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

ECE 498MH Signal and Image Analysis

Homework 10

Fall 2013

Assigned: Friday, November 22, 2013 Due: Friday, December 6, 2013

Reading:

Problem 10.1

Define the cross-correlation and cross-power spectrum to be either

$$R_{zs}[\tau] = E\left[z[n]s[n-\tau]\right], \quad P_{zs}(\omega) = \mathcal{F}\left\{R_{zs}[\tau]\right\} \tag{1}$$

or

$$R_{zs}[\tau] = z[\tau] * s[-\tau] = \sum_{n=-\infty}^{\infty} z[n]s[n-\tau], \quad P_{zs}(\omega) = Z(\omega)S^*(\omega) \tag{2}$$

whichever is more convenient to you. In fact, you can switch back and forth from one part of this problem to the next, if you wish.

Suppose that the speech signal is $s[n] = a^n u[n]$ (for example, maybe $a = e^{-j\pi B_1/F_s}$, as in homework 9).

(a) Find the power spectrum, $P_{ss}(\omega)$, and sketch it as a function of ω. In order to do this, you can either use the fact that $R_{ss}[\tau] = \left(\frac{1}{1-a^2}\right) a^{|\tau|}$, or you can use the fact that $S(\omega) = \frac{1}{1-ae^{-j\omega}}$. Either fact leads to a correct solution; I think the second fact is easier to use, but your mileage may vary.

(b) Suppose that $v[n]$ is a random signal whose autocorrelation is $R_{vv}[\tau] = \delta[\tau]$. Find $P_{vv}(\omega)$, and sketch it as a function of ω.

(c) Suppose that $y[n] = s[n] + v[n]$. You want to create a filter $h[n]$ so that $z[n] = h[n] * y[n]$, in order to minimize $E\left[(z[n] - s[n])^2\right]$. Find $H(\omega)$, and sketch it as a function of ω.

Matlab Exercises

Problem 10.2

The course web page contains two noisy pictures of a baby panda: one corrupted by Gaussian noise, one corrupted by failure noise.

(a) Read in the image. Normalize it by adding together the three colors, and subtracting the mean. Plot the grayscale image, and hand in a copy of this image (using `imagesc`, which automatically shifts and scales the pixel values so that the full range of pixel intensities is visible):

```
gaussianpanda = double(imread('gaussianpanda.png'));
sumy = sum(gaussianpanda,3);
normedy = sumy - mean(mean(sumy));
colormap gray
imagesc(normedy);
```

(b) Compute the power spectrum of the noisy image as the magnitude-squared Fourier transform. Estimate the power spectrum of the original image by subtracting the average value, $P_{ss}(\omega) \approx \max(0, P_{yy}(\omega) - \text{mean}(P_{yy}(\omega)))$. Plot the power spectra of $y[n]$ and $s[n]$, over only the first 20 or so frequency components, and hand in a copy:

```
Pyy = abs(fft2(normedy)).^2;
Pss = max(0,Pyy - mean(mean(Pyy)));
N = length(Pss(1,:));
omega = [0:(N-1)]*2*pi/N;
subplot(2,1,1);
plot(omega(1:20),Pss(1,1:20));
subplot(2,1,2);
plot(omega(1:20),Pyy(1,1:20));
```

A few things to notice: (1) $P_{yy}(0) = 0$. That's because you subtracted the mean. (2) $P_{yy}(2\pi/N)$ is really big. Most of the power in an image is concentrated at very low frequencies. (3) Subtracting the mean doesn't change the power spectrum very much. That's because the mean value is very small – most samples have very small amplitude.

(c) Estimate the Wiener filter as $H(\omega) = P_{ss}(\omega)/P_{yy}(\omega)$, then in verse FFT. In the first sub-plot, show 100 samples from the first row of the filter. In the second sub-plot, show a scaled image of the first 100×100 pixels of the filter.

```
H = Pss./Pyy;
h = ifft2(H);
subplot(2,1,1);
plot(h(1:100));
subplot(2,1,2);
imagesc(h(1:100,1:100));
```

(d) Create a practical filter by applying a Hamming window to $h[n]$. You'll then have to make it symmetric by flipping the impulse response top-to-bottom and left-to-right, and prepending them. Create a sub-plot with three sub-images: one showing the 2D hamming window, one showing the windowed filter, and one showing the complete symmetric filter.

```
n=[0:99];
hamwin=0.54+0.46*cos(pi*n/99);
ham2 = repmat(hamwin,[100 1]).*repmat(hamwin',[1 100]);
h2 = h(1:100,1:100).*ham2;
h3 = [fliplr(h2), h2];
h4 = [flipud(h3); h3];
subplot(2,2,1);
imagesc(ham2);
subplot(2,2,2);
imagesc(h2);
subplot(2,1,2);
imagesc(h4);
```

(e) Wiener filter the Gaussian panda, `imagesc` the result, and hand in a copy. Has it been effectively denoised?

```
for color=1:3,
  denoisedpanda(:,:,color)=conv2(gaussianpanda(:,:,color),h4,'same');
end
denoisedpanda=max(0,min(255,denoisedpanda));
image(denoisedpanda/255);
```

(f) Try using the same technique to remove shot-noise from shotnoisepanda.png. Create (and hand in) a plot with four sub-plots: one sub-plot showing the shotnoisepanda, one showing the Wiener-filtered shotnoisepanda, one showing the gaussianpanda, and one showing the Wiener-filtered gaussianpanda. You might find that Wiener filtering works pretty well for both types of noise, but that it works better for Gaussian noise; for example, Wiener-filtering a shot-noise-corrupted image results in a darker output image, because shot noise is one-sided (it only blacks out pixels, never makes them brighter).

```
shotnoisepanda = double(imread('shotnoisepanda.png'));
for color=1:3,
  d2(:,:,color)=conv2(shotnoisepanda(:,:,color),h4,'same');
end
d2 = max(0,min(255,d2));
subplot(2,2,1);
image(gaussianpanda/255);
subplot(2,2,2);
image(denoisedpanda/255);
subplot(2,2,3);
image(shotnoisepanda/255);
subplot(2,2,4);
image(d2/255);
```

(g) Try median filtering the shotnoisepanda, taking the row-median of the column-median of pixels in each 3×3 local square:

```
medianpanda=shotnoisepanda;
[M,N,K]=size(medianpanda);
for m=2:(M-1), for n=2:(N-1), for k=1:K,
  medianpanda(m,n,k)=median(median(shotnoisepanda(m+[-1:1],n+[-1:1],k));
end, end, end
image(medianpanda/255);
```