2.161 Signal Processing: Continuous and Discrete

Fall 2008

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
DEPARTMENT OF MECHANICAL ENGINEERING

2.161 Signal Processing - Continuous and Discrete
Fall Term 2008

Problem Set 9

Assigned: November 20, 2008 **Due:** December 4, 2008

Problem 1: Given a waveform $x(t)$ with a pdf of the form

$$\begin{aligned} p(x) &= \lambda e^{-\lambda x} \qquad x > 0 \\ &= 0 \qquad \text{otherwise} \end{aligned}$$

calculate (a) the mean μ_x, and (b) the variance σ_x^2.

Problem 2: White noise is passed through an ideal low-pass filter with a cut-off frequency of ω_c. Find the autocorrelation function of the filtered noise.

Problem 3: A Mini-project: Deblurring the Hubble Telescope Images

> Read the Class Handout: *Introduction to Two-Dimensional/Image Processing* before attempting this mini-project.

As you know, the scientific world was shocked by the poor quality of the initial images from the Hubble Space Telescope after its launch April 24, 1990. After it went into orbit, it became clear that something was wrong. While the pictures were clearer than those of ground-based telescopes, they were blurry, and not the pristine images promised.

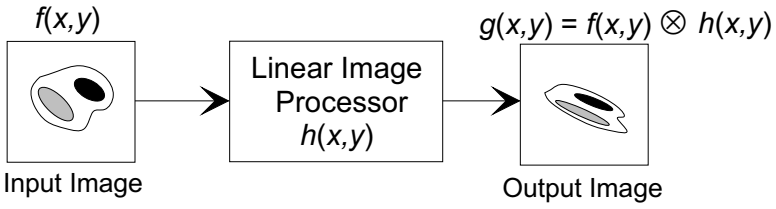
The telescope's primary mirror had been polished with a "spherical aberration" flaw. It was just slightly the wrong shape, causing the light that reflected from the center of the mirror to focus in a different place than the light reflected from the edge. The tiny flaw about 1/50th the thickness of a sheet of paper, was enough to seriously degrade the images.

While the problem was well understood, the hardware fix was time consuming and very expensive. The rest is history, we now look in awe at the images from Hubble.

In this mini-project we are going to use signal processing techniques to deblur some Hubble images. We provide you with two blurred images, your task is to sharpen them and restore the full fidelity.

- An gray-scale image can be thought of as a two-dimensional function $f(x, y)$, where f is the intensity of the image at point x, y in the image.
- A RGB (red-green-blue) colored image can be thought of having three separate component images (a red $f_r(x, y)$, a green $f_g(x, y)$, and a blue $f_b(x, y)$) image.

- A digital image is made up of pixels - in our case our images will be 512×512 pixels.
- Many optical distortions, such as blurring, can be thought of as a linear filtering operation, with the difference that 1) the functions being filtered are functions of space (not time), and 2) are two-dimensional.

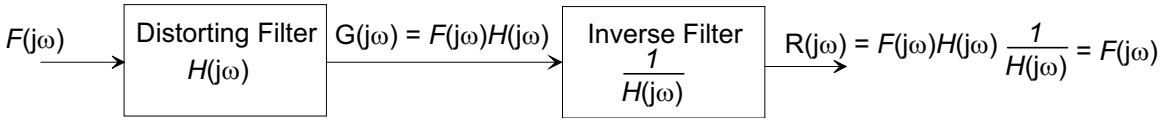


- An image processing filter has a two-dimensional impulse response, known as the *point-spread function* $h(x, y)$, reflects how an impulsive point at the center of the source image is "spread" out in space by the filter.
- The operation of a linear image processing filter is that of 2-D convolution of the 2-D input image with the 2-D point-spread function (see the class handout).
- The two-dimensional Fourier transform is defined (see the class handout), and analogous with 1-D signal processing, the filtering may be done my multiplication in the Fourier domain, that is

$$Y(\mathrm{j}\,u, \mathrm{j}\,v) = F(\mathrm{j}\,u, \mathrm{j}\,v)H(\mathrm{j}\,u, \mathrm{j}\,v)$$

where $F(\mathrm{j}\,u, \mathrm{j}\,v) = \mathcal{F}_2\{f(x, y)\}$, $Y(\mathrm{j}\,u, \mathrm{j}\,v) = \mathcal{F}_2\{y(x, y)\}$, $H(\mathrm{j}\,u, \mathrm{j}\,v) = \mathcal{F}_2\{h(x, y)\}$ is the image filter's 2-D frequency response, and u, and v are spatial frequencies with units of radians/unit length.

Let's assume that the aberrant mirror in the Hubble telescope has caused the true image $f(x, y)$ to be blurred by convolution with a known point-spread function $h(x, y)$, so that the image we are given is $g(x, y)$. The process of recovering $f(x, y)$ is known as *deconvolution*, or *inverse filtering*. In one-dimensional temporal linear processing, we might do the following:



By definition, the inverse filter has a frequency response of $1/H(\mathrm{j}\,\omega)$ so that $F(\mathrm{j}\,\omega)$ is recovered at its output. While it sounds easy, the process can be fraught with difficulties, for example if $H(s)$ has zeros on the imaginary axis the gain of the inverse filter will go to infinity at that frequency. Also inverse filters often tend to amplify any high frequency noise.

In the two-dimensional case, if we know the point-spread function $h(x, y)$ that blurred the image, we can construct the inverse filter $1/H(\mathrm{j}\,u, \mathrm{j}\,v)$ and use it to restore the original.

Details of the MATLAB Tools You Might Need:

- There are two blurred Hubble images to deal with. These are contained in the MATLAB files `blurredimage1.mat`, and `blurredimage2.mat`. Each image is a 512×512 RGB colored image. When loaded into MATLAB the image will be named `blurred` in each case (deal with them one at a time). You can view the image with

```
load blurredimage1;
image(blurred);
```

- MATLAB has 2-D fft functions `fft2()` and `ifft2()`. Don't forget that you will have to work on all three color planes in the image.

- The mirror design engineers have told us that the distortion was equivalent to convolution with a Gaussian kernel such as produced by MATLAB's image processing toolbox function `fspecial()`:

```
hsize=25;
sigma = 3;
h = fspecial('gaussian', hsize, sigma);
```

  You might want to use `mesh(h)` to take a look at it.

  If you don't have access to the Image Processing toolbox, here is how you can construct the 2-D kernel:

```
siz   = (hsize-1)/2;
[x,y] = meshgrid(-siz(2):siz(2),-siz(1):siz(1));
arg   = -(x.*x + y.*y)/(2*sigma*sigma);
h     = exp(arg);
sumh = sum(h(:));
h  = h/sumh;
```

  Note that the sum of elements is set to unity.

- When you have processed an RGB image and want to display it, the display function `image(myRGBimage)` requires that all values be real, positive, and in the range 0 to 1.

Do not use the Image Processing Toolbox to solve this problem. We want to see each step you took in solving the problem. Submit the listing of your MATLAB script as a .m file, and upload your deblurred images before 1pm on the due date.

Problem 4: (This is from the take-home Quiz 2 from last year.)

You have probably realized by now that I just love the sound of my own voice! I used

Windows Sound Recorder on my PC to make a recording of me saying "good morning" so that I can play it to myself each morning when I wake up. It just makes my day...

Now, here's the problem. Windows Sound Recorder has a button "Add Echo", - which surprisingly adds an echo to a recorded sound. Unfortunately I clicked on this button before I saved the ".wav" file, and the result is that I sound as if I am speaking from inside a tin can! I hate that, and I want you to help restore my voice to its natural beauty.

Your job is to analyze the recorded file to find out exactly how Microsoft added the echo, and then to remove it.

On the class MIT Server is a Microsoft ".wav" file **PS9Prob2.wav** that contains me saying "good morning" to myself. Download the file and load it into MATLAB using the `wavread()` function. If your computer has speakers (or headphones) you can play the sound using the `sound()` or `soundsc()` functions. Use `help audio` to see the sound handling functions in MATLAB.

(a) Given a function $f(t)$ with autocorrelation $\phi_{ff}(\tau)$, find an expression for the autocorrelation function $\phi_{gg}(\tau)$ of a waveform $g(t)$ where

$$g(t) = f(t) + \alpha f(t - \Delta) \tag{1}$$

is $f(t)$ contaminated by an echo with delay Δ and amplitude α. Express your answer in terms of $\phi_{ff}(\tau)$.

(b) In the case of a discrete-time signal contaminated by an echo, let

$$g_n = f_n + \alpha f_{n-m}. \tag{2}$$

where m is the echo delay. Express the echo generation as a linear filtering operation and derive the the transfer function $H(z)$.

Use MATLAB to analyze the .wav file, using the results of part(a), to estimate the parameters α and m that Microsoft **Windows Sound Recorder** used to create the echo in the .wav file.

(c) Using your results from (b), design a digital filter that will eliminate the echo contamination from the .wav file. Derive the transfer function, and the difference equation.

(d) Use MATLAB to apply your inverse filter to the sound file and show (hopefully) that it has removed the echo.

Document your process so that we can understand exactly what you did.