Activity 12: Fourier Transforms

Primary Goal: Learn how to work with Fourier Transforms.
Secondary Goal: Learn how to apply spectral filtering.

1.) Viewing Fourier Transforms

First load and display the "pout" image.

```
A = imread ('pout.tif');
```

To compute the 2D Fast Fourier Transform (FFT), we type:

```
F = fft2 (A);
```

The Fourier Transform is complex-valued, so we have to display the absolute value (magnitude) of the complex numbers. If you display the raw image F, you won't see much.

```
imagesc(abs(F));  colormap gray;
```

If you zoom in on the upper left corner of the image, you will see a single white dot. This is the DC-term, which is the sum of all values in the original matrix A. We generally like to display the DC-term in the center of the image. We can shift the DC-term to the center with the `fftshift` command.

```
imagesc( abs(fftshift(F)) );
```

But we still just see a white dot because the DC-term is much larger than the other values in the matrix F. To normalize the image, we can examine the log image.

```
imagesc( log(1 + abs(fftshift(F))) );
```

To transform the image back to the spatial domain, we can use the Inverse Fast Fourier Transform (IFFT). We take the absolute value to remove any imaginary terms caused by numerical imprecision.

```
A2 = abs( ifft2(F) );
```

You should see this new image A2 is identical to the original image A.

2.) Spectral Filtering

Recall the equation of a circle with radius R and center (n/2,m/2) is given by:

$$\left(x-\frac{n}{2}\right)^2+\left(y-\frac{m}{2}\right)^2=R^2$$

To create a binary 0-1 mask D that describes a circle of radius R=20 pixels for an image A, we type the following code into a script:

```
R = 20;
[m,n] = size(A);
[x,y] = meshgrid(1:n,1:m);
D = [ (x-n/2).^2 + (y-m/2).^2 < R^2 ];
```

Display this image D. You should see a white circle on a black background.

A shifted Fourier Transform image has low frequencies in the middle near the DC-term and high frequencies farther out. A low-pass filter will remove the high frequencies of a Fourier Transform, leaving just the low frequencies behind. If we apply our mask D to a shifted Fourier Transform, it will create an ideal low-pass filter.

```
F2 = D.*fftshift(F);
```

View the log image of this Fourier Transform. You should see the high frequencies have all been set to black.

Now we use the Inverse Fourier Transform to make a new image.

```
A2 = abs( ifft2(F2) );
```

a.) Try the radius values R=50 and 100. What happens to A2 as R gets larger?

As R gets larger, more high frequencies pass the filter and the Inverse Fourier Transform resembles the original image more.

b.) A high-pass filter blocks the low frequencies. allowing only the high frequencies to pass through. Modify your mask D so that it inverts the circle. That is, we now want a black circle on a white background. Apply this filter and view the resulting image A2. What does a high-pass filter do?

3.) Notch Filtering

Fourier Transforms are particularly good for picking up on periodic (repeating) patterns in an image. Create a new script. Let's load the "pout" image again, convert to double format, and look up its size.

```
A = imread('pout.tif');
A = double(A);
[m,n] = size(A);
```

Now let's create a vertical striped image S that has the same size as the image A.

```
[x,y] = meshgrid(1:n, 1:m);
S = 3 + sin(x / 3);
```

View the image S. Now compute the Fourier Transform of S and view the shifted image. You should see just 3 dots arranged horizontally. The 2 dots to either side of the DC-term indicate that vertical stripes were detected at a specific frequency.

Now let's corrupt the image A with stripes.

```
B = A.*S;
```

View the resulting image. You should see a striped version of the original image A.

Now compute the Fourier Transform of B.

```
F = fft2(B);
```

View the shifted image. You should also look at the log image. Try to locate the 2 bright spots that indicate the vertical stripes.

Now let's create a mask D that removes these 2 bright spots. First create a mask D of all 1's that has the same size as image B.

```
D = ones( size(B) );
```

Next let's locate the bright spots on F. We want the user to click on the bright spots in the Fourier image and then mask out those spots. The vector P stores the (x,y) coordinates of the point you click on. Then we can add a small black circle of radius 5 pixels centered around the coordinates of P.

```
for i = 1:2
     P = ginput(1);
     D( find( (x-P(1)).^2 + (y-P(2)).^2 < 5^2 ) ) = 0;
end
```

You should now have an image D with 2 small black circles. Note the `find` command locates all (x,y) points that satisfy the inequality in parentheses. Also note that if we wanted to mask out more than 2 spots, we just need to increase the loop counter's ending.

Apply this mask to F.

```
F2 = D .* fftshift(F);
```

Now take the Inverse Fourier Transform of F2 and view the result. If it worked, you should see the stripes have been *(mostly)* removed. This process of removing specific frequencies is called notch filtering.