Computer Assignment 4

Goal

The goal of this computer assignment is to introduce you to the world of image processing, and to demonstrate that much of the math and intuition you've learned this semester are useful for processing images. Believe it or not, images have frequency content (spatial frequency, not temporal frequency) and can be high and low-pass filtered to do things like de-noising, edge detection, and smoothing. Because images have two dimensions (x and y), their Fourier Transforms will also be two dimensional (representing frequencies F_x in the x-direction and F_y in the y-direction). By comparison, time signals such as the one we are used to seeing in class are only one dimensional (time) and therefore produce one dimensional Fourier Transforms.

Documentation

You will be using the Matlab "Image Processing Toolbox". Mathworks has placed extensive documentation online at `http://www.mathworks.com/help/toolbox/images`. There is lots of good information there that will give you a good starting point for thinking about image processing. However, if you don't want to read the whole thing then at a bare minimum you should read `https://www.mathworks.com/help/images/what-is-image-filtering-in-the-spatial-domain.html` These pages also have great resources:

- `https://www.mathworks.com/help/images/_f16-15268.html`
- `http://www.mathworks.com/help/images/linear-filtering.html`

myFFT2

Download the `myFFT2` function from Canvas. This function uses the built-in Matlab `fft2` function to create a plot of the 2-dimensional FFT for an image. It is super easy to use. Suppose you have some image named `image1`, you would just type `myFFT2(image1)`. The resulting 3D image can be rotated by dragging the mouse to see it from different angles. You can also create the plot using decibels on the z-axis, which will make for nicer looking plots for some of the images. You do that using `myFFT2(image,'db');` Note that the x- and y-axes represent *normalized* frequency. A normalized frequency of 1 represents exactly half of the sampling frequency, essentially equivalent to $\Omega = \pi$ rads/sample.

Part 1

Download the file `hw4Data.mat` from Blackboard. Inside it you should find six images named `im1` through `im6`. You can view an image by typing:

```
imagesc(im1);
```

A couple of command are useful for making the image black-and-white (as opposed to some other colormap) and also for scaling the x- and y- axes to the same size so that the image isn't stretched:

```
colormap gray;
axis equal;
```

(There are a bunch of other colormaps. Try `colormap default` to get a blue-red image or `help graph3d` to get a complete list of available colormaps.)

View the six `im` images as well as their Fourier Transforms. Then answer the following questions:

- `im1` and `im2` have the same x-frequency (16 pixels per period) and y-frequency (constant), but are obviously very different images. How are the images different and how are those differences manifest in the Fourier Transforms?

- `im2` and `im3` are identical but rotated by 90°. How can this difference be seen in the Fourier Transforms?
- `im1` and `im4` are both vertical stripes, but with different frequencies. What are those frequencies and can they be detected in the Fourier Transforms?
- In the Fourier Transform of `im5` we see a sinc pulse on the x- *and* y-axes. Can you justify why this is based on what `im5` looks like?
- I created `im6` by multiplying `im2` and `im3`. Using that information, explain why the FT of `im6` is what you would expect.

In each case, explain how the differences in the images correlate to the differences in the Fourier Transforms. You should be able to apply all of your signal processing intuition to answering these questions.

Part 2

Here you will create and compare low-pass image filters using some built-in Matlab functions. Start with these commands:

```
load hw4Data
figure(1); imagesc(moon); axis equal off; colormap gray;

h_lpf = ftrans2(fir1(16,0.1));
moon_low = imfilter(moon,h_lpf);
figure(2); imagesc(moon_low); axis equal off; colormap gray;
```

Note that `fir1` will create a 16-point FIR filter with a cutoff frequency of 0.1π rads/sample (the "π" is inferred); `ftrans2` converts that filter into a 2-dimentional (image) filter.

Examine the filtered image. What is the effect of lowpass filtering the image? Use `myFFT2` to look at the Fourier Transform of the lowpass filter `h_lpf`. Does the FT make sense? Vary the cutoff frequency (as low as 0 and as high as 1). How does the value effect the filtered image?

Part 3

Repeat Part 2 but using a *highpass* filter:

```
h_hpf = ftrans2(fir1(16,0.2),'high');
```

Answer all the same questions as in Part 2. What is the effect of highpass filtering an image?

Honors Students

Read the Image Processing Toolbox help page titled "Designing Linear Filters in the Frequency Domain".

The `hw4Data.mat` file has two images in it named `pears` (a picture of some pears) and `pears_noisy` (the pears picture plus white noise). Recall that white noise is noise that exists at all frequencies.

Create a filter to try to remove some of that white noise. There is no perfect solution: just do the best you can and explain your filtering strategy.

Write a 1-2 page paper (using the IEEE template) that explains your findings. Submit your paper (MSWord format only) along with any code you've written in a single zip file via Canvas. Only one team member needs to submit. Submissions due Monday April 16th at 5pm.