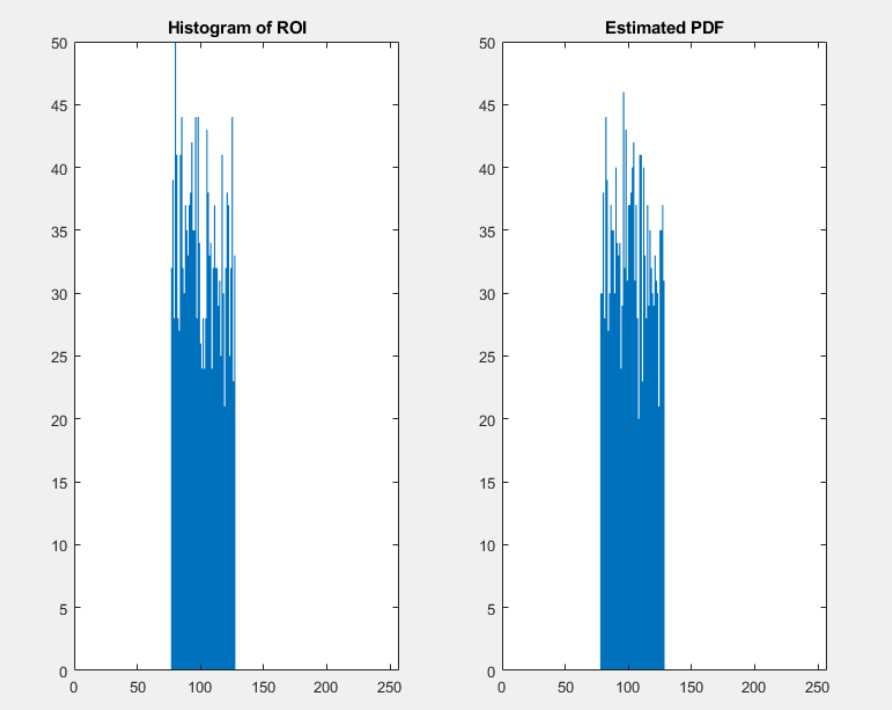
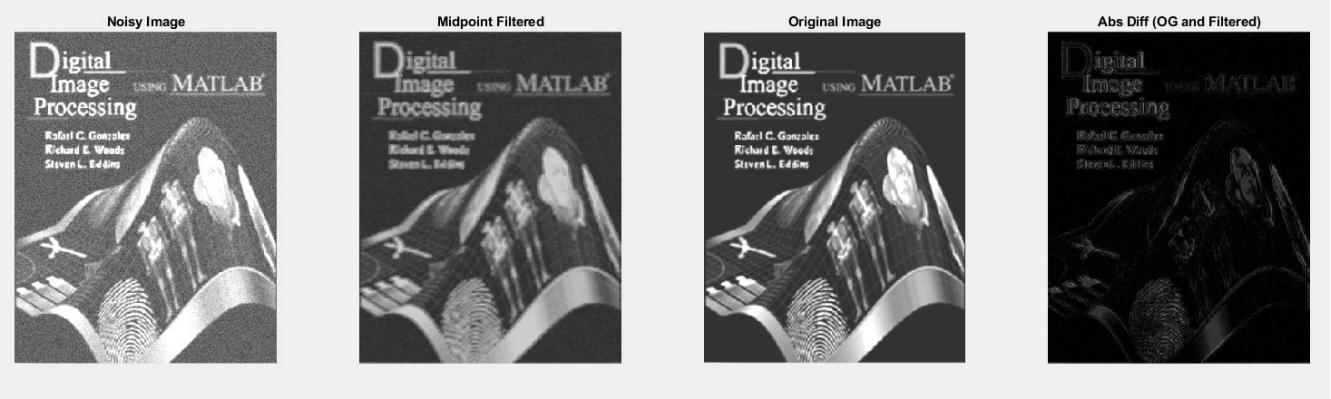
Ryan English
Homework #2
09/27/2021

1) File “DIP.png” contains the original clean image. Files “DIP_n1.png”, “DIP_n2.png”, “DIP_n3.png”, “DIP_n4.png”, and “DIP_n5.png” contain the same image corrupted by different types of noise. For each noisy image
    a. Estimate the PDF of the noise (including parameters of the PDF). For the case of periodic noise, estimate the frequencies of the noise.
    b. Restore the image using the most appropriate filter. Justify your choice. You need to experiment with different filters with different sizes.
    c. For each case, compare the restored image with the original clean image.

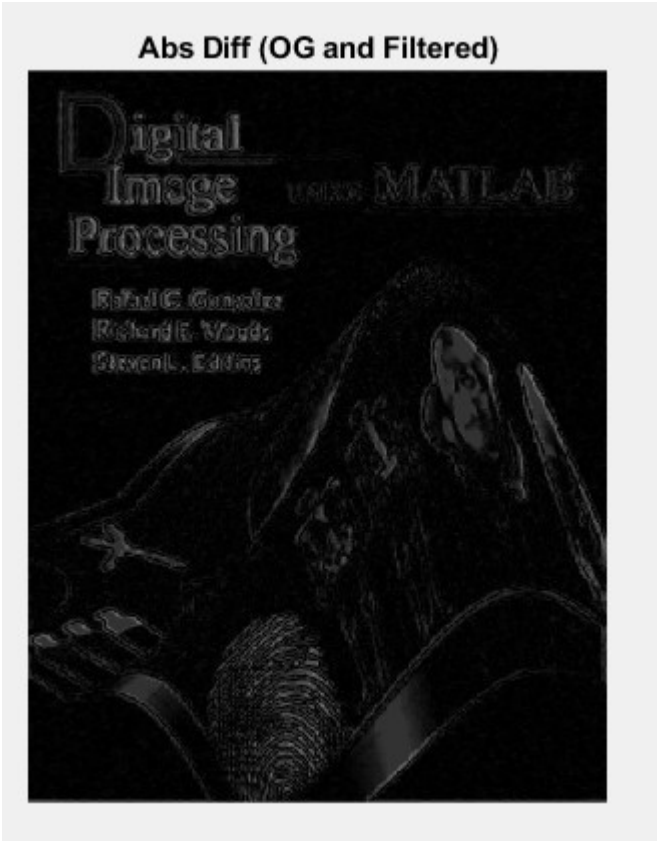
DIP_1.png



PDF = uniform a =.3 b = .5

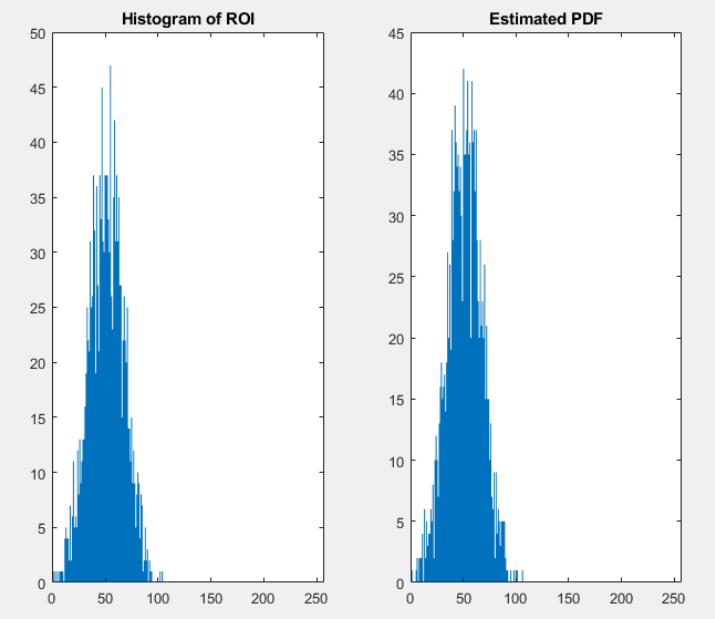


For restoring DIP_1 I used the midpoint [3 3] filter as well as darkening the intensity by 50. The midpoint filter was chosen because it takes the half way point between the max and the min filters, and with the “white noise” and the black image it seemed that taking the midpoint between would be the best option to restore the image. However, this did lighten the image, since the noise was brighter, so to counter act that I added to the restored images intensity.

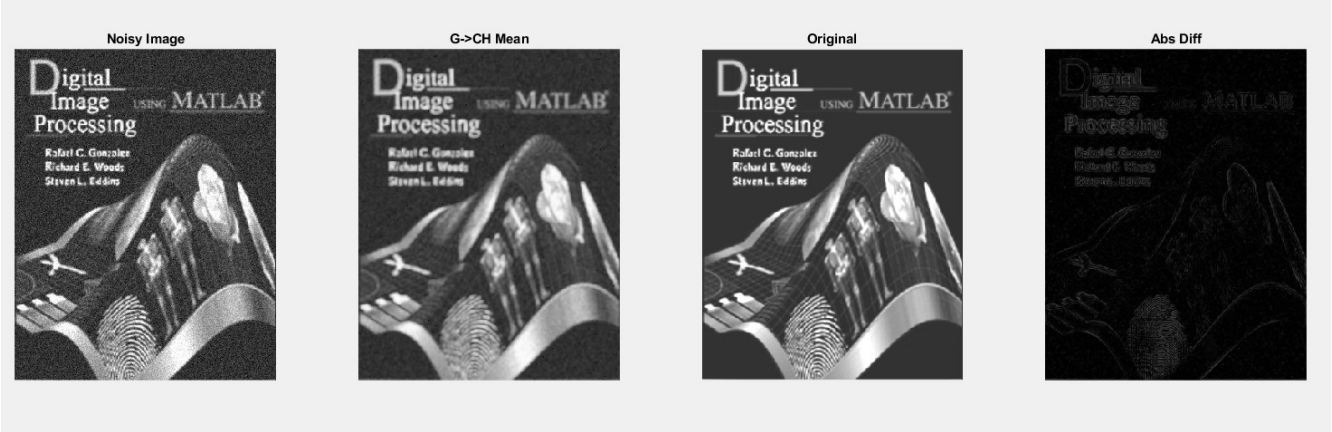


When comparing the denoised image with the original image, we can see that there is still a pretty big difference between the images; but, this is mainly due to the change in intensity as there is no extra noise between the two images.

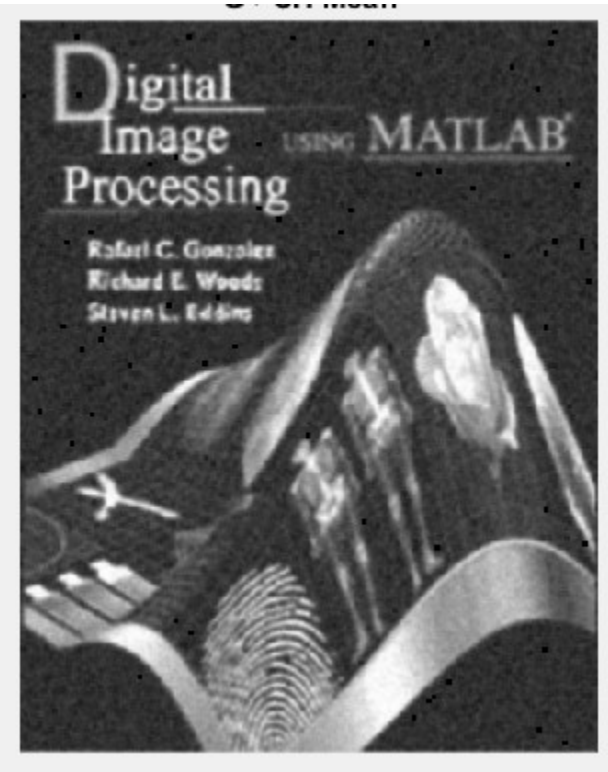
DIP_2.png



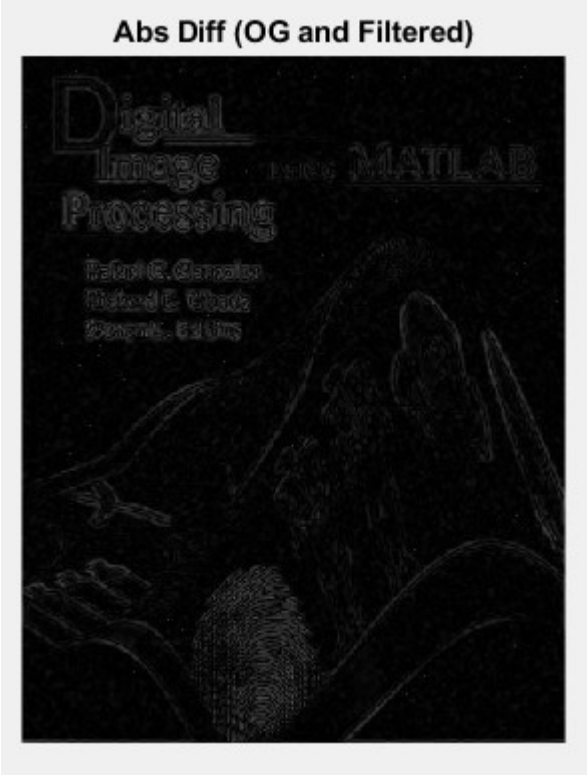
PDF = Gaussian a = 2nd stat moment's mean (.197 pictured above) b = square root of the 2nd stat moment variance (.064 pictured above). Used Example 5.4 p 266 to determine.



For DIP_2 I used the geometric [3 3] mean filter and then the contraharmonic [3 3] Q=3 mean to return the noise image to its original state. I denoised like this because the geometric mean is well suited for Gaussian noise, but when I used the geometric mean by itself I ended up with a peppery image, see below:

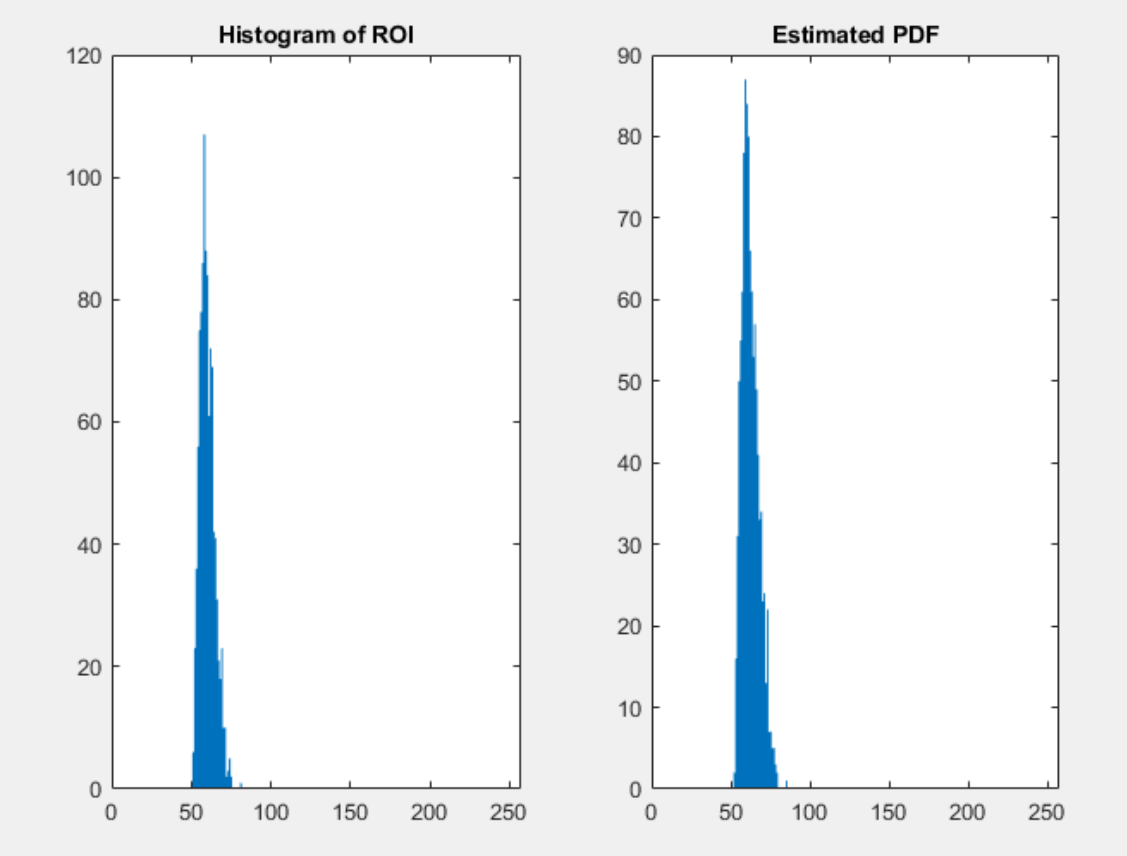


And the contraharmonic is suited well for salt & pepper noise, so I denoised the filtered image with contraharmonic filter with Q=3 in order to get those pepper spots.



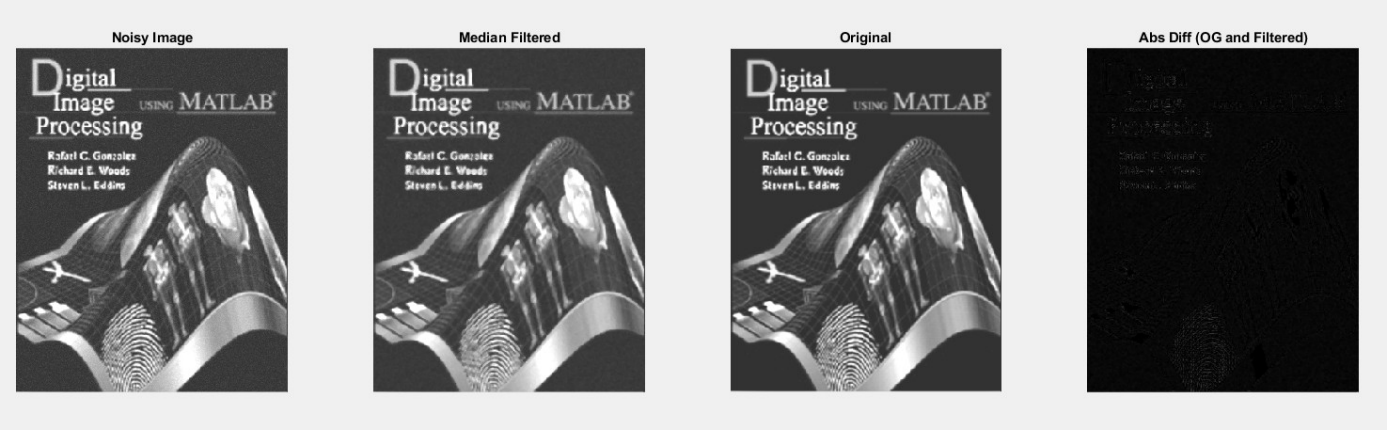
In the comparison of the denoised image and the original image, you can see there is still a bit of noise left in the denoised image, but it is relatively close to the original image. The mean filters also blur the image a bit when using them, so the edges are in the comparison.

DIP_3.png



PDF = rayleigh a = .2 b = .002 (This one took A LOT of trial and error, but finally found a PDF that is relatively close to the ROI hist)

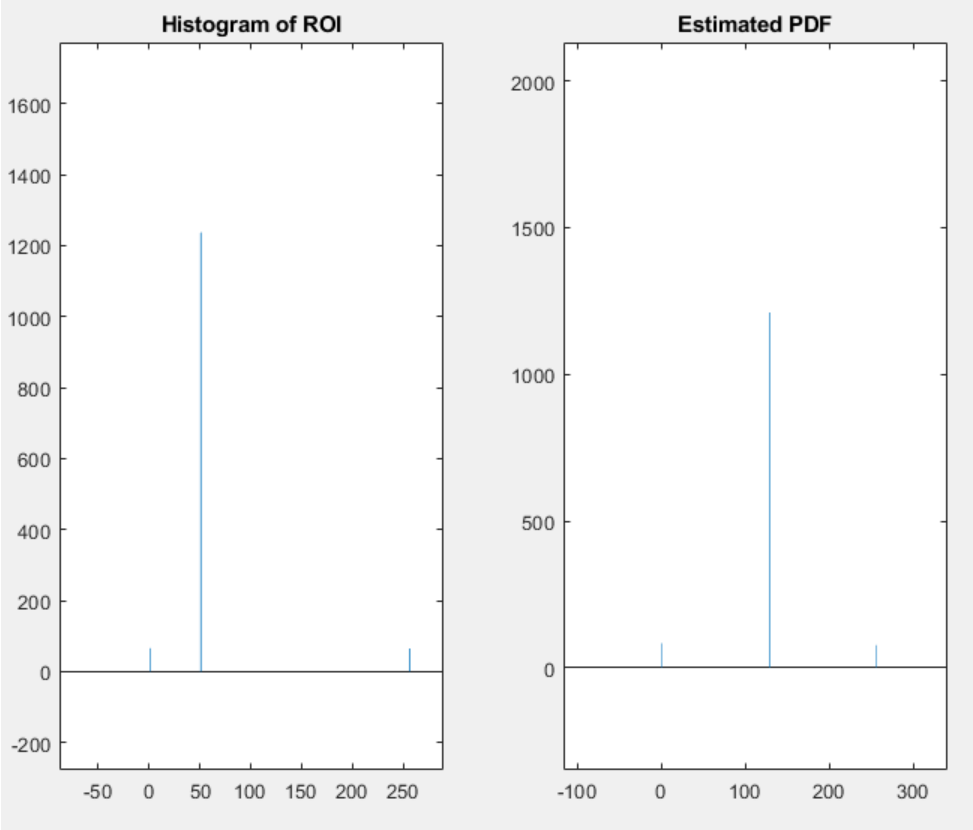
For DIP_3 I used the median [3 3] filter because it is good for static noise without the blurred affect of the average filters.



You can see that the median filter almost returned the image to its original state, with the exception of a bit of the header and the fingerprint:

The median takes the median of g(s,t) to produce the denoised image, and given the noise is just a “static” noise it reduces the noise because the static affect would not be overpowering in the [3 3] filter, thus the pixel will be slowly returned to its original state (or blurred after multiple passes).

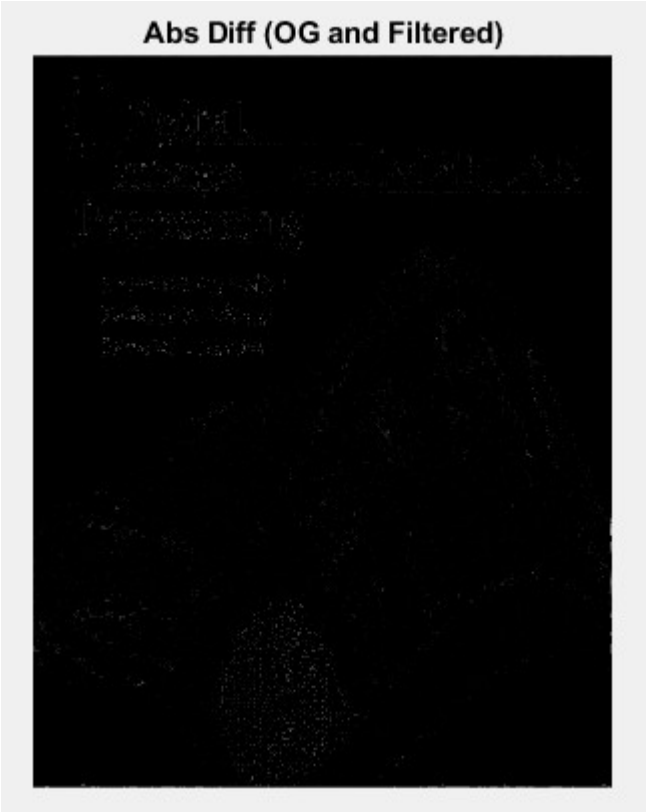
DIP_4.png



PDF = salt & pepper a= .05 and b = .05

The PDF is not exact, however it is relatively close to the values. The middle bar is vastly different because on the random imnoise it is generating 0.5 if neither salt nor pepper, so it gets 128; whereas, in the actual image it is half of the POI intensity scale.

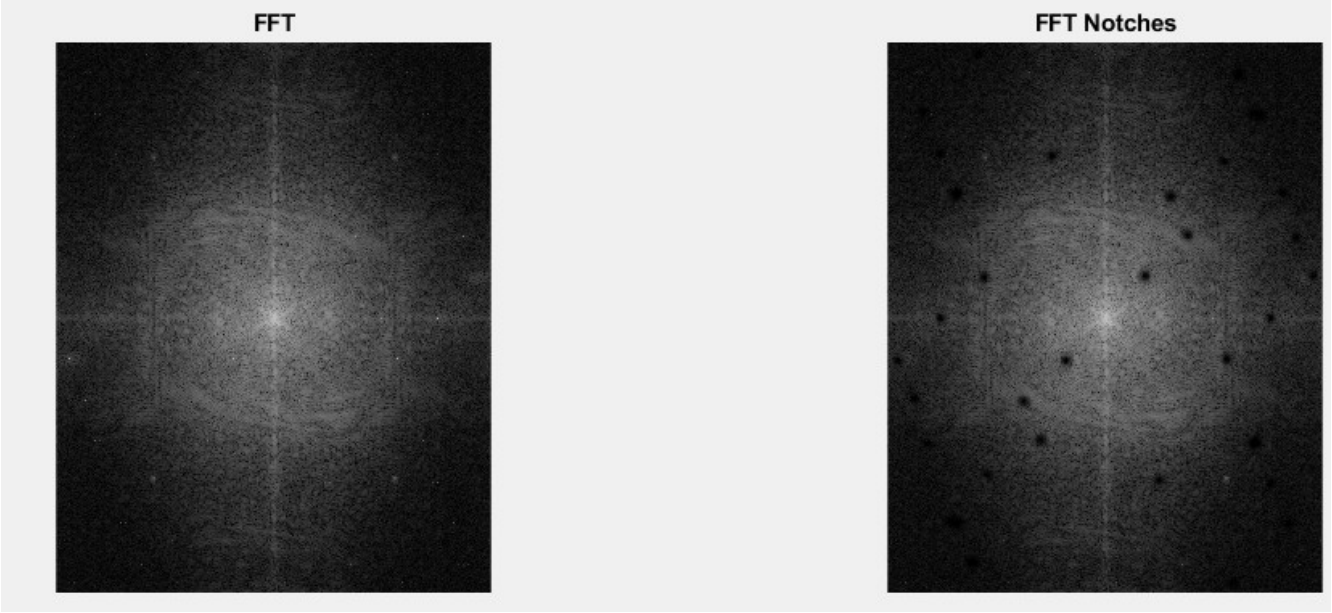
For DIP_4 I used the adaptive mean Smax = 7. I chose the adaptive mean because the median filter is a good choice for the salt & pepper noise, however, the adaptive mean generates better results in a better denoised image if the salt/pepper does not have the same value as the POI background. Considering the salt and pepper are 0/256 respectively and the background in the POI was 104, it was worth an attempt. As you can see there is hardly a difference between the original image and the filtered image:



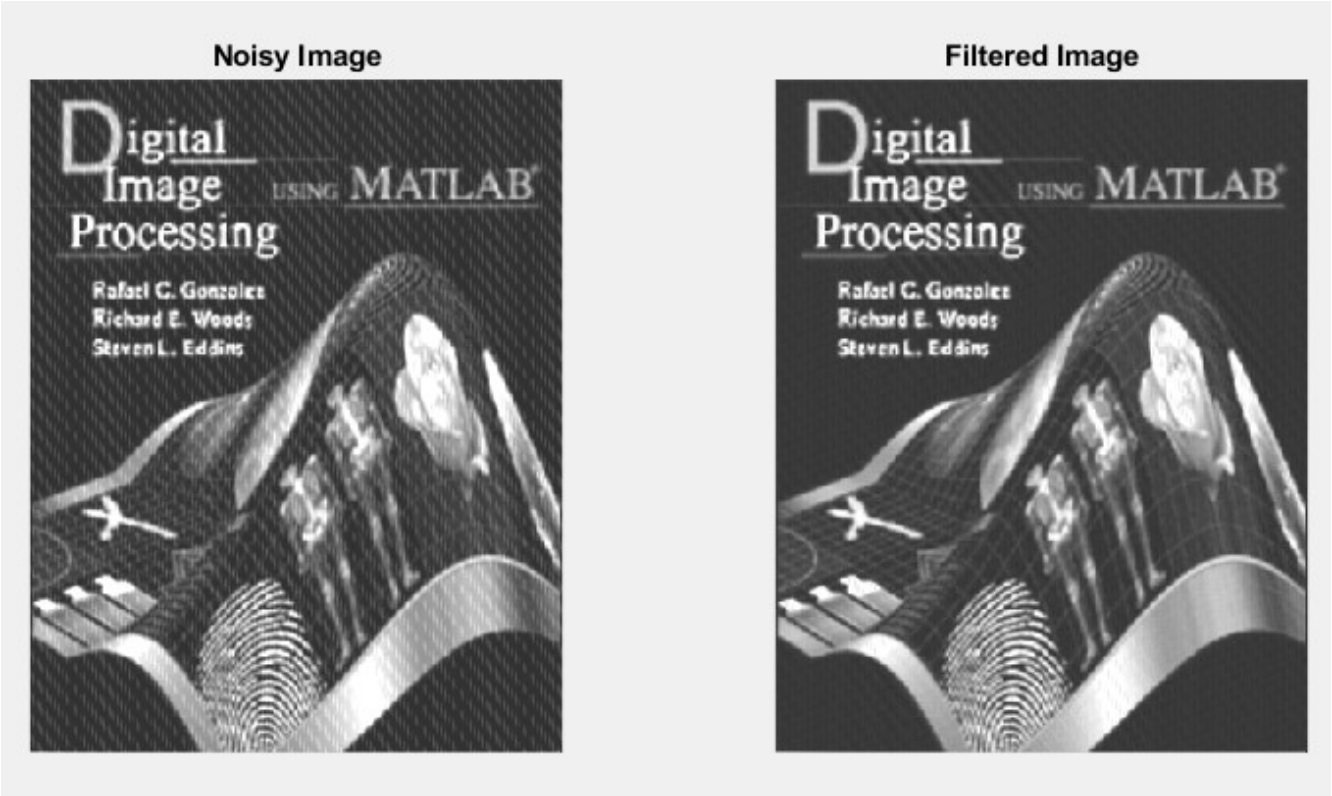
DIP_5.png

DIP_5 has noise in the frequency domain. So utilizing a notch guassian filter at the specific spikes in the frequency domain then rejecting these spots we get a denoised image.

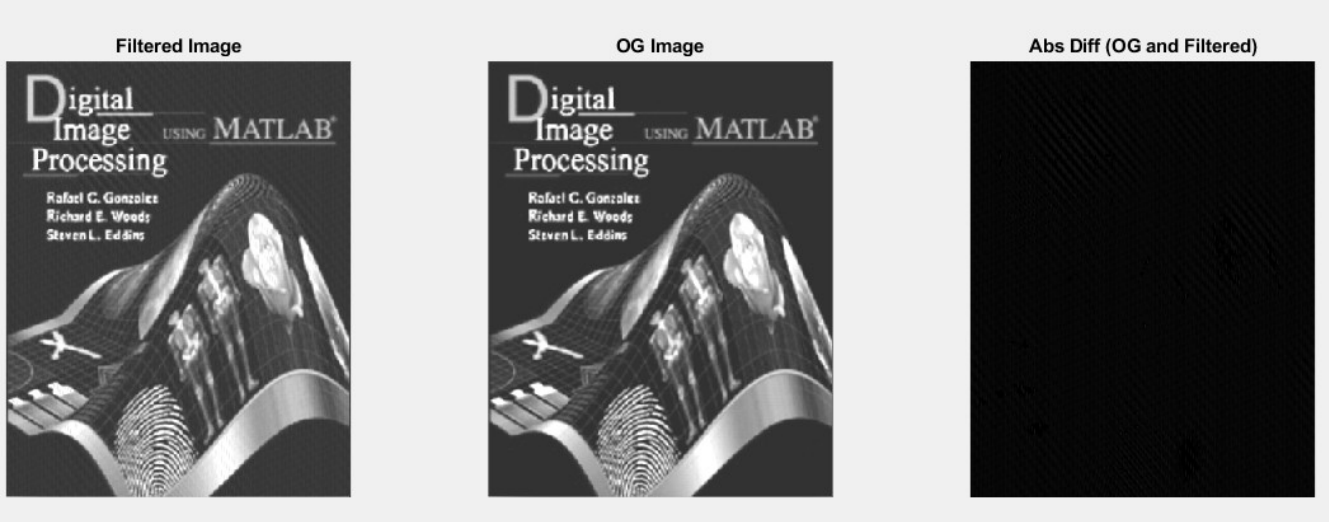
In order to accomplish this I used *impixelinfo* to capture all the spots in the frequency domain:



All of the spikes in frequency are captured by the “notch spots” (D0=3) and rejected with a filter in the frequency domain. Below you can see the noise and filtered image.



And then if we compare the filtered image with the original image:



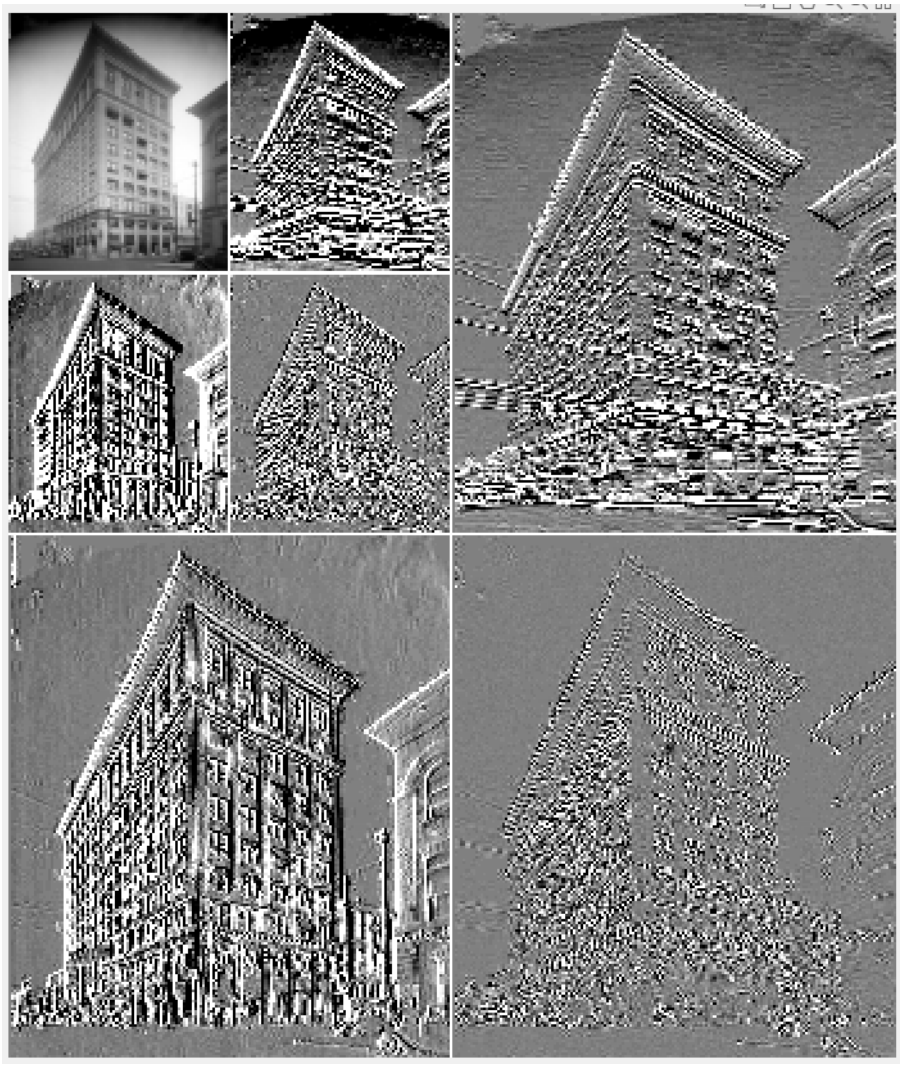
You can see that only a bit of noise is left with almost the entire image returned.

2) **Select one (real) image of your choice that is a good candidate to illustrate the concept of multi-resolution analysis. Using a 2-scale wavelet decomposition (Haar), analyze and compare**
    a. **The vertical edges at scale 1 only, scale 2 only, and both scales.**
    b. **The horizontal edges at scale 1 only, scale 2 only, and both scales.**
    c. **The diagonal edges at scale 1 only, scale 2 only, and both scales.**

To illustrate the multi-resolution analysis I tried to find an image of a building with good hard edges, without lots of noise like bushes at the bottom, so that edge detail of the analysis would be very easy to decipher. The best I could find was:

Unfortunately, the telephone poles and the jaded roof do not make for a perfect picture; but it works nonetheless.

Depiction of to 2 layer Haar:

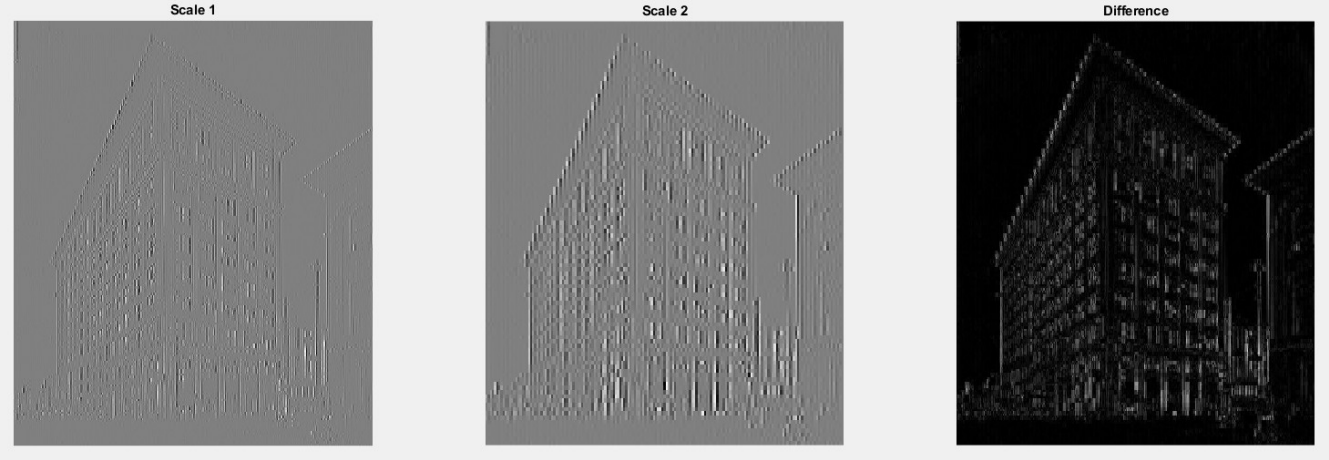


As I mentioned above the horizontal/diagonal have a bit of noise due to the telephone poles, but otherwise the horizontal is nice and clean.

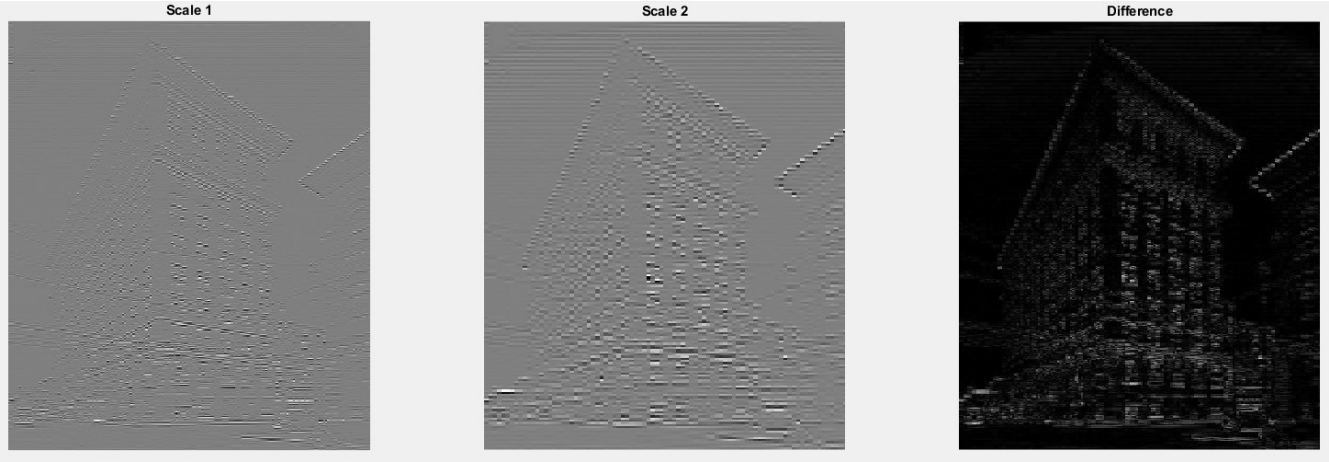
Next we want to compare the details at layer 1 to the details at layer 2 for each type of detail. In order to do this, I had to generate the 2 layer haar wavelet and for each section (h, v, d) I had to zero out all other layers, then recompose the wavelet. Thus, creating original 400x388 sized images of just the given layer.

For all of the below, taking layer 1 and layer 2 of each detail type then comparing results in the original version of said detail, since we are taking the decomposition and removing the other layers then re-compositing the image back to the original state; we would only obtain an image of the given detail. This makes for a good method of edge detection.

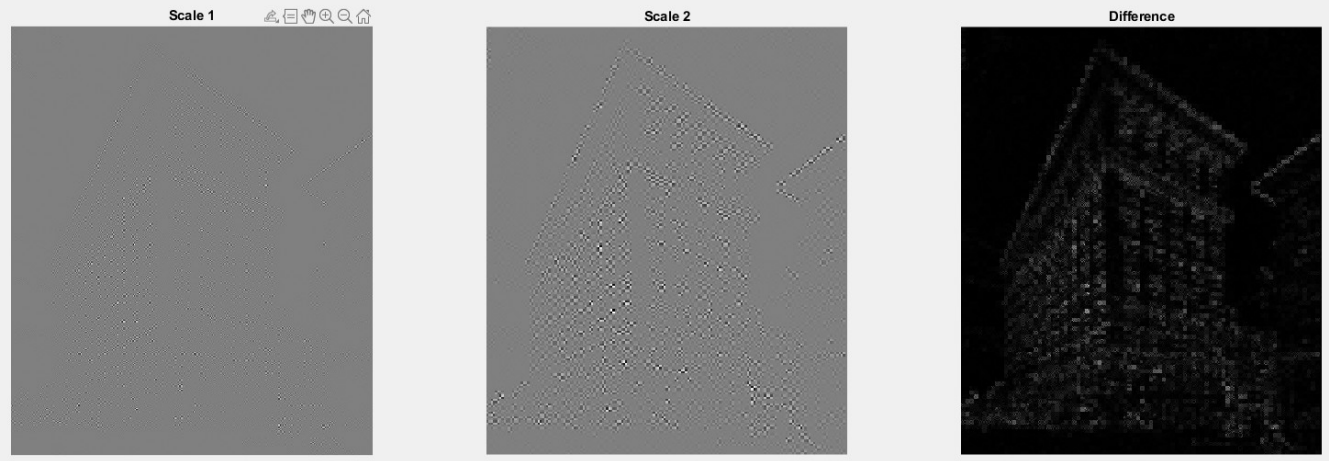
a. Vertical



b. Horizontal



c. Diagonal



** NOTE: A lot of reworks went into this with lots of copy paste between the problem sections. Comments likely don't make any sense at this point. **

Problem 1 – DIP_1

```
ogImg = imread('DIP.png');
img = imread('DIP_n1.png');

% Get a section of the image interactively
[B, c, r] = roipoly(img);

% Look at the histogram to determine the noise
[hist, npix] = histroi(img, c, r);

% Looks like uniform? Try to get the hist from imnoise
[~, noise] = imnoise2(ones(npix, 1), 'uniform', .3, .5);
hist2 = imhist(noise);

figure(2),
subplot(1, 2, 1), bar(hist, 1), title('Histogram of ROI'),
subplot(1, 2, 2), bar(hist2, 1), title('Estimated PDF');

% According to slides gmean is good for uniform
% After trying the means seems midpoint is the best
mid = spfilt(img, 'midpoint', 3, 4);
mid = mid - 50;

K = imabsdiff(mid, ogImg);

figure(3),
subplot(1, 4, 1), imshow(img), title('Noisy Image')
subplot(1, 4, 2), imshow(mid), title('Midpoint Filtered')
subplot(1, 4, 3), imshow(ogImg), title('Original Image')
subplot(1, 4, 4), imshow(K), title('Abs Diff (OG and Filtered)');
%DONE
```

Problem 1 – DIP_2

```
ogImg = imread('DIP.png');
img = imread('DIP_n2.png');

% Get a section of the image interactively
[B, c, r] = roipoly(img);

% Look at the histogram to determine the noise
[hist, npix] = histroi(img, c, r);

% Looks like gaussian? Try to get the hist from imnoise
[v, unv] = statmoments(hist, 2);
[~, noise] = imnoise2(ones(npix, 1), 'gaussian', v(1), sqrt(v(2)));
hist2 = imhist(noise);

figure(2),
subplot(1, 2, 1), bar(hist, 1), title('Histogram of ROI'),
subplot(1, 2, 2), bar(hist2, 1), title('Estimated PDF');

% According to slides gmean is good for uniform
% After trying the means seems midpoint is the best
mid = spfilt(img, 'gmean', 3, 3);
chmean = spfilt(mid, 'chmean', 3, 3, 3);
```

```
K = imabsdiff(chmean, ogImg);

figure(3),
subplot(2, 2, 1), imshow(img), title('Noisy Image')
subplot(2, 2, 2), imshow(chmean), title('G->CH Filtered')
subplot(2, 2, 3), imshow(ogImg), title('Original Image')
subplot(2, 2, 4), imshow(K), title('Abs Diff (OG and Filtered)');
%DONE
```

Problem 1 – DIP_3

```
ogImg = imread('DIP.png');
img = imread('DIP_n3.png');

% Get a section of the image interactively
[B, c, r] = roipoly(img);

% Look at the histogram to determine the noise
[hist, npix] = histroi(img, c, r);

% Looks like exp? Try to get the hist from imnoise
[~, noise] = imnoise2(ones(npix, 1), 'rayleigh', .2, .002);
hist2 = imhist(noise);

% WRONG
figure(2),
subplot(1, 2, 1), bar(hist, 1), title('Histogram of ROI'),
subplot(1, 2, 2), bar(hist2, 1), title('Estimated PDF');

% Just blurryin the noise, it is not too strong...
mid = spfilt(img, 'median', 3, 3);

K = imabsdiff(mid, ogImg);

figure(3),
subplot(1, 4, 1), imshow(img), title('Noisy Image')
subplot(1, 4, 2), imshow(mid), title('Median Filtered')
subplot(1, 4, 3), imshow(ogImg), title('Original')
subplot(1, 4, 4), imshow(K), title('Abs Diff (OG and Filtered)');
```

Problem 1 – DIP_4

```
ogImg = imread('DIP.png');
img = imread('DIP_n4.png');

% Get a section of the image interactively
[B, c, r] = roipoly(img);

% Look at the histogram to determine the noise
[hist, npix] = histroi(img, c, r);

% Looks like exp? Try to get the hist from imnoise
[~, noise] = imnoise2(ones(npix, 1), 'salt & pepper', .045, .045);
hist2 = imhist(noise);

figure(2),
subplot(1, 2, 1), bar(hist, 1), title('Histogram of ROI'),
subplot(1, 2, 2), bar(hist2, 1), title('Estimated PDF');
```

```
% According to slides gmean is good for uniform
% After trying the means seems midpoint is the best
ad = adpmedian(img, 7);

K = imabsdiff(ad, ogImg);

figure(3),
subplot(1, 4, 1), imshow(img), title('Noisy Image')
subplot(1, 4, 2), imshow(ad), title('Adaptive Median')
subplot(1, 4, 3), imshow(ogImg), title('Original')
subplot(1, 4, 4), imshow(K), title('Abs Diff (OG and Filtered)');
%DONE
```

Problem 1 – DIP_5

```
ogImg = imread('DIP.png');
img = imread('DIP_n5.png');

f = tofloat(img);
F = fft2(f);
freqimg = fftshift(log(1 + abs(F)));

notchSpots = [
    183 75;
    248 8;
    313 31;
    117 54;
    54 28;
    248 139;
    280 106;
    311 287;
    215 299;
    152 319;
    344 299;
    56 291;
    24 275;
    375 54;
    405 67;
    422 269;
    342 212;
    310 119;
    338 77;
    74 15;
    ];

[M, N] = size(img);
F = cnotch('gaussian', 'reject', M, N, notchSpots, 3);
S = intensityScaling(fftshift(F).*(freqimg));

filtered = dftfilt(img, F);
filtered = im2uint8(filtered);

K = imabsdiff(filtered, ogImg);

freq = [
    0 100;
    30 70;
    ];
```

```
[rb, ~, noise] = imnoise3(M, N, freq);
noise = imdilate(noise, ones(3));

figure(2),
subplot(2, 4, 1), imshow(freqimg, []), title('FFT')
subplot(2, 4, 2), imshow(S), title('FFT Notches')
subplot(2, 4, 3), imshow(noise), title('FFT Notches')
subplot(2, 4, 5), imshow(img), title('Noisy Image')
subplot(2, 4, 6), imshow(filtered), title('Filtered Image')
subplot(2, 4, 7), imshow(ogImg), title('OG Image')
subplot(2, 4, 8), imshow(K), title('Abs Diff (OG and Filtered)')
impixelinfo;
```

Problem 2 – Diagonal

```
img = imread('R.jpg');
img = rgb2gray(img);

[LoD, HiD, LoR, HiR] = wfilters('haar');
[c, s] = wavedec2(img, 2, LoD, HiD);

figure(1);
wavedisplay(c, s, 20);

[nc,~] = wavecut('a', c, s, 1);
[nc,~] = wavecut('v', nc, s, 1);
[nc,~] = wavecut('h', nc, s, 1);
[nc,~] = wavecut('a', nc, s, 2);
[nc,~] = wavecut('h', nc, s, 2);
[nc,~] = wavecut('d', nc, s, 2);
[nc,~] = wavecut('v', nc, s, 2);
i = waveback(nc, s, 'haar');
ddet = im2uint8(mat2gray(i));

% Reconstruct this by zeroing everything but it and reconstruct it
[nc,~] = wavecut('a', c, s, 1);
[nc,~] = wavecut('v', nc, s, 1);
[nc,~] = wavecut('h', nc, s, 1);
[nc,~] = wavecut('d', nc, s, 1);
[nc,~] = wavecut('a', nc, s, 2);
[nc,~] = wavecut('v', nc, s, 2);
[nc,~] = wavecut('h', nc, s, 2);
i = waveback(nc, s, 'haar');
ddet2 = im2uint8(mat2gray(i));

K = imabsdiff(ddet,ddet2);

figure(2),
subplot(1, 3, 1), imshow(ddet), title('Scale 1')
subplot(1, 3, 2), imshow(ddet2), title('Scale 2')
subplot(1, 3, 3), imshow(K), title('Difference');
```

Problem 2 – Vertical

```
img = imread('R.jpg');
img = rgb2gray(img);
```

```
[LoD, HiD, LoR, HiR] = wfilters('haar');
[c, s] = wavedec2(img, 2, LoD, HiD);

figure(1);
wavedisplay(c, s, 20);

[nc,~] = wavecut('a', c, s, 1);
[nc,~] = wavecut('h', nc, s, 1);
[nc,~] = wavecut('d', nc, s, 1);
[nc,~] = wavecut('a', nc, s, 2);
[nc,~] = wavecut('h', nc, s, 2);
[nc,~] = wavecut('d', nc, s, 2);
[nc,~] = wavecut('v', nc, s, 2);
i = waveback(nc, s, 'haar');
hdet = im2uint8(mat2gray(i));

% Reconstruct this by zeroing everything but it and reconstruct it
[nc,~] = wavecut('a', c, s, 1);
[nc,~] = wavecut('v', nc, s, 1);
[nc,~] = wavecut('h', nc, s, 1);
[nc,~] = wavecut('d', nc, s, 1);
[nc,~] = wavecut('a', nc, s, 2);
[nc,~] = wavecut('h', nc, s, 2);
[nc,~] = wavecut('d', nc, s, 2);
i = waveback(nc, s, 'haar');
hdet2 = im2uint8(mat2gray(i));

K = imabsdiff(hdet,hdet2);

figure(2),
subplot(1, 3, 1), imshow(hdet), title('Scale 1')
subplot(1, 3, 2), imshow(hdet2), title('Scale 2')
subplot(1, 3, 3), imshow(K), title('Difference');
```

Problem 2 – Horizontal

```
img = imread('R.jpg');
img = rgb2gray(img);

[LoD, HiD, LoR, HiR] = wfilters('haar');
[c, s] = wavedec2(img, 2, LoD, HiD);

figure(1);
wavedisplay(c, s, 20);

[nc,~] = wavecut('a', c, s, 1);
[nc,~] = wavecut('v', nc, s, 1);
[nc,~] = wavecut('d', nc, s, 1);
[nc,~] = wavecut('a', nc, s, 2);
[nc,~] = wavecut('h', nc, s, 2);
[nc,~] = wavecut('d', nc, s, 2);
[nc,~] = wavecut('v', nc, s, 2);
i = waveback(nc, s, 'haar');
vdet = im2uint8(mat2gray(i));

% Reconstruct this by zeroing everything but it and reconstruct it
[nc,~] = wavecut('a', c, s, 1);
[nc,~] = wavecut('v', nc, s, 1);
```

```
[nc,~] = wavecut('h', nc, s, 1);
[nc,~] = wavecut('d', nc, s, 1);
[nc,~] = wavecut('a', nc, s, 2);
[nc,~] = wavecut('v', nc, s, 2);
[nc,~] = wavecut('d', nc, s, 2);
i = waveback(nc, s, 'haar');
vdet2 = im2uint8(mat2gray(i));

K = imabsdiff(vdet,vdet2);

figure(2),
subplot(1, 3, 1), imshow(vdet), title('Scale 1')
subplot(1, 3, 2), imshow(vdet2), title('Scale 2')
subplot(1, 3, 3), imshow(K), title('Difference');
```