonic Analysis and Image Processing

Figure 9

Although the noise patch is much less noticeable especially for the Gaussian noise and the uniform noise, it is present but may be hard to distinguish depending on the display device that this report is read on.

```
%--- Calculating Pepper Noise (probability) all other intensities except 255 set to 0
ProbOfSalt = numel(find(Im_Diff_SP_blur == 255))/numel(Im_Diff_SP_blur);

%--- Equal salt and pepper noise assumption
d_sp_est = ProbOfSalt *2;

%--- Estimating noise variance from difference filtered image
var_est_gaus_blur = var(Im_Diff_Gaus_blur(:));
var_est_SP_blur = SP_variance_calc(d_sp_est);
var_est_Uform_blur = var(Im_Diff_Uform_blur(:));

Est_Blur_Var = [var_est_gaus_blur var_est_SP_blur var_est_Uform_blur];

%--- Getting real noise variance based on inputted values
var_real_gaus_blur = ab_gauss(2);
var_real_SP_blur =   SP_variance_calc(d_sp);
var_real_Uform_blur = uform_var_calc(ab_uform(1),ab_uform(2));

Real_Blur_Var = [var_real_gaus_blur var_real_SP_blur var_real_Uform_blur];
```

Nikeet Pandit

214118806

Assignment 3

Harmonic Analysis and Image Processing

```
%--- Calculating the difference between estimate and actual
Diff_Blur_Var = abs(Real_Blur_Var - Est_Blur_Var);
```

Estimate noise variance from degraded (and motion added) image

```
%--- Smoothing blur degredation image with noise added
%--- And returning difference filter (high pass)
%--- ... = original(degraded  %noise) - smoothed(degraded + noise)

DO = 120; %cut off frequency
[Im_Filter_Gaus_motion, Im_Diff_Gaus_motion, ~, ~, ~, ~] = lowpass_im(Im_Gaus_motion, DO,
'gaussian', 'FFT'); %Smoothing gaussian noise added image
[Im_Filter_SP_motion, Im_Diff_SP_motion, ~, ~, ~, ~] = lowpass_im(Im_SP_motion, DO, 'gaussian',
'FFT'); %Smoothing S&P noise added image
[Im_Filter_Uform_motion, Im_Diff_Uform_motion, ~, ~, ~, ~] = lowpass_im(Im_Uform_motion, DO,
'gaussian', 'FFT'); %Smoothing uniform noise added image

%--- Extracting quadrants of image (lower right, lower left, upper right, upper left)
%... then averaging to take noise estimate where there is only background

Im_Diff_Gaus_motion = quadrant_avg_func(Im_Diff_Gaus_motion);
Im_Diff_Uform_motion = quadrant_avg_func(Im_Diff_Uform_motion);

%--- For S&P taking only upper left quadrant
Im_Diff_SP_motion = rescale(uint8((Im_Diff_SP_motion(1:100,1:100))),0,255);

%--- Plotting Noise Patches Where Noise Variance is Est from
fh = figure(5); fh.WindowState = 'maximized';
subplot_tight(1,3,1); imshow(Im_Diff_SP_motion); title('S&P Noise + Motion');
subplot_tight(1,3,2); imshow(Im_Diff_Gaus_motion); title('Gaussian Noise + Motion');
subplot_tight(1,3,3); imshow(Im_Diff_Uform_motion); title('Uniform Noise + Motion');
suptitle('Noise Patch Plots (noise var is est. from here)')
```

Nikeet Pandit

214118806

Assignment 3

Harmonic Analysis and Image Processing

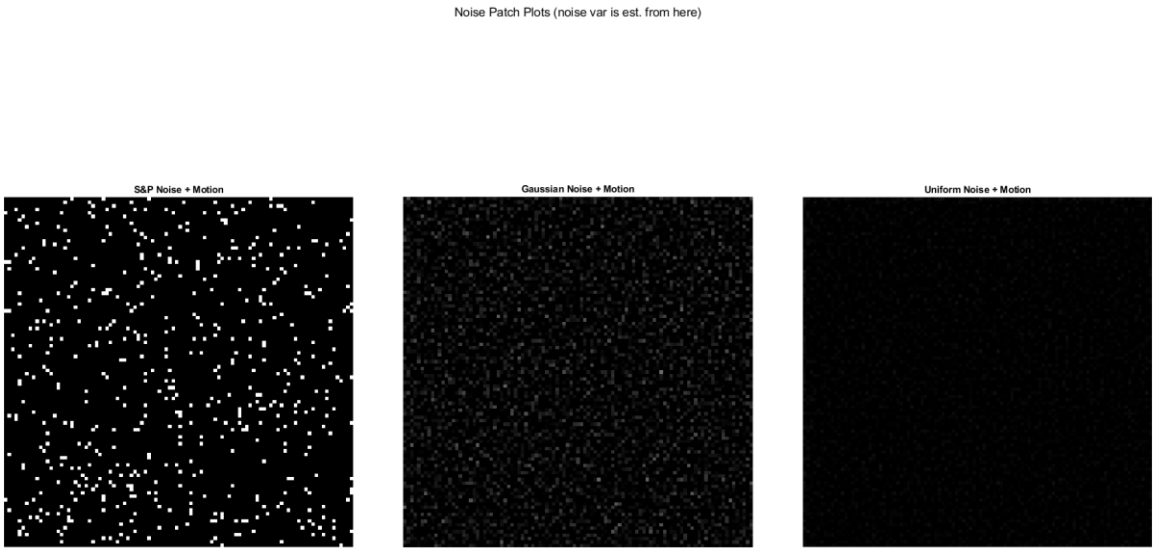

Figure 10

```
%--- Calculating Pepper Noise (probability) all other intensities except 255 set to 0
ProbOfSalt = numel(find(Im_Diff_SP_motion == 255))/numel(Im_Diff_SP_motion);

%--- Equal salt and pepper noise assumption
d_sp_est = ProbOfSalt *2;

%--- Estimating noise variance from difference filtered image
var_est_gaus_motion = var(Im_Diff_Gaus_motion(:));
var_est_SP_motion = SP_variance_calc(d_sp_est);
var_est_Uform_motion = var(Im_Diff_Uform_motion(:));

Est_motion_Var = [var_est_gaus_motion var_est_SP_motion var_est_Uform_motion];

%--- Getting real noise variance based on inputted values
var_real_gaus_motion = ab_gauss1(2);
var_real_SP_motion =   SP_variance_calc(d_sp1);
var_real_Uform_motion = uform_var_calc(ab_uform1(1),ab_uform1(2));

Real_motion_Var = [var_real_gaus_motion var_real_SP_motion var_real_Uform_motion];

%--- Calculating the difference between estimate and actual
Diff_motion_Var = abs(Est_motion_Var-Real_motion_Var);
```

Nikeet Pandit

214118806

Assignment 3

Harmonic Analysis and Image Processing

Testing Inverse Filter

Inverse filter uses the image and the PSF No estimated noise is provided, except knowledge of the PSF is required. Tests will be done for the two PSF with noise and without noise to see when the Wiener filter works well.

```
%--- Test without noise and use wiener filter
Im_recov_Blur = deconvwnr(Im_blur, H_blur);
Im_recov_Motion = deconvwnr(Im_motion, H_motion);

%--- Test without noise but adding noise (uncertainty) to knowledge of PSF
H_blur_noisy = H_blur + 0.0005*rand(size(H_blur,1),size(H_blur,2));
H_motion_noisy = H_motion .*(1e-1*rand(size(H_motion,1),size(H_motion,2))) + H_motion;

Im_recov_Blur_uncertain = deconvwnr(Im_blur, H_blur_noisy);
Im_recov_Motion_uncertain = deconvwnr(Im_motion, H_motion_noisy);

fh = figure(6); fh.WindowState = 'maximized';
subplot_tight(2,2,1); imshow(Im_recov_Blur); title('Blurred PSF (known exact)');
subplot_tight(2,2,2); imshow(Im_recov_Motion); title('Motion PSF (known exact)');
subplot_tight(2,2,3); imshow(Im_recov_Blur_uncertain); title('Blurred PSF (uncertain)');
subplot_tight(2,2,4); imshow(Im_recov_Motion_uncertain); title('PSF (uncertain)');
suptitle('Restorations');
```

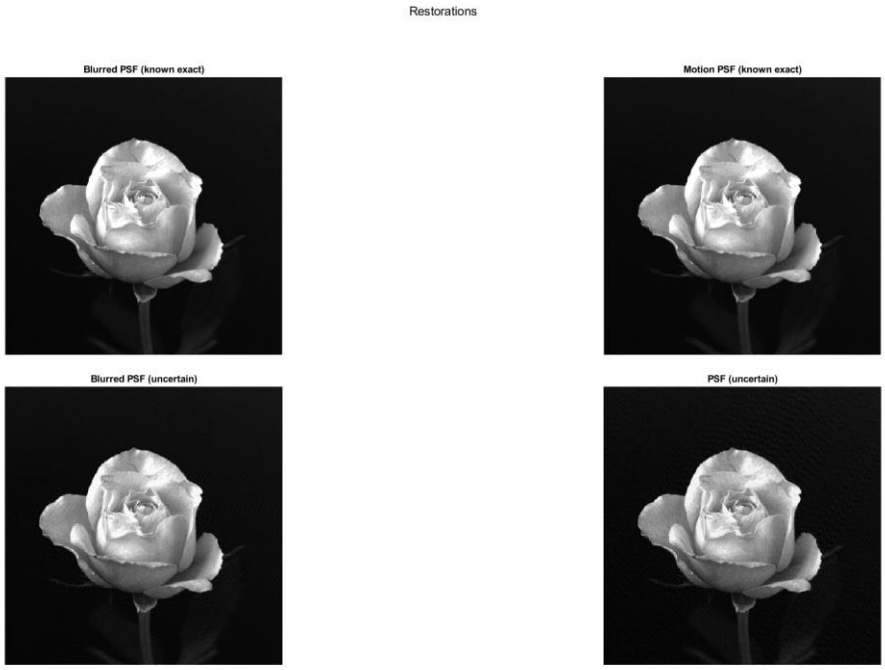

Figure 11

Nikeet Pandit

214118806

Assignment 3

Harmonic Analysis and Image Processing

As we can see, exact knowledge of the PSF is essential. Even a bit of noise added to the PSF causes distortion to restoration of the original image for inverse filtering. When knowledge of the PSF is known exactly, inverse filtering provides a MSE of 6.8e-7 and a SSI (structural similarity index) of 0.99, showing a near perfect score. When visually comparing these restored with the original image there is no visual difference either. So, perfect reconstruction is achieved with perfect knowledge of the PSF under the presence of no noise. When, however, the PSF are not known exactly the MSE increases to ~5.5e-4 for both PSF and the SSI decreases to 0.93 for the blur PSF and 0.84 for the motion PSF. Visually, there is also distortion present, especially for the motion PSF which shows rippling in the black background. Again, this is more noticeable on the MATLAB figures and less obvious on the appended results in the report. Although, the quality apparent as indicated by the quality metrics.

Next, tests will be done for inverse filtering when there is noise and a degradation function to the input image.

```
%--- Test with noise for blur PSF
Im_recov_Blur_SP = deconvwnr(Im_SP_blur, H_blur);
Im_recov_Blur_Gaus = deconvwnr(Im_Gaus_blur, H_blur);
Im_recov_Blur_Uform = deconvwnr(Im_Uform_blur, H_blur);

fh = figure(7); fh.WindowState = 'maximized';
subplot_tight(1,3,1); imshow(Im_recov_Blur_SP); title('S&P Noise added');
subplot_tight(1,3,2); imshow(Im_recov_Blur_Gaus); title('Gaus Noise added');
subplot_tight(1,3,3); imshow(Im_recov_Blur_Uform); title('Uniform Noise added');
suptitle('Restorations From Blur PSF');
```

Nikeet Pandit

214118806

Assignment 3

Harmonic Analysis and Image Processing

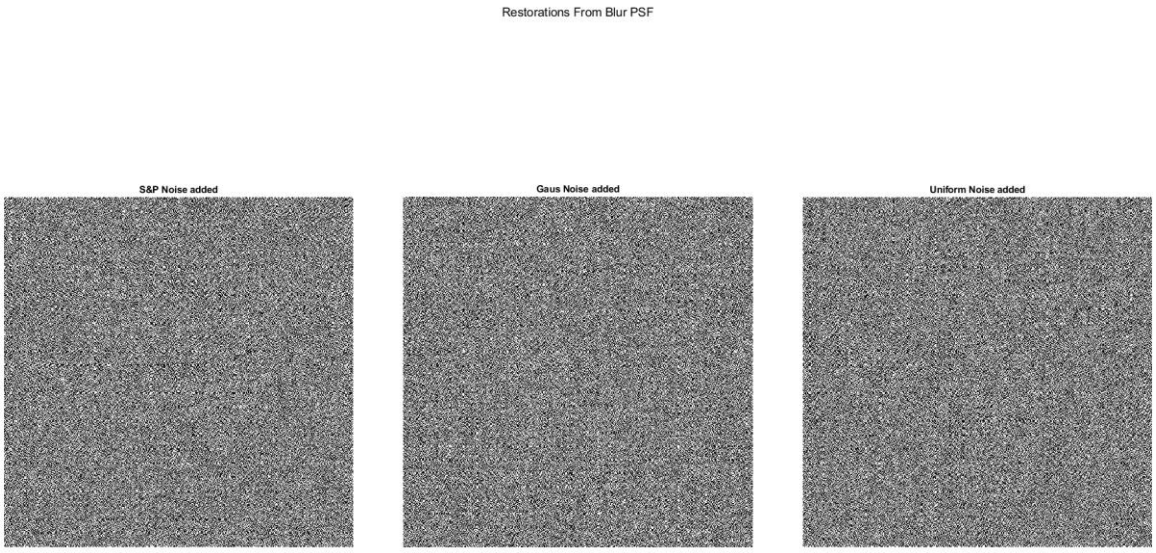

Figure 12

As we can see, inverse filtering completely breaks down under the presence of noise and the image cannot be restored at all! The same process happens when using the motion PSF and those results will not be appended for brevity. Overall, these experiments demonstrate the linear filtering is great when the PSF is known exactly under the presence of no noise to restore the original image. Otherwise, inverse filtering is not a suitable option.

Next, there will be tests for Wiener testing for the different PSF under different noises and different levels of noises. Also, there will be tests to see how sensitive to uncertainty Wiener filtering is to the noise parameters, namely the NSR, and the ACF for the noise and the ACF for the signal. In figure (8), the "true" values for the NSR are inputted, using knowledge of original image and the inputted variance for the different types of noise. In figure (9), estimates for the noise are made from the noise patches as determined and shown in figure (4) and figure (5).

Testing Wiener Filtering

```
%--- Test with noise for blur PSF

Im_recov_Blur_SP = deconvwnr(Im_SP_blur, H_blur,var_real_SP_blur/var(Im(:)));
Im_recov_Blur_Gaus = deconvwnr(Im_Gaus_blur, H_blur, var_real_gaus_blur/var(Im(:)));
Im_recov_Blur_Uform = deconvwnr(Im_Uform_blur, H_blur, var_real_Uform_blur/var(Im(:)));

fh = figure(8); fh.WindowState = 'maximized';
```

Nikeet Pandit

214118806

Assignment 3

Harmonic Analysis and Image Processing

```
subplot_tight(1,3,1); imshow(Im_recov_Blur_SP); title('S&P Noise added');
subplot_tight(1,3,2); imshow(Im_recov_Blur_Gaus); title('Gaus Noise added');
subplot_tight(1,3,3); imshow(Im_recov_Blur_Uform); title('Uniform Noise added');
suptitle('Wiener - Blur PSF with known NSR');
```

Figure 13

```
%--- Test with noise for blur PSF using estimated SNR
Im_recov_Blur_SP = deconvwnr(Im_SP_blur, H_blur,var_est_SP_blur/var(Im_SP_blur(:)));
Im_recov_Blur_Gaus = deconvwnr(Im_Gaus_blur, H_blur, var_est_gaus_blur/var(Im_Gaus_blur(:)));
Im_recov_Blur_Uform = deconvwnr(Im_Uform_blur, H_blur, var_est_Uform_blur/var(Im_Uform_blur(:)));

fh = figure(9); fh.WindowState = 'maximized';
subplot_tight(1,3,1); imshow(Im_recov_Blur_SP); title('S&P Noise added');
subplot_tight(1,3,2); imshow(Im_recov_Blur_Gaus); title('Gaus Noise added');
subplot_tight(1,3,3); imshow(Im_recov_Blur_Uform); title('Uniform Noise added');
suptitle('Wiener - Blur PSF with est.  noise power and signal power');
```

Nikeet Pandit

214118806

Assignment 3

Harmonic Analysis and Image Processing

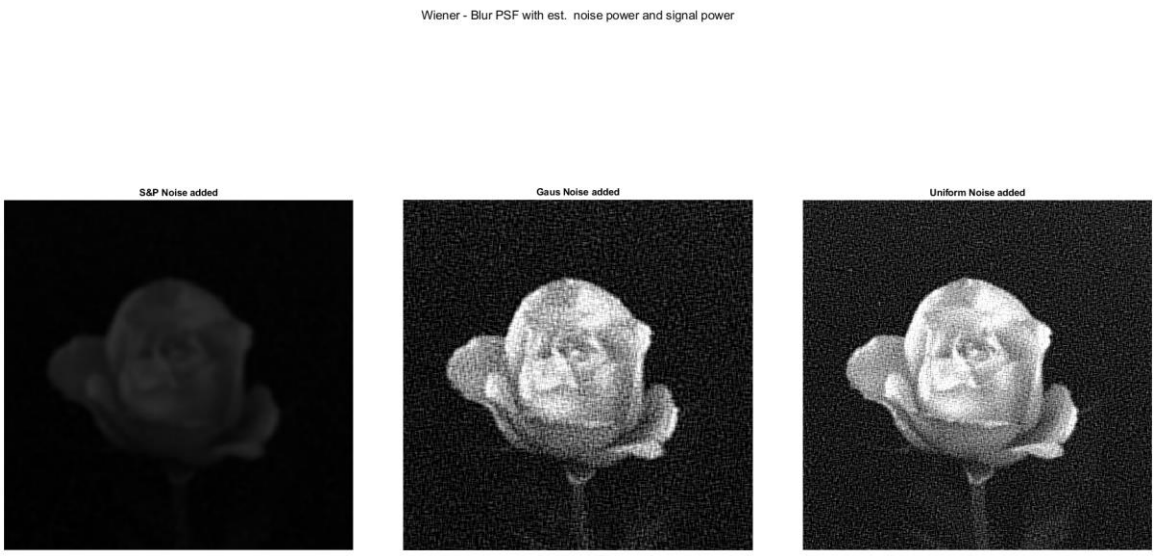

Figure 14

As these tests show, for Wiener filtering, exact knowledge of the NSR knowledge of the variance of the original signal and its noise is critical. Although the estimated noise is very close to the actual inputted noise (estimate noise variance has less then 0.01 absolute difference from the real or even less then for all estimates).

Even with knowledge of the input signal variance (original image) and variances for all the noise (i.e., inputted parameters), the reconstruction is not perfect. For S&P noise the SSI is 0.72 and the MSE is 0.02. For Gaussian noise, the SSI is 0.65 and the MSE is 0.0035. For unform noise, the SSI is 0.3 and the MSE is 0.0054. Interestingly, there is an inverse relationship for the tests between the SSI and MSE. Theoretically, the for a better image the SSI should be higher the MSE should be lower, however, this relationship is not maintained as expected. Visually, however, the images are not very well reconstructed, and the effect of noise is obvious, especially the S&P which has a very low dynamic range despite its high SSI.

For the test using estimated NSR ratio, the effect is visual as well as reflected strongly in the SSI with much lower values than compared to when using the known NSR ratio.

In the next test, the Wiener estimate is attempted to be improved using the ACF for the noise and signal.

Nikeet Pandit

214118806

Assignment 3

Harmonic Analysis and Image Processing

```
% --- Try improving the weiner estimate by using acf of noise and original image
% ifft2 of power spectrum yields acvf... normalized by rms squared then gives acf

%--- Noise is determined by subtracting noise added image by original
acf_Image = (ifft2((fft2(abs(Im).^2)/numel(Im))))/(rms(Im(:))^2);
acf_NoiseSP = (ifft2(fft2(abs(Im_SP_blur - Im).^2))/numel(Im))/rms(Im(:))^2;
acf_NoiseGaus = (ifft2(fft2(abs(Im_Gaus_blur - Im).^2))/numel(Im))/rms(Im(:))^2;
acf_NoiseUform = (ifft2(fft2(abs(Im_Uform_blur - Im).^2))/numel(Im))/rms(Im(:))^2;

%--- Using Wiener filter
Im_recov_Blur_SP = deconvwnr(Im_SP_blur, H_blur,acf_NoiseSP, acf_Image);
Im_recov_Blur_Gaus = deconvwnr(Im_Gaus_blur, H_blur,acf_NoiseGaus, acf_Image);
Im_recov_Blur_Uform = deconvwnr(Im_Uform_blur, H_blur,acf_NoiseUform, acf_Image);

fh = figure(10); fh.WindowState = 'maximized';
subplot_tight(1,3,1); imshow(Im_recov_Blur_SP); title('S&P Noise added');
subplot_tight(1,3,2); imshow(Im_recov_Blur_Gaus); title('Gaus Noise added');
subplot_tight(1,3,3); imshow(Im_recov_Blur_Uform); title('Uniform Noise added');
suptitle('Wiener Restoration - Blur PSF with ACF of noise and signal');
```

Figure 15

When inputting the ACF (estimated through the IFFT2) visually the results faired much better compared to figure (8) and figure (9), especially for the uniform noise. Now, using only the SSI for the performance

metric (seemed to fair better and correspond better my visual preference than MSE), the SSI for the S&P noise, gaussian noise, and uniform noise are 0.68, 0.61, 0.6. Although not reflected in the SSI, I prefer the restoration using the ACF for gaussian noise and the S&P noise as well. The uniform noise restoration does particularly well compared to the others.

Testing Wiener Filtering with Motion PSF (and more noise)

Now, there are tests for the motion PSF with using knowledge of the ACF for the signal and noise which seemed to offer the best restoration result of the original signal. For the motion PSF, I added significantly more noise to these images.

```
%--- Noise is determined by subtracting noise added image by original
acf_Image = (ifft2((fft2(abs(Im).^2)/numel(Im))))/(rms(Im(:))^2);
acf_NoiseSP = (ifft2(fft2(abs(Im_SP_motion - Im).^2))/numel(Im))/rms(Im(:))^2;
acf_NoiseGaus = (ifft2(fft2(abs(Im_Gaus_motion - Im).^2))/numel(Im))/rms(Im(:))^2;
acf_NoiseUform = (ifft2(fft2(abs(Im_Uform_motion - Im).^2))/numel(Im))/rms(Im(:))^2;

%--- Using Wiener filter
Im_recov_motion_SP = deconvwnr(Im_SP_motion, H_motion,acf_NoiseSP, acf_Image);
Im_recov_motion_Gaus = deconvwnr(Im_Gaus_motion, H_motion,acf_NoiseGaus, acf_Image);
Im_recov_motion_Uform = deconvwnr(Im_Uform_motion, H_motion,acf_NoiseUform, acf_Image);

fh = figure(11); fh.WindowState = 'maximized';
subplot_tight(1,3,1); imshow(Im_recov_motion_SP); title('S&P Noise added');
subplot_tight(1,3,2); imshow(Im_recov_motion_Gaus); title('Gaus Noise added');
subplot_tight(1,3,3); imshow(Im_recov_motion_Uform); title('Uniform Noise added');
suptitle('Wiener Restoration - Motion PSF with ACF of noise and signal');
```

Nikeet Pandit

214118806

Assignment 3

Harmonic Analysis and Image Processing

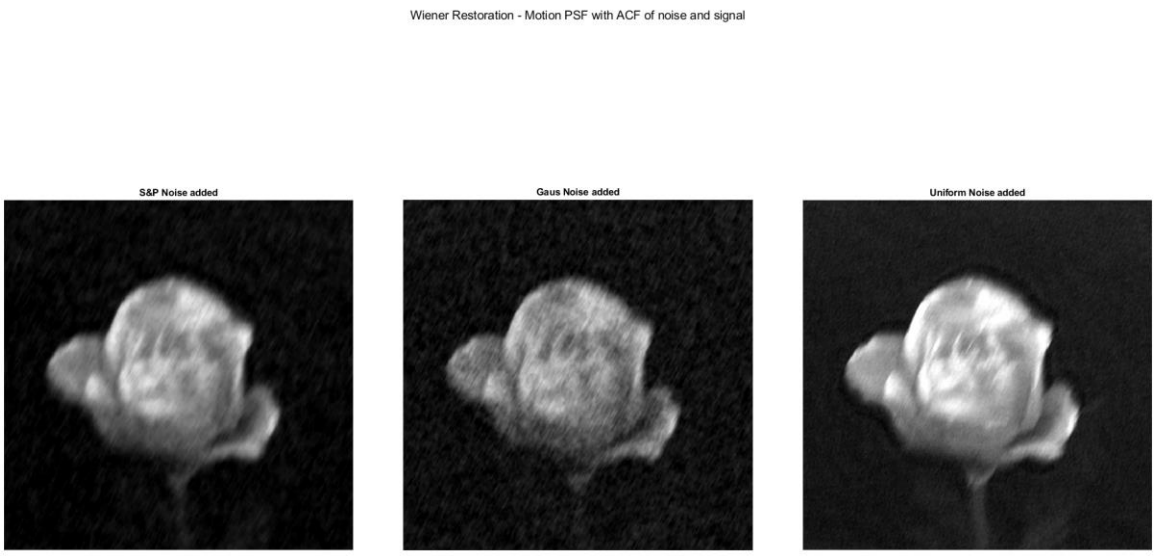

Figure 16

In this case, although the noise removal visually quite good and comparable to the previous example (which has less noise) the effect the motion is not completely removed and is present even with the full knowledge of the motion PSF, despite it being able to be removed in inverse filtering (with no noise present). All of this means and goes to show that for Wiener filtering knowledge of the signal and noise ACF and knowledge of the PSF is essential to optimally restore the image.

Testing Least Squares Constrained Filtering

Since for Wiener filtering the ACF for the noise and signal must be known, this brings additional difficulty as these may not be known and estimating them can be quite challenging. As seen in the above examples, estimates that differ even slightly from the "true" values yield restoration results that are not very effective. In the CLS, tests will be focused to see the effect of noise and see how it compares against the Wiener filtering restoration result and how sensitive the CLS is to noise power estimate uncertainty, as well as no knowledge of noise power to be able to compare against inverse filtering.

```
%--- Using LSC filter with no input for noise power (Blur PSF)
Im_recov_motion_SP = deconvreg(Im_SP_blur, H_blur);
Im_recov_motion_Gaus = deconvreg(Im_Gaus_blur, H_blur);
Im_recov_motion_Uform = deconvreg(Im_Uform_blur, H_blur);

fh = figure(12); fh.WindowState = 'maximized';
subplot_tight(1,3,1); imshow(Im_recov_motion_SP); title('S&P Noise added');
```

```
subplot_tight(1,3,2); imshow(Im_recov_motion_Gaus); title('Gaus Noise added');
subplot_tight(1,3,3); imshow(Im_recov_motion_Uform); title('Uniform Noise added');
suptitle('CLS Restoration - Blur PSF w/o input for noise power');
```

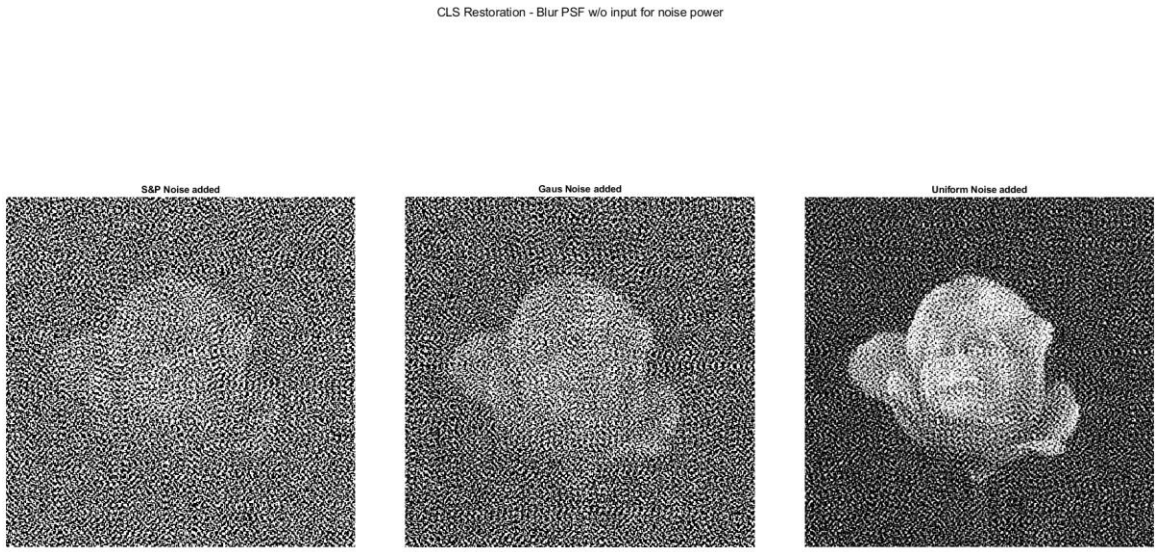

Figure 17

Comparing inverse filtering in figure (7) with CLS in figure (12), where both filters require only the input image and knowledge of the PSF, the results for the CLS are significantly better as the rose structure can at least be uncovered under the corrupting noise, compared to absolutely no rose structure in the inverse filtering example.

Next, tests will be done for CLS with no input for noise power with the motion PSF.

```
%--- Using LSC filter with no input for noise power (Motion PSF)
Im_recov_motion_SP = deconvreg(Im_SP_motion, H_motion);
Im_recov_motion_Gaus = deconvreg(Im_Gaus_motion, H_motion);
Im_recov_motion_Uform = deconvreg(Im_Uform_motion, H_motion);


fh = figure(13); fh.WindowState = 'maximized';
subplot_tight(1,3,1); imshow(Im_recov_motion_SP); title('S&P Noise added');
subplot_tight(1,3,2); imshow(Im_recov_motion_Gaus); title('Gaus Noise added');
subplot_tight(1,3,3); imshow(Im_recov_motion_Uform); title('Uniform Noise added');
suptitle('CLS Restoration - Motion PSF w/o input for noise power');
```

Nikeet Pandit

214118806

Assignment 3

Harmonic Analysis and Image Processing

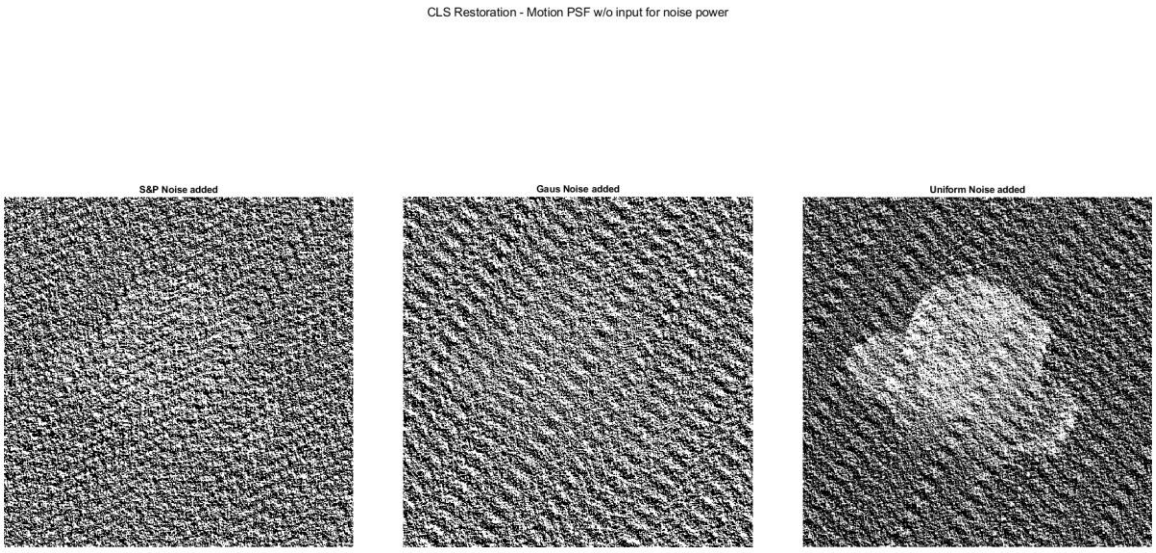

Figure 18

With more noise added and the PSF function set to motion the rose structure cannot be uncovered for gaussian noise, and the outline can only slightly be uncovered for the S&P noise without any noise power information. In the uniform noise example, the structure can barely be distinguished. Nevertheless, the CLS is a significant improvement over the inverse example, despite requiring the same number of input parameters.

In the next example, the CLS is tested with knowledge of the noise power.

```
%--- Using LSC filter with using "true" noise power values (Blur PSF)
n = numel(Im);
Im_recov_motion_SP = deconvreg(Im_SP_blur, H_blur,var_real_SP_blur*n);
Im_recov_motion_Gaus = deconvreg(Im_Gaus_blur, H_blur, var_real_gaus_blur*n);
Im_recov_motion_Uform = deconvreg(Im_Uform_blur, H_blur, var_real_Uform_blur*n);

fh = figure(14); fh.WindowState = 'maximized';
subplot_tight(1,3,1); imshow(Im_recov_motion_SP); title('S&P Noise added');
subplot_tight(1,3,2); imshow(Im_recov_motion_Gaus); title('Gaus Noise added');
subplot_tight(1,3,3); imshow(Im_recov_motion_Uform); title('Uniform Noise added');
suptitle('CLS Restoration - Blur PSF w/ input for noise power (real noise)');
```

Nikeet Pandit

214118806

Assignment 3

Harmonic Analysis and Image Processing

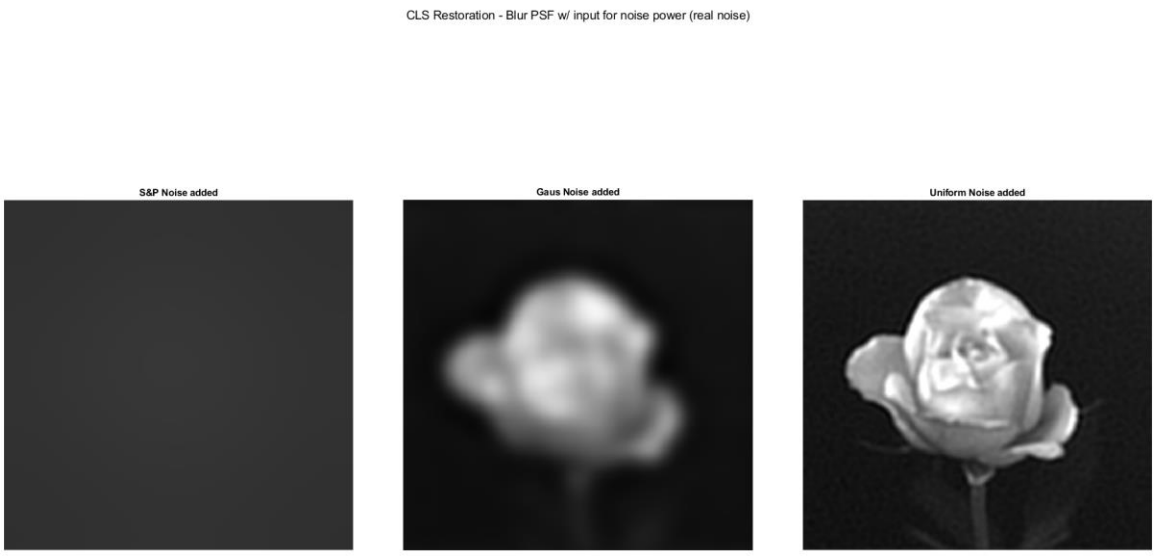

Figure 19

This example is showing clearly there is some issue with the variance calculation for the noise power calculation for the S&P noise. Even though I double checked the formula, I am not sure exactly the issue, considering it seemed to show improvements with Wiener filtering tests. As a result, the S&P noise added tests will be disregarded for the remaining CLS tests.

Including noise power into the image brings a significant improvement, however, the blurring effect is quite evident. Comparing visually, the effect is similar for uniform noise to the Wiener filter (except the Weiner filter requires ACF of signal as well, while CLS only requires noise power).

While the Gaussian noise image seems to have the noise completely removed, all the high frequency information is obviously very smoothed away. Now, there will be some experimentation with the value of noise power to yield a sharpened result. Since there is some error with the S&P variance calculation, only Gaussian and Uniform noise will be tested now.

```
fraction = 0.65; %only include fraction of noise power to sharpen result
fraction1 = 0.995;
Im_recov_motion_Gaus = deconvreg(Im_Gaus_blur, H_blur, (var_real_gaus_blur*n)*fraction);
Im_recov_motion_Uform = deconvreg(Im_Uform_blur, H_blur, fraction1*(var_real_Uform_blur*n));

figure(15);
subplot_tight(1,2,1); imshow(Im_recov_motion_Gaus); title('Gaus Noise added');
```

Nikeet Pandit

214118806

Assignment 3

Harmonic Analysis and Image Processing

```
subplot_tight(1,2,2); imshow(Im_recov_motion_Uform); title('Uniform Noise added');
suptitle('CLS Restoration - Blur PSF w/ input for noise power (adjusted noise)');
```

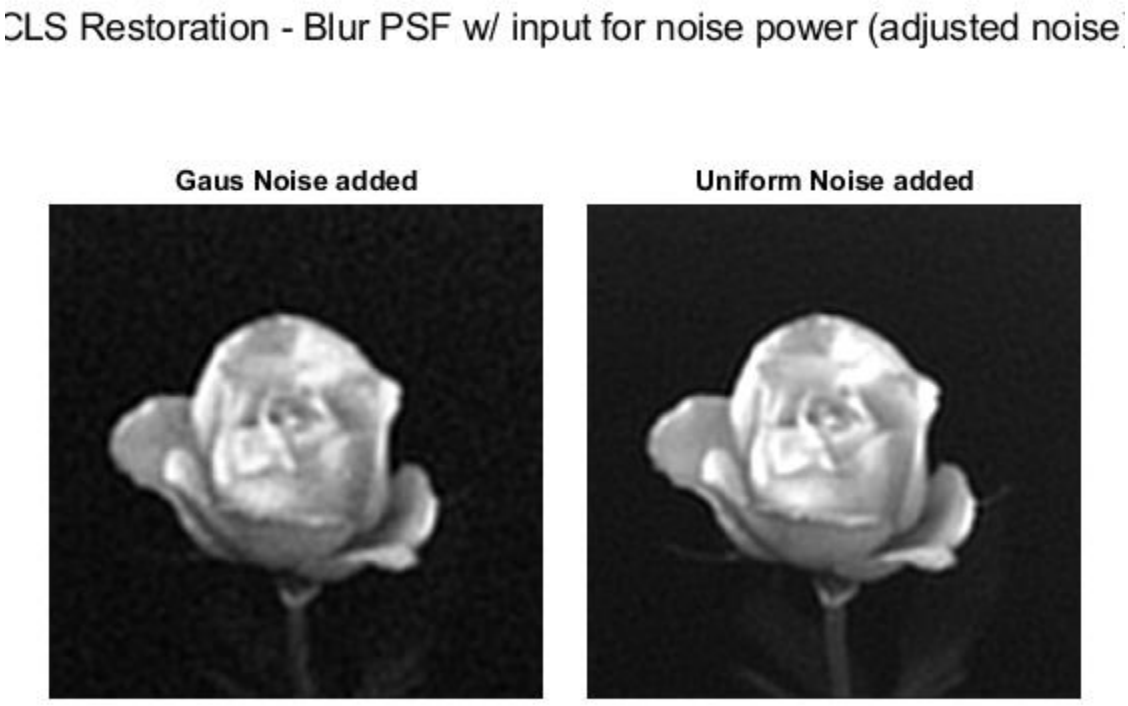

Figure 20

As we can see, variably adjusting the noise power can yield a better result. This is especially true for the Gaussian noise added example. For the uniform noise example, there is little to no improvement with variably changing the noise power. This shows that absolute knowledge of the noise power is not as necessary as the Wiener filter and is thus more flexible and can be parameterized to yield better results. Also, by being able to adjust the noise power level to trade blurring for sharpening and vice versa, the filter offers great flexibility.

```
%--- LSC Filter with noise power values (Motion PSF)
%--- Using LSC filter with using "true" noise power values (Blur PSF)
n = numel(Im);
Im_recov_motion_Gaus = deconvreg(Im_Gaus_motion, H_motion, (var_real_gaus_motion*n)*0.5);
Im_recov_motion_Uform = deconvreg(Im_Uform_motion, H_motion, var_real_Uform_motion*n);

figure(16);
```

Nikeet Pandit

214118806

Assignment 3

Harmonic Analysis and Image Processing

```
subplot_tight(1,2,1); imshow(Im_recov_motion_Gaus); title('Gaus Noise added');
subplot_tight(1,2,2); imshow(Im_recov_motion_Uform); title('Uniform Noise added');
suptitle('CLS Restoration - Motion PSF w/ input for noise power');
```

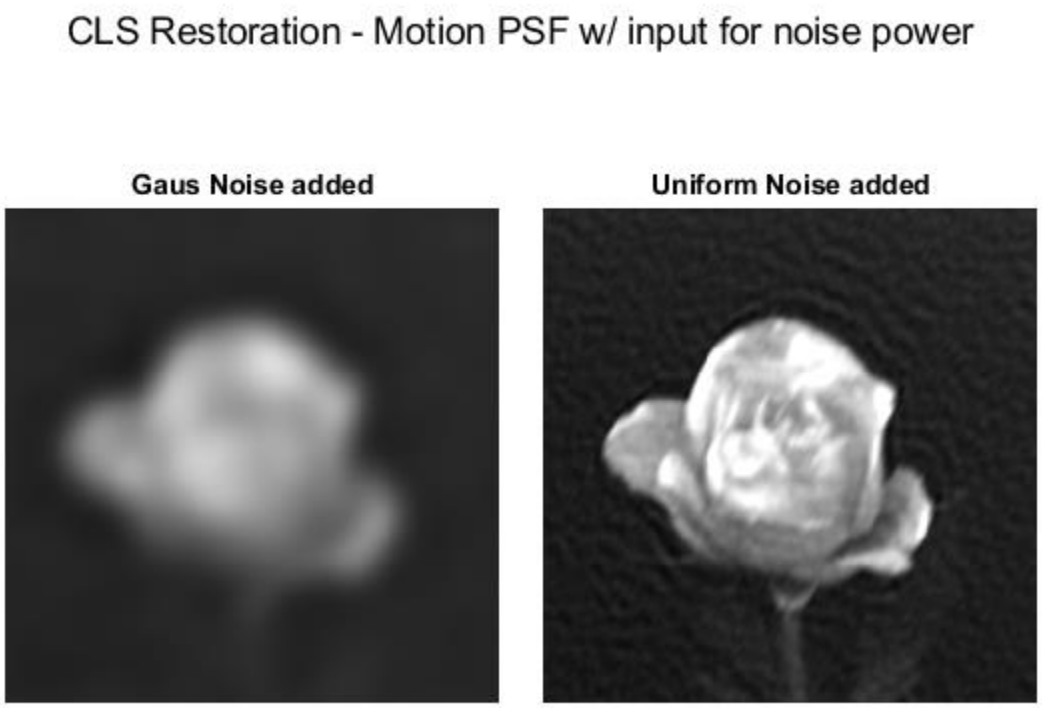

Figure 21

For both the Wiener filter and the LSC filter, when the degradation filter is motion, that is to see there is a blurring effect on top of the movement distortion, the restoration effect of the image is not as effective in both the removal of the noise and removal of the degradation function (deconvolution). Perhaps, the more complicating the degradation function, the more difficult it is for these filtering methods to restore the original image. Visually, the effect of restoration effect of the CLS for the uniform noise introduced additional distortion and rippling throughout the image, which is very apparent in the background.

The Gaussian noise added example also was very blurred and even with tinkering the noise power value, would result in either an image that is too noisy or too blurred.

All of this to say is that the Wiener filter is a much better option provided a good estimate for signal ACF and the noise ACF can be provided. This can be verified visually by comparing figures (11) and figures (16) and improvements in the SSI for the Wiener constructed images.

Nikeet Pandit

214118806

Assignment 3

Harmonic Analysis and Image Processing

However, when these results are not known, or the NSR is not very well estimated, the CLS offers a much better construction result over both Wiener filtering and inverse filtering.

Notch Kernel Construct Notch Function

```
function kernel = kernel_construct_notch(D0, Uk, Vk, M, N, type,varargin)
%%%% Inputs
% DO is cut off frequency
% Uk, Vk are notch pairs (type array)
% M and N are dimensions of kernel size
% type is class string Options are: Gaussian, butterworth
% Vargin(1) specified butterworth order (type double)

%%%% Function Sumary
%D is the distance between point (u,v) in frequency domian to notch of
%PxQ frequency rectangle (Digital Image Processing 4ed)

%Written by Nikeet Pandit

if length(varargin) ==1
    n = varargin{1};
end

if length(Uk) ~= length(Vk)
    disp('Notch pair Uk and Vk must be same length')
    return
end

type = lower(type);
u = 0:M-1; %frequency components
v = 0:N-1;
[U, V] = meshgrid(u,v);

if type == "gaussian"
    kernel_mult = zeros(M,N,length(Uk));

    for i = 1:length(Uk) %if more than 1 notch frequencies specified

        % calculating positive notch distance mat. and positive notch kernel
        D =  my_calc_dist(U,V,Uk(i),Vk(i),M,N);
        kernel_pos = fftshift(1-(exp(-(D.^2)./(2*(D0^2))))); %high pass kernel
centered around notch Uk, Vk

        % calculating negative notch distance and negative notch kernel
        D =  my_calc_dist(U,V,-Uk(i),-Vk(i),M,N);
        kernel_neg = fftshift(1-(exp(-(D.^2)./(2*(D0^2))))); %high pass kernel
centered around notch -Uk, -Vk

        % notch filter are constructed as prod. of high pass kernels (for positive and
negative notch)
        kernel_mult(:,:,i) = kernel_pos.*kernel_neg;  %symmetric kernel (about origin)
(uncentered)
    end

    %Elementwise matrix along 3rd dimension to construct kernel...
    %if more than 1 positive and negative notch specified by user
```

Nikeet Pandit

214118806

Assignment 3

Harmonic Analysis and Image Processing

```
    kernel = prod(kernel_mult,3);

elseif type == "butterworth"
    kernel_mult = zeros(M,N,length(Uk));

    for i = 1:length(Uk)
        %Same as above except with butterworth kernel
        D =  my_calc_dist(U,V,Uk(i),Vk(i),M,N);
        NumExp = D;
        DenExp = D0;
        kernel_pos = fftshift(1./(1+((NumExp./DenExp).^(2*n))))';
        D =  my_calc_dist(U,V,-Uk(i),-Vk(i),M,N);
        NumExp = D;
        DenExp = D0;
        kernel_neg = fftshift(1./(1+((NumExp./DenExp).^(2*n))))';
        kernel_mult(:,:,i) = kernel_pos.*kernel_neg;
    end

    kernel = prod(kernel_mult,3);
end
end

%--- Calculating distance matrix for trasnfer functions
function D =  my_calc_dist(U,V,Uk,Vk,M,N)
    D = hypot(U - M/2 - Uk, V - N/2 - Vk);
end
```

Quadrant Average Function

```
%--- Extracting quadrants of image (lower right, lower left, upper right, upper left)
%--- These quadrants are where there is only the background of the image
%... then averaging to take noise estimate where there is only background
%... taking average is hoped to have a more realistic noise estimate

function avg_image_noise = quadrant_avg_func(ImageIn)
cnt = 0;
quadrants = [1:101; 1:101; 1:101; 400:500; 400:500; 1:101; 400:500; 400:500];
%quadrants where there is image background from rose
for i = 1:2:size(quadrants,1)
    cnt = cnt + 1;
    avg_image_noise(:,:,cnt) = ImageIn(quadrants(i,:),quadrants(i+1,:));

    if i+1 == size(quadrants,1)
        break
    end
end
avg_image_noise = mean(avg_image_noise,3); %taking average along 3rd matrix dimension
```

Low Pass Filter Function

```
In function DFT may mean DFT or DCT!!!
```

Nikeet Pandit

214118806

Assignment 3

Harmonic Analysis and Image Processing

```
% Inputs:
% (1) Im -image file read in by Imread(File)
% (2) D0 -cut off frequency
% (3) Type - 'Gaussian', or 'Butterworth'
% (4) Transform Type 'DCT', 'FFT'
% (5) If Butterworth is selected, must select order

% Outputs:
% (1) Filtered Image
% (2) Centered Kernel
% (3) Filtered Image (Freq domain)
% (4) Original Image in Freq domain

function [Im_Filter, Im_Diff, Im_Filter_Freq, Im_DFT_return, Pt, Telapsed] =
lowpass_im(Im, D0, kernel_type, transform_type, varargin)
Tstart = tic;



transform_type = lower(transform_type);

kernel_type = lower(kernel_type);



if length(varargin) == 1 %Setting order for Butterworth if selected
    n = varargin{1};
else
    n = [];
end
```

Determine image size and calculate padded image size

```
[M, N] = size(Im); P = M*2; Q = N*2;
```

Construct kernel (frequency domain) to be used for filtering

```
H = kernel_construct(D0 , P, Q, kernel_type, n); %function is appended below
```

Take DFT of the image with padding specified by P and Q

```
if transform_type == "fft"
    Im_DFT = fft2((Im),P,Q);
    Im_DFT_return = abs(fftshift((Im_DFT))); %centered spectrum
elseif transform_type == "dct"
    Im_DFT = dct2(Im,P,Q);
    Im_DFT_return = fftshift(Im_DFT);
else
    print("Improper Selection");
```

Nikeet Pandit

214118806

Assignment 3

Harmonic Analysis and Image Processing

```
end
```

Filter in frequency domain

```
Im_Filter = (H.*Im_DFT);



if transform_type == "fft"
    Im_Filter_Freq = fftshift(abs(Im_Filter)); %returning filtered spectrum
elseif transform_type == "dct"
    Im_Filter_Freq = fftshift((Im_Filter)); %returning filtered spectrum
end
```

Check Power

```
Pt = (sum(abs(Im_Filter),'all')/(sum(abs(Im_DFT),'all')))*100;

%%Isolate real components only
if transform_type == "fft"
    Im_Filter = (real(ifft2(Im_Filter)));
elseif transform_type == "dct"
    Im_Filter = uint8(idct2(Im_Filter));
end
```

Crop Image to remove padding

```
Im_Filter = (Im_Filter(1:M, 1:N)); %returning cropped filtered image
H = fftshift(H);  %returning centered (and cropped) kernel function

Telapsed = toc(Tstart);

Im_Diff = Im - Im_Filter; %High pass filter of original and filtered
end
```

Add Noise Function

Function add noise

Inputs (1) Image (input) (2) [a, b] inputs for Gauss (3) S&P density (4) [a, b] for uniform noise

```
% Outputs
% (1) Original Image
% (2) Gauss noise added image
% (3) S&P added image
% (4) Uniform noise added image


function [Im, Im_SP, Im_Gaus, Im_Uform] = add_noise_func(Im, ab_gauss, d_sp, ab_uform)
```

Nikeet Pandit

214118806

Assignment 3

Harmonic Analysis and Image Processing

```
[M, N] = size(Im);
Im_Gaus = imnoise(Im,'gaussian',ab_gauss(1),ab_gauss(2));
Im_SP = imnoise(Im,'salt & pepper',d_sp);
Im_Uform = Im+ ab_uform(1) + (ab_uform(2) - ab_uform(1))*rand(M,N); %uniform noise
end
```

Uniform Variance Calculation Function

```
%--- Function to calculate uniform noise variance
function var_uform = uform_var_calc(a,b)
    var_uform = ((b-a)^2)/12;
end
```

S&P Variance Calculation Function

```
%--- Function to calculate S&P noise variance
function SPvar = SP_variance_calc(noise_density)
Ps = noise_density/2; Pp = Ps;
K = 2; %intensity levels
SPmean = 0*Pp + K*(1-Ps-Pp) + (2^K-1)*Ps; %from Digital Image Processing 4ed
SPvar = (0-SPmean)^2*Pp + (K-SPmean)^2*(1-Ps-Pp) + (2^K-1)^2*Ps;
end
```