Lab 2: Optimal Filtering

Daniel Sherman (0954083), *Student, University of Guelph*

***Abstract*—Optimal filters were designed and implemented to observe the effects of noise reduction and removing distortion from filters for test images. Frequency domain approaches and spatial domain approaches were both implemented for comparison. 11x11 and 21x21 window sizes were implemented for the spatial domain filter to observe and compare the effectiveness. It was evident that for pure noise reduction, both the spatial domain and frequency domain filter's effectiveness was near identical, but for removing the effects of filter blurring as well as noise, the frequency domain approach was superior.**

***Index Terms*—Imaging, Optimal Filtering, Optimization, Deconvolution, Wiener Filters, Least-Squares**

I. Introduction

WHEN obtaining an image, typically, the acquisition method adds some sort of distortion, no matter how small, with random noise (1). This can impede the analysis of an image, especially if important features are masked by the distortion. A massive challenge in filtering out noise in images is that the desired image and noise characteristics are not known, so designing an effective filter can be difficult. While noise characteristics can be determined using phantoms and evaluating the imaging modality's modulation transfer function, the desired clean image is still unknown.

Designing a filter to remove noise and distortion heuristically is often ineffective. Most times, the filter removes too much, or too little frequency content. For the application of noise removal, this can either leave too much noise when under filtering, or remove important image features when over filtering.

Fortunately, an effective way of estimating the unknown desired image is to minimize the mean-squared error of the desired image, using its auto- and cross-correlation functions (2). This can be done, as while the actual desired signal is unknown, the stochastic behaviour of the signal is deterministic and unchanging. This turns the incredibly difficult problem of removing random noise into an easier optimization problem, where minimizing the variance or the mean squared error is the solution to the optimal filtering issue.

In this lab, sample images were corrupted with noise and blurring filters, and by optimally designing filters, these distortions were removed the most they could. The effectiveness of the optimal filters were compared by investigating whether a frequency or spatial domain filter would work better, changing the variance of the noise, and by changing the window size of the spatial domain filters.

II. Methods

Test images were obtained and loaded into MATLAB, which can be seen in Figure 1 and Figure 2.

Lab Report received May 26, 2020

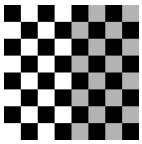

Fig. 1: Test image 'checker.png' (128x128 Pixels)

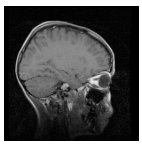

Fig. 2: Test image 'mri.jpg' (256x256 Pixels)

MATLAB code used for this lab can be found in section B.

A. Noise Reduction

Figure 1 and Figure 2 were corrupted with random noise ($\mu = 0, \sigma^2 = 100, 900$). An Optimal Wiener filter was designed to be implemented in the Frequency domain for Figure 2. The Spatial domain approach was applied to Figure 2 and to Figure 1.

The Power Spectral Density of the original image was approximated by using the magnitude squared of its Fourier Transform. The Power Spectral Density of the noise was approximated by a constant image.

The Frequency Domain filter was defined, then applied using element-wise multiplication.

The Spatial Domain Approach was also taken to apply an optimal filter, with window sizes of 11x11 and 21x21. The optimal filter was found using the auto-correlation and cross-correlation functions and applied in the spatial domain using convolution.

Image subtraction was utilized to compare the results of the frequency domain approach and the spatial domain approach to optimal filtering.

B. Deconvolution

The test images were passed through the custom Low-Pass filter defined in Equation 1 (k being a normalization factor to have DC gain of 1), then corrupted with random noise ($\mu = 0, \sigma^2 = 25, 400$).

$$h(x, y) = k(0.8)^{(|x|+|y|)} \tag{1}$$

Optimal Wiener Filters were applied in the Frequency Domain and in the Spatial Domain, much like in subsection II-A. Figure 1 was only filtered through the Spatial

Domain approach, whereas Figure 2 was filtered through both the Frequency Domain and Spatial Domain approaches.

The Frequency Domain filter needed to be applied carefully, accounting for the applied filter, and its magnitude. Other than that, the same methods were taken as mentioned above.

III. Results and Discussion

Just as would be expected from an optimal filter, the images were filtered nicely. Much of the high frequency noise content was removed, while preserving the image features that make it appealing to look at.

The frequency domain approach success is easy to see. This is seen nicely in Figure 4 and Figure 3, where the graininess of the added noise was largely removed, leaving an intelligible picture of the MRI. Because the noise has a smaller standard deviation in Figure 3, the resulting optimally filtered image was visually crisper.

Filtering in the spatial domain had good results too. The fuzziness of the high frequency noise was largely filtered out. The larger the window size, the more noise is filtered out, as there is a larger subsample to test the auto- and cross-correlation against. This is seen nicely in Figure 8, and in Figure 11, as almost all of the noise was filtered out. The effect is still seen in Figure 5, Figure 6, Figure 7, Figure 9, Figure 10, and Figure 12, but not as much.

When being applied on images that have only corrupting noise added to them, the effectiveness of the optimal filter in the frequency domain and in the spatial domain is near identical. This is seen in Figure 13 and Figure 14 where the only differences between the two subtracted images appears to be where the noise happened to be. It is worth noting that the highest grey level in Figure 13, approximately 25, is lower than that in Figure 14, approximately 40. This is due to the variance of noise being higher in Figure 14, allowing smaller numbers being subtracted from higher numbers in the image subtraction than in Figure 13 where the variance was lower.

When applying a frequency domain filter to the images that have been filtered through Equation 1, much of the image distortion was removed, but the resulting image quality is nowhere near the level of the figures listed above. This is because the images that are bring put through the optimal filter have much more distortion than just random noise. Furthermore, Figure 15 was less blurry than Figure 16, because the variance of noise is smaller, so more noise can be filtered out.

For the spatial domain approach to optimally filtering images that have been passed through Equation 1, the same trend is seen as above. The larger the window size, the more effective the filter is. This is seen in Figure 18, as it is less blurry than Figure 17. The same effect is seen in Figure 20 and Figure 19. Furthermore, as the noise increases, the less effective the filter is. This is seen comparing Figure 20 and Figure 18, the former image was much more blurry than the latter. While the filters are optimal, they are nowhere near perfect. This is seen in Figure 21, Figure 22, Figure 23, and Figure 24. A false outline is created around the image that was not in the original (Figure 1), as very high frequency content is filtered out of the image, such as the edges of each checker. More high frequency content that is removed is seen in Figure 23 and Figure 24, as the edges of each square can be visually seen as blurred.

When trying to remove a filter in addition to noise, there are differences to be seen in the optimal filter behaviour between a frequency and spatial domain approach. With a large variance of noise, and a large filter window, differences in the results are not really seen. This is why the right image in Figure 26 looks just like noise. It is worth noting that an outline of a head is just barely seen in the left image of Figure 26. As the outline of the head can be seen in Figure 25, it is evident that the frequency domain filter is more effective, as if the effectiveness of the frequency and spatial domain were the same, only noise would remain in the comparison.

IV. Conclusion

The optimal filters were successful in removing much of the noise and applied filters to the sample images, while preserving the image content.

When removing noise, the frequency domain approach and the spatial domain approach produce nearly identical effects. This is evident in the comparison images, as only noise can be seen after the image subtraction has been performed.

As the window size increased for the spatially domain approach to filtering, the more effective the filters are. This comes with a massive trade off between filter effectiveness and computation speed, as it could be visually seen that the 21x21 window took much more time to calculate than the 11x11 window.

A frequency domain approach and a spatial domain approach to removing filter effects produce nearly the same effects at high noise levels, but differs at low levels. At low noise levels, the frequency domain approach is more effective at preserving image features. This is evident in the comparison images, as more of an outline in the MRI image is seen after performing image subtraction. At high noise levels, more of the noise remains in the optimally filtered images, and the filtered images are more distorted than when noise levels were low. This is evident in the comparison between a frequency domain and a spatial domain approach to filtering, as at high noise levels, only noise remains after the image subtraction has been performed.

APPENDIX A
IMAGE RESULTS

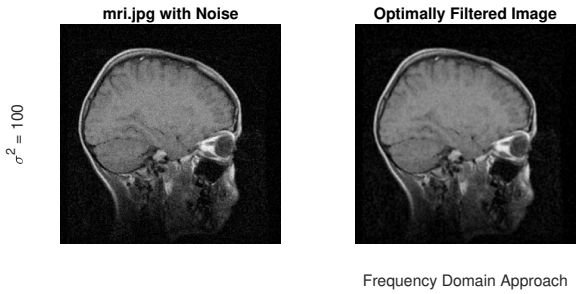

Fig. 3: 'mri.jpg' corrupted with $\sigma^2 = 100$ noise (left) and filtered through an optimal Wiener Frequency Domain Filter (right)

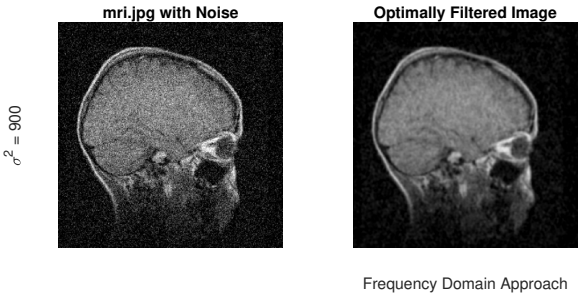

Fig. 4: 'mri.jpg' corrupted with $\sigma^2 = 900$ noise (left) and filtered through an optimal Wiener Frequency Domain Filter (right)

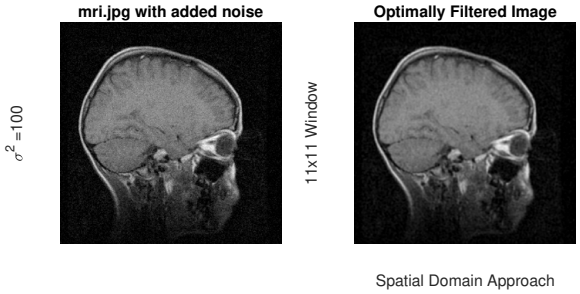

Fig. 5: 'mri.jpg' corrupted with $\sigma^2 = 100$ noise (left) and filtered through an optimal Spatial Domain Least-Squares Filter (window size: 11x11) (right)

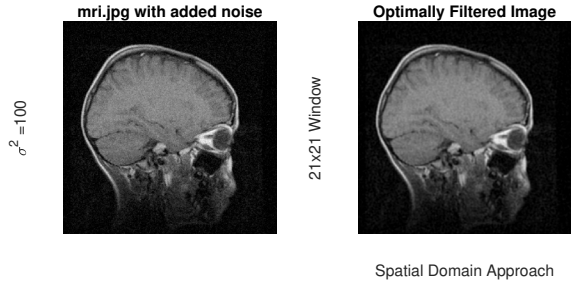

Fig. 6: 'mri.jpg' corrupted with $\sigma^2 = 100$ noise (left) and filtered through an optimal Spatial Domain Least-Squares Filter (window size: 21x21) (right)

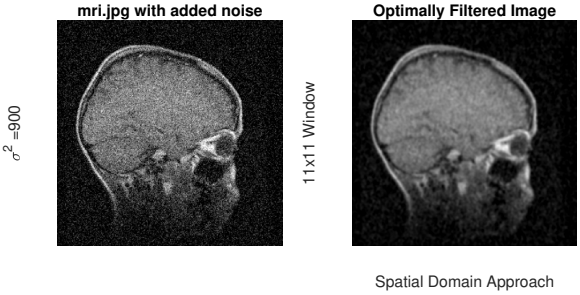

Fig. 7: 'mri.jpg' corrupted with $\sigma^2 = 900$ noise (left) and filtered through an optimal Spatial Domain Least-Squares Filter (window size: 11x11) (right)

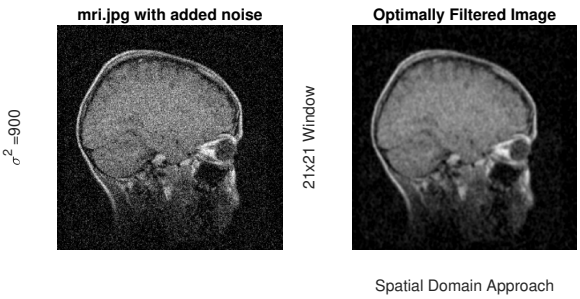

Fig. 8: 'mri.jpg' corrupted with $\sigma^2 = 900$ noise (left) and filtered through an optimal Spatial Domain Least-Squares Filter (window size: 21x21) (right)

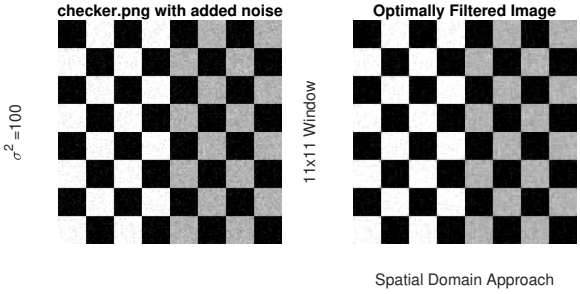

Fig. 9: 'checker.png' corrupted with $\sigma^2 = 100$ noise (left) and filtered through an optimal Spatial Domain Least-Squares Filter (window size: 11x11) (right)

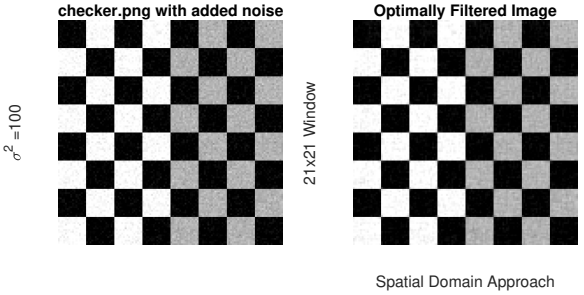

Fig. 10: 'checker.png' corrupted with $\sigma^2 = 100$ noise (left) and filtered through an optimal Spatial Domain Least-Squares Filter (window size: 21x21) (right)

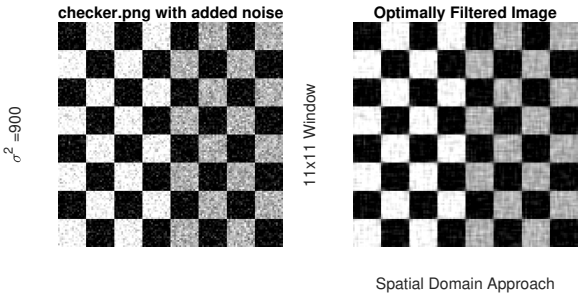

Fig. 11: 'checker.png' corrupted with $\sigma^2 = 900$ noise (left) and filtered through an optimal Spatial Domain Least-Squares Filter (window size: 11x11) (right)

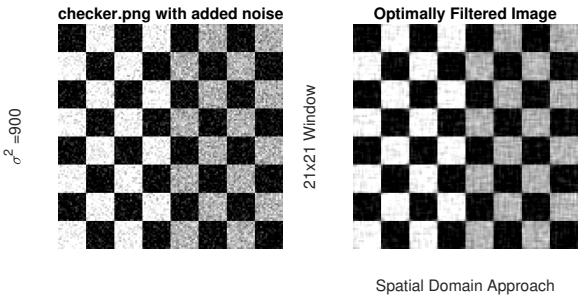

Fig. 12: 'checker.png' corrupted with $\sigma^2 = 900$ noise (left) and filtered through an optimal Spatial Domain Least-Squares Filter (window size: 21x21) (right)

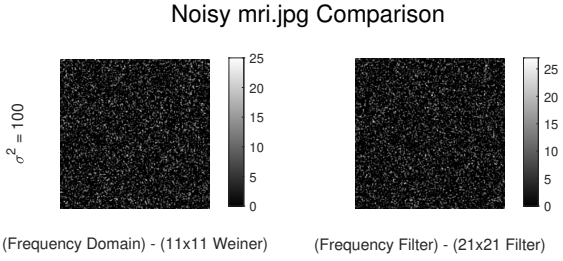

Fig. 13: Image Subtraction between 'mri.jpg' corrupted with $\sigma^2 = 100$ noise optimally filtered through the Frequency Domain and filtered in the Spatial Domain (window size: 11x11 [left] and 21x21 [right])

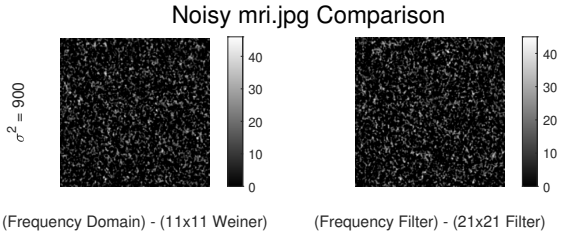

Fig. 14: Image Subtraction between 'mri.jpg' corrupted with $\sigma^2 = 900$ noise optimally filtered through the Frequency Domain and filtered in the Spatial Domain (window size: 11x11 [left] and 21x21 [right])

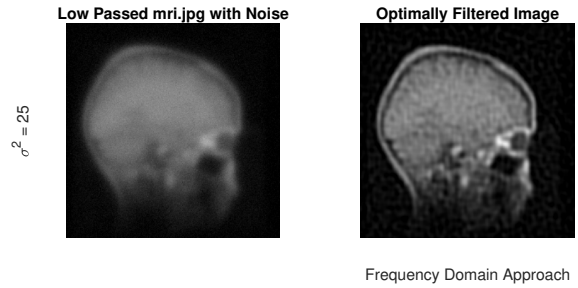

Fig. 15: 'mri.jpg' filtered through the Low-Pass Filter defined in Equation 1, corrupted with $\sigma^2 = 25$ noise (left), then optimally filtered in the Frequency Domain (right)

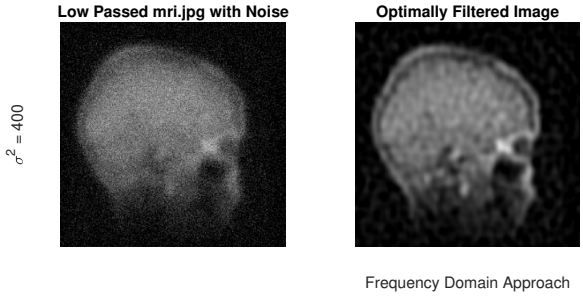

Fig. 16: 'mri.jpg' filtered through the Low-Pass Filter defined in Equation 1, corrupted with $\sigma^2 = 400$ noise (left), then optimally filtered in the Frequency Domain (right)

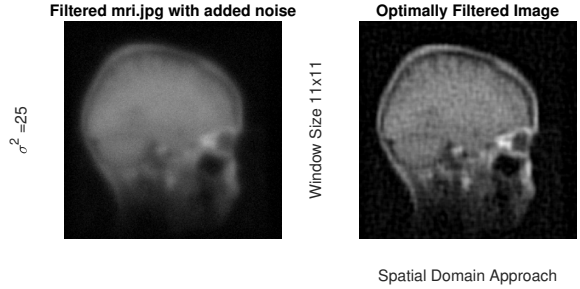

Fig. 17: 'mri.jpg' filtered through the Low-Pass Filter defined in Equation 1, corrupted with $\sigma^2 = 25$ noise (left), then optimally filtered in the Spatial Domain (window size: 11x11) (right)

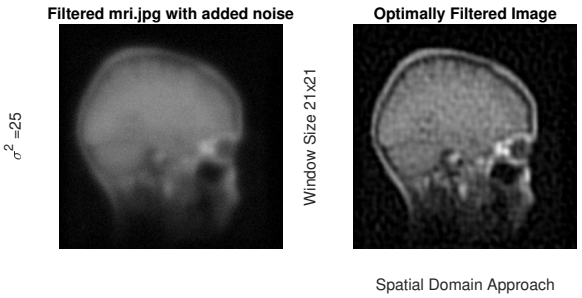

Fig. 18: 'mri.jpg' filtered through the Low-Pass Filter defined in Equation 1, corrupted with $\sigma^2 = 25$ noise (left), then optimally filtered in the Spatial Domain (window size: 21x21) (right)

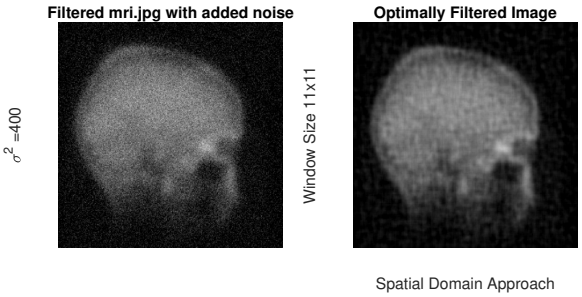

Fig. 19: 'mri.jpg' filtered through the Low-Pass Filter defined in Equation 1, corrupted with $\sigma^2 = 400$ noise (left), then optimally filtered in the Spatial Domain (window size: 11x11) (right)

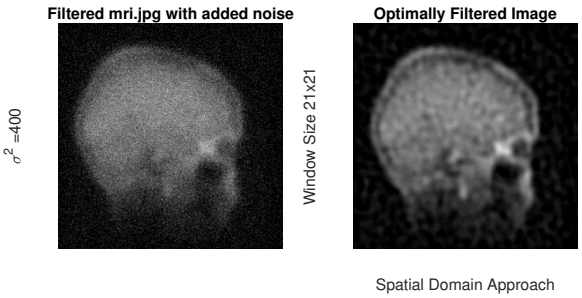

Fig. 20: 'mri.jpg' filtered through the Low-Pass Filter defined in Equation 1, corrupted with $\sigma^2 = 400$ noise (left), then optimally filtered in the Spatial Domain (window size: 21x21) (right)

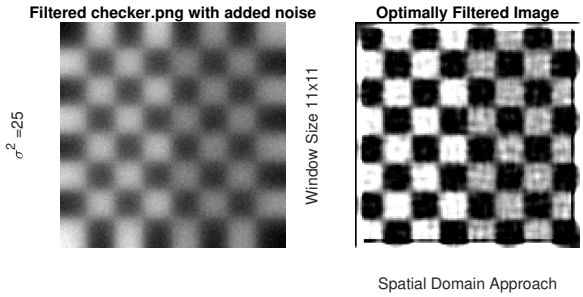

Fig. 21: 'checker.png' filtered through the Low-Pass Filter defined in Equation 1, corrupted with $\sigma^2 = 25$ noise (left), then optimally filtered in the Spatial Domain (window size: 11x11) (right)

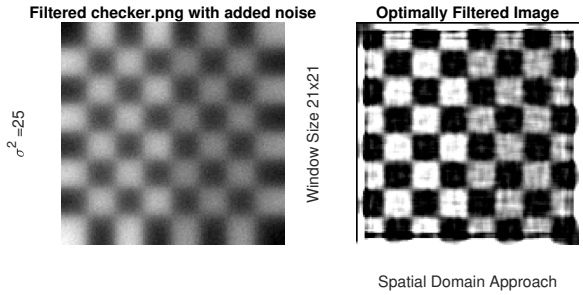

Fig. 22: 'checker.png' filtered through the Low-Pass Filter defined in Equation 1, corrupted with $\sigma^2 = 25$ noise (left), then optimally filtered in the Spatial Domain (window size: 21x21) (right)

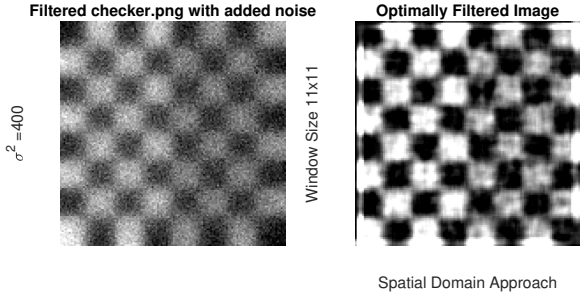

Fig. 23: 'checker.png' filtered through the Low-Pass Filter defined in Equation 1, corrupted with $\sigma^2 = 400$ noise (left), then optimally filtered in the Spatial Domain (window size: 11x11) (right)

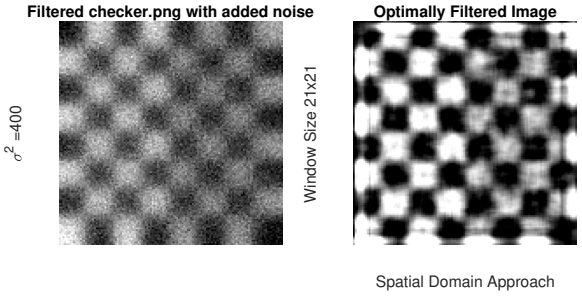

Fig. 24: 'checker.png' filtered through the Low-Pass Filter defined in Equation 1, corrupted with $\sigma^2 = 25$ noise (left), then optimally filtered in the Spatial Domain (window size: 21x21) (right)

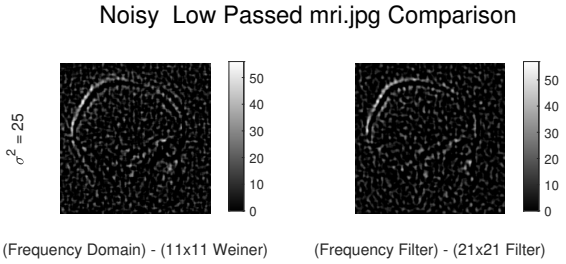

Fig. 25: Image Subtraction between 'mri.jpg' corrupted with $\sigma^2 = 25$ noise and filtered through the Low-Pass Filter defined in Equation 1, then optimally filtered through the Frequency Domain and filtered in the Spatial Domain (window size: 11x11 [left] and 21x21 [right])

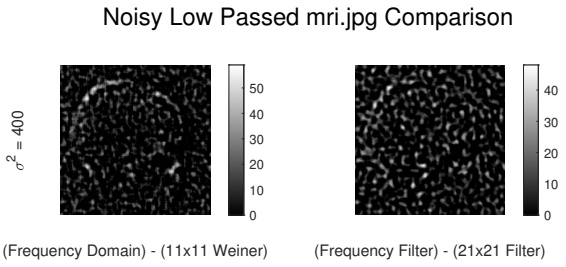

Fig. 26: Image Subtraction between 'mri.jpg' corrupted with $\sigma^2 = 400$ noise and filtered through the Low-Pass Filter defined in Equation 1, then optimally filtered through the Frequency Domain and filtered in the Spatial Domain (window size: 11x11 [left] and 21x21 [right])

APPENDIX B
MATLAB CODE

Listing 1: Main Code used for filtering images

```
%% ENGG 4660: MEDICAL IMAGE PROCESSING
% LAB 2: OPTIMAL FILTERING
% DANIEL SHERMAN
% 0954083
% FEBRUARY 3, 2020

%% CLEAN UP

close all
clear all
clc

%%  LOAD IN IMAGES

checker = imread('checker.png');
mri = imread('mri.jpg');

%% APPLY WEINER FILTERS IN THE FREQUENCY DOMAIN

mri100f = wiener_freq(mri, 100, 'mri.jpg');
mri900f = wiener_freq(mri, 900, 'mri.jpg');

%% APPLY LEAST-SQUARES FILTER IN THE SPATIAL DOMAIN

mri100_11s = least_squares(mri, 'mri.jpg', 100, 11);
mri900_11s = least_squares(mri, 'mri.jpg', 900, 11);
mri100_21s = least_squares(mri, 'mri.jpg', 100, 21);
mri900_21s = least_squares(mri, 'mri.jpg', 900, 21);

checker100_11 = least_squares(checker, 'checker.png', 100, 11);
checker900_11 = least_squares(checker, 'checker.png', 900, 11);
checker100_21 = least_squares(checker, 'checker.png', 100, 21);
checker900_21 = least_squares(checker, 'checker.png', 900, 21);

%% COMPARE IN FUCNTION

compare_images(mri100f, mri100_11s, mri100_21s, 100, [11 21], 'mri.jpg');
compare_images(mri900f, mri900_11s, mri900_21s, 900, [11 21], 'mri.jpg');

%% APPLY WIENER FILTER IN THE FREQUENCY DOMAIN TO THE LOW PASSED IMAGE

fmri25f = wiener_filt_img_freq(mri, 25, 'mri.jpg');
fmri400f = wiener_filt_img_freq(mri, 400, 'mri.jpg');

%% APPLY FILTERING IN THE SPATIAL DOMAIN AFTER LOW PASS FILTER

```

```
fmri25_11s = least_squares_spat(mri, 25, 11, 'mri.jpg');
fmri25_21s = least_squares_spat(mri, 25, 21, 'mri.jpg');
fmri400_11s = least_squares_spat(mri, 400, 11, 'mri.jpg');
fmri400_21s = least_squares_spat(mri, 400, 21, 'mri.jpg');

fchecker25_11s = least_squares_spat(checker, 25, 11, 'checker.png');
fchecker25_21s = least_squares_spat(checker, 25, 21, 'checker.png');
fchecker400_11s = least_squares_spat(checker, 400, 11, 'checker.png');
fchecker400_21s = least_squares_spat(checker, 400, 21, 'checker.png');

%% COMPARE IMAGES THAT HAVE BEEN FILTERED

compare_images(fmri25f, fmri25_11s, fmri25_21s, 25, [11 21], ' Low Passed mri.jpg');
compare_images(fmri400f, fmri400_11s, fmri400_21s, 400, [11 21], 'Low Passed mri.jpg');
```

Listing 2: Code used to define and apply Wiener filters in the frequency domain

```
function filtered_spat = wiener_filter_freq(image, variance, name)
%% DOCUMENTATION

% FUNCTION ACCEPTS AN IMAGE, VARIANCE OF NOISE, AND IMAGE NAME. FUNCTION
% ADDS NOISE AND APPLIES AN OPTIMAL WIENER FILTER IN THE FREQUENCY DOMAIN.
% FUNCTION RETURNS THE OPTIMALLY FILTERED IMAGE

% MADE BY: DANIEL SHERMAN
% JANUARY 27, 2020

%% ADD NOISE TO MRI IMAGE

noise_image = double(image) + sqrt(variance)*randn(size(image));

%% OPTIMAL LEAST-SQUARES FILTERING

P_simage = abs(fftshift(fft2(image))).^2; %find PSD of image
P_n = (row*col).* variance*ones(row, col); %find PSD of noise, based on equation from lab manual

opti_filter = P_simage./(P_simage + P_n); %define optimal filter

filtered = fftshift(fft2(noise_image)).* opti_filter; %apply optimal filter in frequency domain

filtered_spat = ifft2(fftshift(filtered)); %convert to spatial domain

%plot appropriately
figure()
subplot(1,2,1)
imshow(uint8(noise_image))
title(strcat([name ' with Noise']))
ylabel(strcat(['\sigma^2 = ', num2str(variance)]))
subplot(1,2,2)
imshow(uint8(filtered_spat))
title('Optimally Filtered Image')
xlabel('Frequency Domain Approach')
```

Listing 3: Code used to define and apply Least-Squares filter in the spatial domain

```
function filtered_image = least_squares(image, name, noise, n_window)
%% DOCUMENTATION

% FUNCTION ACCEPTS AN IMAGE, THE IMAGE NAME, THE VARIANCE OF NOISE, AND A
% FILTER WINDOW. FUCNTION ADDS NOISE TO THE IMAGE, AND APPLIES THE
% LEAST-SQUARES METHOD TO FILTERING THE IMAGE IN THE SPATIAL DOMAIN USING
% THE CROSS AND AUTOCORRELATION FUNCTIONS. FUNCTION RETURNS THE OPTIMALLY
% FILTERED IMAGE AND PLOTS APPROPRIATELY

% MADE BY: DANIEL SHERMAN
% FEBRUARY 5, 2020


%% APPLY LEAST-SQUARES FILTERING

noisy_image = double(image) + sqrt(noise)*randn(size(image)); %add noise to the image

n = fix(n_window/2); %define half window size, ensure is an integer

R_xx = zeros(n_window.^2, n_window.^2); %initialize autocorrelation function
r_sx = zeros(n_window.^2, 1); %initialize crosscorrelation function

for i = n + 1: row - n %iterate through
```

```
        the image, stopping where the window
        hits the image edge
24      for j = n + 1: col - n
25          x = double(noisy_image(i - n:i +
                n, j - n:j + n)); %take
                subsample of noisy image
26          x_reshape = reshape(x, n_window
                .^2, 1); %reshape to an
                n_window^2 x 1 vector for easy
                 processing
27          R_xx = R_xx + x_reshape*(
                x_reshape.'); %find and update
                 autocorrelation of the noisy
                subsample
28          r_sx = r_sx + x_reshape*image(i,j
                ); %find and update
                crosscorrelation of the noisy
                subsample with the clean image
29      end
30  end
31
32  filter = reshape(inv(R_xx)*r_sx, n_window
        , n_window); %define the optimal Least
        -Squares filter
33
34  filtered_image = imfilter(noisy_image,
        filter); %apply the filter
35
36  %plot appropriately
37  figure()
38  subplot(1,2,1)
39  imshow(uint8(noisy_image))
40  title(strcat([name, ' with added noise']))
    )
41  ylabel(strcat('\sigma^2 = ', num2str(
        noise)))
42  subplot(1,2,2)
43  imshow(uint8(filtered_image))
44  title('Optimally Filtered Image')
45  xlabel('Spatial Domain Approach')
46  ylabel(strcat([num2str(n_window), 'x',
        num2str(n_window), ' Window']))
```

Listing 4: Code used to filter an image with the custom Low-Pass filter and to define and apply Wiener filters in the frequency domain

```
1  function spatial_image =
       wiener_filt_img_freq(image, noise,
       name)
2  %% DOCUMENTATION
3
4  % FUNCTION ACCPETS AN IMAGE, VARIANCE OF
       NOISE, AND IMAGE NAME.  FUNCTION
5  % APPLIES A CUSTOM LOW-PASS FILTER, ADDS
       NOISE, AND APPLIES AN OPTIMAL
6  % WIENER FILTER IN THE FREQUENCY DOMAIN.
        FUNCTION RETURNS THE OPTIMALLY
7  % FILTERED IMAGE
8
9  % MADE BY: DANIEL SHERMAN
10 % JANUARY 27, 2020
11
12
13 %% START OF CODE
14
15 % define custom low pass filter size for
       the image
16 h0 = 0.8.^(abs(-128:127));
17 k = 1/sum(h0);
18 h = k.*h0;
19 filt_mag = abs(fftshift(fft2((h.')*h)))
       .^2; %determine magnitude of the
       filter
20 H = fftshift(fft2((h.')*h)); %find the
       fourier transform of the filter matrix
21
22 P_s = abs(fftshift(fft2(image))).^2; %
       find the PSD of the clean image
23
24 %apply custom low-pass filter and add
       noise
25 clow_nimage = imfilter(imfilter(double(
       image), h, 'replicate'), h.', '
       replicate') + sqrt(noise)*randn(size(
       image));
26
27 %find size of the image and find the PSD
       of the noise
28 [row, col] = size(image);
29 P_n = (row*col).* noise*ones(row, col);
30
31 %define optimal filter
32 opti_filter = H.*P_s./(filt_mag.*P_s +
       P_n);
33
34 %apply filter in the frequency domain
35 filtered_nimage = fftshift(fft2(
       clow_nimage)).* opti_filter;
36
37 %convert back to spatial domain
38 spatial_image = fftshift(ifft2(fftshift(
       filtered_nimage)));
39
40 %plot appropriately
41 figure
42 subplot(1,2,1)
43 imshow(uint8(clow_nimage))
44 ylabel(strcat(['\sigma^2 = ', num2str(
       noise)]))
45 title(strcat(['Low Passed ', name, ' with
        Noise']))
46 subplot(1,2,2)
47 imshow(uint8(spatial_image))
48 title('Optimally Filtered Image')
49 xlabel('Frequency Domain Approach')
```

Listing 5: Code used to filter an image with the custom Low-Pass filter and to define and apply Least-Squares filter in the spatial domain

```
function filtered_image = least_squares_spat(image, noise, n_window, name)
%% DOCUMENTATION

% FUNCTION ACCEPTS AN IMAGE, VARIANCE OF NOISE, WINDOW SIZE, AND IMAGE
% NAME. FUNCTION DEFINES AND APPLIES CUSTOM LOW-PASS FILTER, THEN ADDS
% NOISE TO THE IMAGE. FUNCTION APPLIES A SPATIAL DOMAIN APPROACH
% LEAST-SQUARES OPTIMAL FILTER TO FILTER NOISY FILTERED IMAGE. FUNCTION
% RETURNS THE OPTIMALLY FILTERED IMAGE

%MADE BY: DANIEL SHERMAN
% FEBRUARY 11, 2020


%% APPLY DECONVOLUTION IN SPATIAL DOMAIN WITH A FILTERED IMAGE

[row, col] = size(image); %find image size

% define custom low pass filter size for the image
h0 = 0.8.^(abs(-row/2:row/2 - 1));
k = 1/sum(h0);
h = k.*h0;
H = fftshift(fft2((h.')*h));

%apply filter and add noise
clow_nimage = imfilter(imfilter(double(image), h, 'replicate'), h.', 'replicate') + sqrt(noise)*randn(size(image));

n = fix(n_window/2); %define half window size, ensure is an integer

R_xx = zeros(n_window.^2, n_window.^2); % initialize autocorrelation function
r_sx = zeros(n_window.^2, 1); %initialize crosscorrelation function

for i = n + 1: row - n %iterate through the image, stopping where the window hits the image edge
    for j = n + 1: col - n
        x = double(clow_nimage(i - n:i + n, j - n:j + n)); %take subsample of noisy image
        x_reshape = reshape(x, n_window.^2, 1); %reshape to an n_window^2 x 1 vector for easy processing
        R_xx = R_xx + x_reshape*(x_reshape.'); %find and update autocorrelation of the noisy subsample
        r_sx = r_sx + x_reshape*double(image(i,j)); %find and update crosscorrelation of the noisy subsample with the clean image
    end
end

filter = reshape(inv(R_xx)*r_sx, n_window, n_window); %define the optimal Least-Squares filter

filtered_image = imfilter(clow_nimage, filter); %apply the filter

%plot appropriately
figure()
subplot(1,2,1)
imshow(uint8(clow_nimage))
title(strcat(['Filtered ' name, ' with added noise']))
ylabel(strcat('\sigma^2 = ', num2str(noise)))
subplot(1,2,2)
imshow(uint8(filtered_image))
ylabel(strcat(['Window Size ', num2str(n_window), 'x', num2str(n_window)]))
xlabel('Spatial Domain Approach')
title('Optimally Filtered Image')
```

Listing 6: Code used to apply image subtraction for comparison purposes

```
function compare_images(frequencyD, spatialD11, spatialD21, noise, window, name)

%% DOCUMENTATION

% FUNCTION ACCEPTS 3 IMAGES: 1 FILTERED IN THE FREQUENCY DOMAIN,
% AND 2 FILTERED IN THE SPATIAL DOMAIN. ALSO ACCEPTS VARIANCE OF NOISE,
% FILTERING WINDOW (IN VECTOR) AND THE IMAGE NAME.
% FUNCTION DOES A SUBTRACTION OF FREQUENCY DOMAIN IMAGE
% AND THE SPATIAL DOMAIN AND PLOTS THE RESULTS NICELY
%
% MADE BY: DANIEL SHERMAN
% FEBRUARY 10, 2020

```

```
%% START OF CODE

%perform image subtraction: frequency domain - spatial domain
comp100_11 = frequencyD - spatialD11;
comp100_21 = frequencyD - spatialD21;

%plot results
figure()
subplot(1,2,1)
imshow(uint8(comp100_11), [])
xlabel(strcat(['(Frequency Domain) - (', num2str(window(1)),'x', num2str(window(1)), ' Weiner)']))
ylabel(strcat(['\sigma^2 = ' num2str(noise)]))
colorbar
subplot(1,2,2)
imshow(uint8(comp100_21), [])
xlabel(strcat(['(Frequency Filter) - (', num2str(window(2)),'x', num2str(window(2)), ' Filter)']))
colorbar
sgtitle(strcat(['Noisy ' name ' Comparison']))
```

REFERENCES

[1] E. Sekko, G. Thomas, and A. Bourkrouche, *A Deconvolution Technique Using Optimal Wiener Filtering and Regularization*, 72nd vol., pp. 23-32, Signal Processing, 1999.

[2] O.B. Pogrebnyak and V.V. Lukin, *Wiener Discrete Cosine Transform-Based Image Filtering*, 21st vol., Journal of Electronic Imaging, 2012. https://doi.org/10.1117/1.JEI.21.4.043020