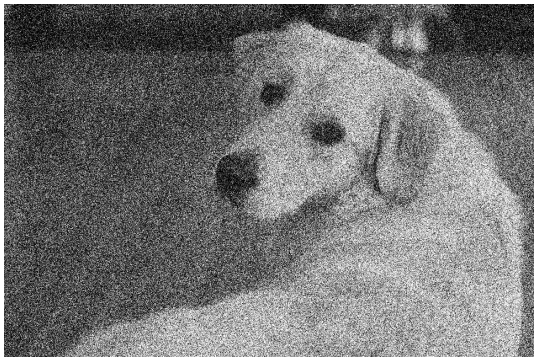
Lecture 9
Linear Filtering for Image Denoising

Jason J. Bramburger

Our goal for this lecture will be to remove noise from a noisy image. Unsurprisingly, the tool that we are going to use is filtering. We can start with the noisy dog from last lecture.

```
I = imread('sherlock.jpg');
I = rgb2gray(I);
I = im2double(I);
In = imnoise(I,'gaussian',0,0.1);

figure(1)
imshow(I)

figure(2)
imshow(In)
```



Now let's take the FFT of the noisy image. When plotting we won't worry about the frequency axes because images don't really have a length so we don't know what to scale by. We still want to use fftshift() though, so the low frequency components show up in the middle.

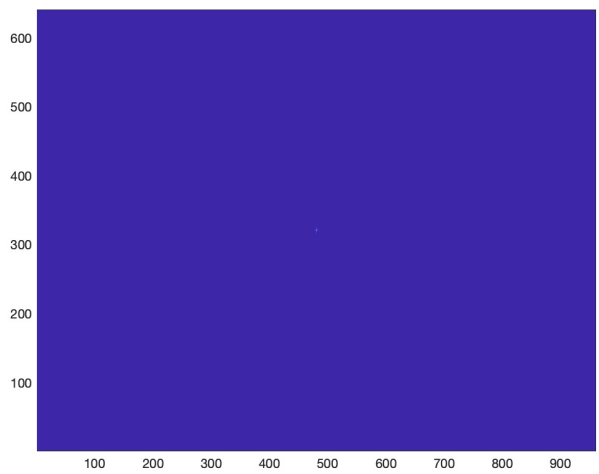
```
Int = fft2(In);
figure(3)
pcolor(fftshift(abs(Int)))
shading interp
```

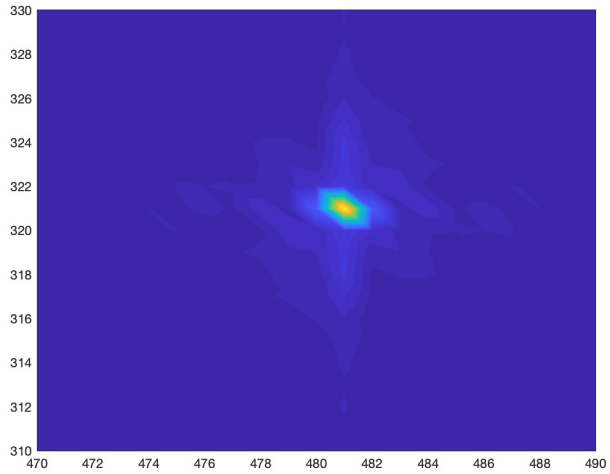
We can barely see anything in this image. That is because most of the coefficients in the Fourier transform are zero. We can zoom in a bit to see more structure (image is shown on next page).

```
axis([470 490 310 330])
```

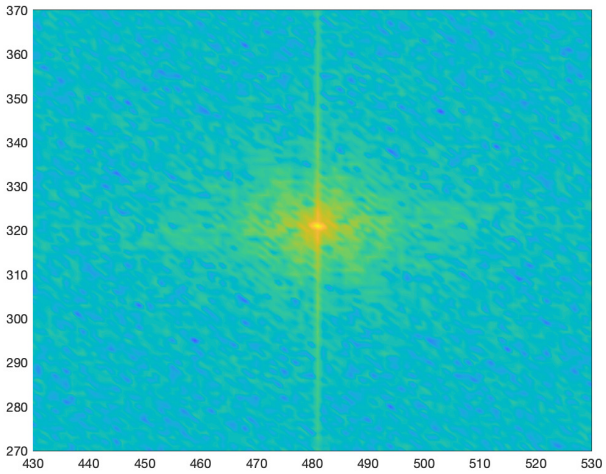
It is quite common for images to have much more low frequency content than high frequency content. Therefore, it is often useful to plot on a log scale in order to see things of different magnitudes.

```
figure(3)
pcolor(log(fftshift(abs(Int))))
shading interp
```






```
4 axis([430 530 270 370])
```



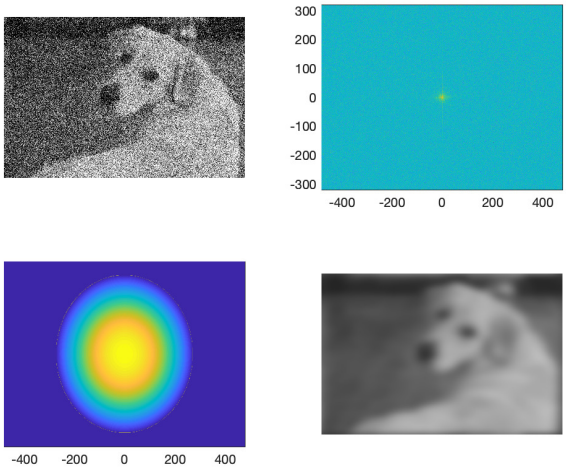
There is still a lot of zero, but we are doing better. Notice there are some strong vertical and horizontal lines. These correspond to horizontal and vertical lines in the image.

Now let's do some filtering to remove some of the noise. We can see that there is a lot of low-frequency content, so we can use a low-pass filter. Again we will use a Gaussian filter. As mentioned above, there is no spatial dimension, just pixel numbers. Therefore, we can define our frequencies without any scaling. We can define vectors for the x and y directions and then use meshgrids.

```
1 [Ny,Nx] = size(In);
2 kx = [0:Nx/2-1 -Nx/2:-1];
3 ky = [0:Ny/2-1 -Ny/2:-1];
4 [Kx,Ky] = meshgrid(kx,ky);
5
6 % Filter function
7 a = 1e-2;
8 filter = exp(-a*(Kx.^2 + Ky.^2));
```

When applying the filter, we need either the FFT and the filter to be fftshifted or neither. In this case, we didn't shift the FFT (except in the plot command) so we can apply the filter as is. We use ifft2 to recover the filtered image.

```
Intf = Int.*filter;
Inf = ifft2(Intf);

figure(4)
subplot(2,2,1)
imshow(In)
subplot(2,2,2)
pcolor(fftshift(Kx),fftshift(Ky),log(fftshift(abs(Int))))
shading interp
subplot(2,2,3)
pcolor(fftshift(Kx),fftshift(Ky),log(fftshift(abs(filter))))
shading interp
subplot(2,2,4)
imshow(Inf)
```

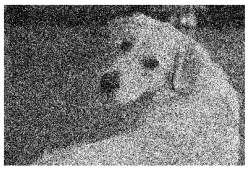
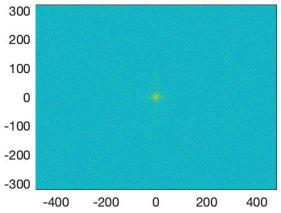


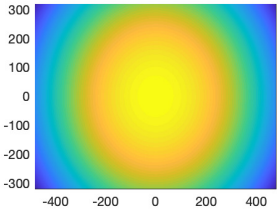
The noise appears to be gone, but now the image is blurry. That is because we removed high-frequency content which is what allows for fine details. We over-filtered. Let's try to widen the filter in order to keep more of the detail and hopefully get a better approximation of the original image.

```
a = 1e-3;
filter = exp(-a*(Kx.^2 + Ky.^2));

Intf = Int.*filter;
Inf = ifft2(Intf);

figure(5)
subplot(2,2,1)
imshow(In)
subplot(2,2,2)
pcolor(fftshift(Kx),fftshift(Ky),log(fftshift(abs(Int))))
shading interp
subplot(2,2,3)
pcolor(fftshift(Kx),fftshift(Ky),log(fftshift(abs(filter))))
shading interp
subplot(2,2,4)
imshow(Inf)
```

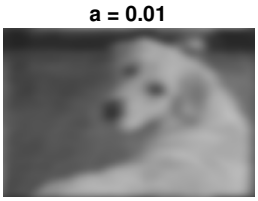
That's much better! Let's try a bunch of different filter widths to compare:

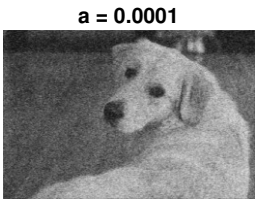
```
% Different filtering widths
a = [1e-2 1e-3 1e-4 1e-5 1e-6 1e-7];

```

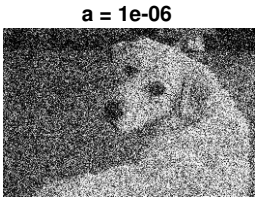



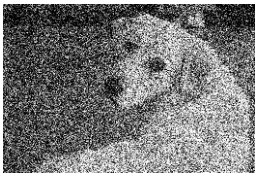


```
% Plot to compare filter widths
figure(6)
for j = 1:length(a)
    filter = exp(-a(j)*(Kx.^2 + Ky.^2));

    Intf = Int.*filter;
    Inf = ifft2(Intf);

    subplot(3,2,j)
    imshow(Inf)
    title(['a = ',num2str(a(j))])
end
```









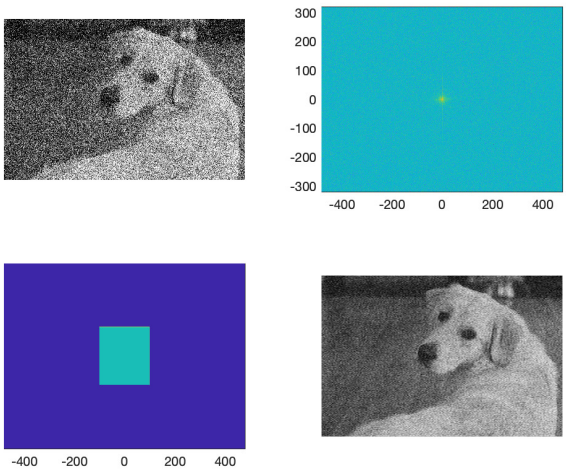
You can see that if we make the filter too narrow (over-filter), we lose all of the high-frequency detail. Conversely, if we choose the filter too wide, we hardly remove any of the noise. In practice, the ideal filter is somewhere in the

middle.

The Shannon Filter

As we've pointed out before, the Gaussian is just one of infinitely many filters that could be used. Let's investigate another one: the Shannon filter. This is just a step function - it takes the values 0 or 1 - so it is just a sharp cutoff in frequency space that removes all frequencies above some threshold.

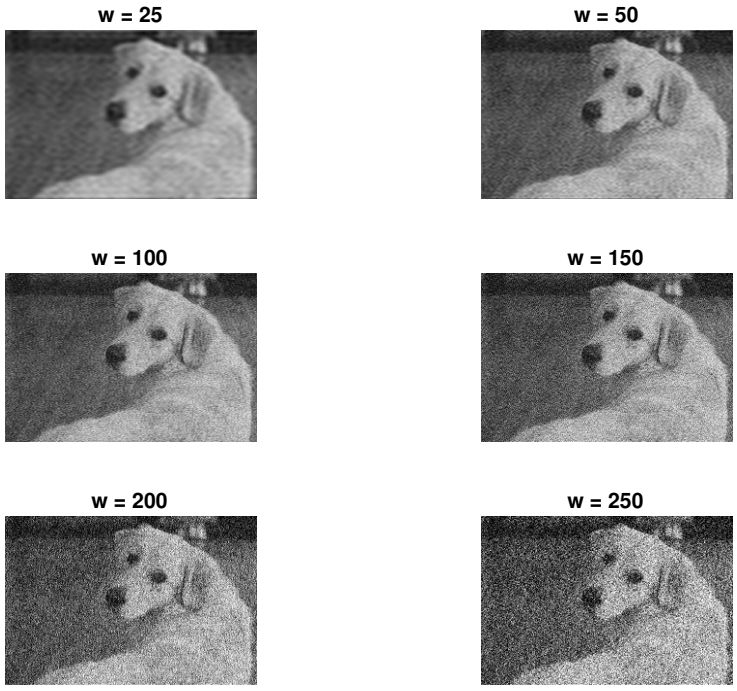
```
wx = 100;
wy = 100;
filter = zeros(size(Int));
filter([1:wy+1 Ny-wy+1:Ny],[1:wx+1 Nx-wx+1:Nx]) = ones(2*wy+1,2*wx+1);

Intf = Int.*filter;
Inf = ifft2(Intf);

figure(7)
subplot(2,2,1)
imshow(In)
subplot(2,2,2)
pcolor(fftshift(Kx),fftshift(Ky),log(fftshift(abs(Int))))
shading interp
subplot(2,2,3)
pcolor(fftshift(Kx),fftshift(Ky),log(fftshift(abs(filter))))
shading interp
subplot(2,2,4)
imshow(Inf)
```



Let's compare different filter widths:

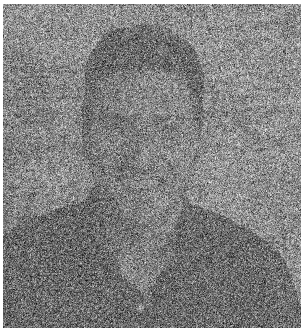
```
% Different filtering widths
w = [25 50 100 150 200 250];

% Plot to compare filter widths
figure(8)
for j = 1:length(a)
    wx = w(j);
    wy = w(j);
    filter = zeros(size(Int));
    filter([1:wy+1 Ny-wy+1:Ny],[1:wx+1 Nx-wx+1:Nx]) = ones(2*wy+1,2*wx+1);


    Intf = Int.*filter;
```

```
    Inf = ifft2(Intf);

    subplot(3,2,j)
    imshow(Inf)
    title(['w = ',num2str(w(j))])
end
```



In the case of our dog picture, we've looked at the original image so much that it can be hard to recognize the gains we are making by denoising the image. Let's try an example where you don't know what the original image looks like. To accomplish this, we will load an image and add a lot of noise to it before viewing it.

```
I = imread('mystery.jpg');
I = rgb2gray(I);
I = im2double(I);
In = imnoise(I,'gaussian',0,3);
figure(1)
imshow(In)
```



Let's use our code from above to denoise it with Shannon filtering.

```
Int = fft2(In);
[Nx,Ny] = size(In);

% Different filtering widths
w = [25 50 100 150 200 250];

% Plot to compare filter widths
figure(10)
for j = 1:length(w)
   wx = w(j);
   wy = w(j);
   filter = zeros(size(Int));
   filter([1:wx+1 Nx-wx+1:Nx],[1:wy+1 Ny-wy+1:Ny]) = ones(2*wx+1,2*wy+1);

   Intf = Int.*filter;
   Inf = ifft2(Intf);

   subplot(3,2,j)
   imshow(Inf)
   title(['w = ',num2str(w(j))])
end
```