UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
Department of Electrical and Computer Engineering

ECE 498MH Signal and Image Analysis

Lab 6

Fall 2014

Assigned: Thursday, November 13, 2014 Due: Thursday, November 20, 2014

Reading: `http://www.owlnet.rice.edu/~elec539/Projects99/BACH/proj2/wiener.html`

Lab 6.1

This lab will design several filters for the purpose of image denoising. Download a noisy image from `http://courses.engr.illinois.edu/ece498mh/fa2014/lab6image.png`. Read it into matlab using `imread`. You should find that this is a $144 \times 111 \times 3$ matrix; the three planes are the red, green, and blue planes.

(a) Download `http://courses.engr.illinois.edu/ece498mh/fa2014/plotimage.m`. Your `runlab.m` function should call `imread` to load the noisy image, then call `plotimage` in order to create figure 1. This is a function that rescales the image (so every pixel value is between zero and one), and then plots it. The image itself is shown in the upper left corner. Its middle column is shown in three plots in the upper right corner (red, green, and blue planes). Its middle row is shown in three plots in the lower left corner (red, green, and blue planes). Compare the signal plots to the image. Notice that the signals show the image features, but that the noise looks much worse in the signal plots than in the image; the eye naturally denoises the image, so it looks better than it actually is.

(b) The desirable image features are generally constant-color regions, ranging from 5 pixels to 50 pixels in width, depending on the angle. As a first de-noising filter, let's create an ideal low-pass filter. A constant color region could be considered half of a sine wave, so let's set the cutoff frequency to be $\pi/5$ radians/sample.

In your `runlab.m` function, use the `firpm` function to create a 20-sample FIR lowpass filter. Set the desired response equal to $D(\omega) = 1$ for $0 \leq \omega \leq \frac{\pi}{5}$ (note that $\frac{\pi}{5}$ radians/sample is 0.1 cycle/sample, which is 0.2 times the Nyquist frequency). Give your filter a wide transition band: set the desired response to $D(\omega) = 0$ for $\frac{2\pi}{5} \leq \omega \leq \pi$ (that is, 0.4 Nyquist up to Nyquist).

You have now created a one-dimensional filter, but we need a two-dimensional filter. There are more computationally efficient ways to do this, but for now, let's create a 2D filter by taking the outer product of the 1D filter with itself: If your 1D filter is in the row vector `B`, then the 2D filter can be `B2=B'*B;`. Use `size` to verify that `B2` is a 20×20 impulse response.

Run `for c=1:3, denoised1(:,:,c)=conv2(A(:,:,2),B2,'same'); end`. In `figure(2)`, use `plotimage` to show the image `denoised1`. It should look considerably denoised, and also considerably smoothed.

Plot the impulse response of the lowpass filter in the lower right quadrant as follows: `subplot(2,2,4); imagesc(B2);`. Notice that it looks like a two-dimensional sinc function!

(c) Wiener filters are easiest to design in the spectral domain, so let's find the level spectrum (power spectrum, expressed in decibels) of the noisy image.

Find the power spectrum of A as `for c=1:3, SAA(:,:,c)=abs(fft2(A(:,:,c))).^2;end`.

Find the level spectrum as `LAA=10*log10(SAA);`. In order to make it easier to plot, clip it 60dB below its peak: `peak=max(LAA(:)); LAA=max(LAA,peak-60);`

Use `plotimage` to show the level spectrum in `figure(3);`. Oops: `fft2` puts the low frequencies in the four corners, which is counter-intuitive, since in class, we always show the low frequencies in the middle. Fix this using the `fftshift` function: `plotimage(fftshift(LAA));`.

Do the same thing to show the power spectrum of the filtered image, `denoised1`, in `figure(4);`. Find the level response of the lowpass filter, using something like

`LBB=20*log10(abs(fft2(B2)));peak=max(LBB(:));LBB=max(LBB,peak-60);`.

Use `subplot(2,2,2); imagesc(fftshift(LBB));` to show the power spectrum of the filter you designed. Notice that it's not exactly the filter you wanted—it has a little bit of ripple, and a very wide transition band—but it's not too bad.

(d) Let's design a Wiener filter. First, let's use the image itself to estimate the noise power spectrum. Find a 20×20 region of the image where you believe the original image has perfectly constant color, so that the only variation is noise. Cut this out, as `V=A(m+[0:19],n+[0:19],:);` where (m, n) is the upper right corner of the region you've chosen (use `imagesc` to make sure you've cut the right region). Use `for c=1:3, V(:,:,c)=V(:,:,c)-mean(mean(V(:,:,c))); end` to remove the average from each color plane (try using `imagesc` to plot this; the constant color should have been removed, and it should be just noise). Compute the power spectrum of the noise as `for c=1:3,SVV(:,:,c)=abs(fft2(V(:,:,c))).^2; end`.

Now that we have the noise power spectrum, let's estimate the signal power spectrum. The clean signal is unknown, but the noisy signal is available. We want to find a 20×20 estimate of the noisy signal power spectrum, `SXX`. One way to do this is by averaging the power spectra from all 20×20 blocks in the image. Figure out how many such blocks exist in the image, then do something like

```
[M,N,C]=size(A);
MBLOCKS=floor(M/20);
NBLOCKS=floor(N/20);
for m=0:(MBLOCKS-1),
  for n=0:(NBLOCKS-1),
    for c=1:3,
      foo=abs(fft2(A(20*m+[1:20],20*n+[1:20],c))).^2;
      SXX(:,:,c)=SXX(:,:,c)+foo;
    end
  end
end
SXX = SXX/(M*N);
```

If the signal and the noise are independent, their power spectra should just add, so we should be able to find $S_{SS}(\omega) = S_{XX}(\omega) - S_{VV}(\omega)$. Try `SSS=max(0,SXX-SVV);` to keep it from going negative.

Create a Wiener filter frequency response as `H2=SSS./SXX;`. Inverse Fourier transform it as `h2=ifft2(H2);` Note: although this formula for the Wiener filter is exactly right using a DTFT, it is not quite right using a DFT, therefore it's possible this Wiener filter may not be quite as optimal as previously claimed.

Let's try it. It should be interesting to see what benefit, if any, comes from having color-dependent Wiener filters. Try `for c=1:3, denoised2(:,:,c)=conv2(A(:,:,c),h2(:,:,c),'same'); end;`.

In `figure(5);`, use `plotimage` to show the denoised filter, then use `imagesc` to show the impulse response `h2` in the lower right corner. In `figure(6);`, do the same thing with the level spectra of the denoised image and of the filter.