Image Processing I

EECE/CS 253 Fall 2011

Assignment 4: The 2D Fourier Transform and Convolution

Due at 12:00 midnight on Thursday 20 October 2011

The goals of this lab are to learn (1) about the Fourier Transform of an impulse, (2) how to perform convolution in the spatial domain and in the frequency domain, and (3) how to filter images via convolution.

In this lab assignment (and all the others) you will be writing your own image processing functions. You may not use the functions from the `matlab` image processing toolkit by themselves or within your functions to solve the problems in the labs. You may compare the results of your own functions against those of the IP toolkit for your own information but those results may not be used in your lab reports.

Help can be gotten on any `matlab` command (or function) by typing `help command_name`, *e.g.* to get help on `imwrite()`, type `help imwrite` at the command prompt. Another way is to press the <F1> key, select the **index** tab in the window, and type the command name into the search box.

In completing this assignment (and the others) if specific images are not supplied, you may use any images you like providing that they are of the type specified by the problem description (*e.g.* 24-bit truecolor). Please do not use images that are obscene or gruesome. In general, an image is OK if you could show it to your grandmother without upsetting her or embarrassing yourself. A wide variety of images are available under creative commons license on the web site `www.flickr.com`. If you are not the photographer, be sure to credit the person who did take the picture.

1. **Convolution Programs and Computation Speed.**

   Write two programs to convolve an arbitrary image, I, with a weight matrix, h. The weight matrix should of type double, $m \times n$, single band. Both programs should work on either 3-band (truecolor) or 1-band (grayscale or monochrome) images of type uint8 or double. Each band of a 3-band image should be convolved with the same weight matrix. The output should be of type double. (If you need a uint8 image from the convolution routine, convert it inline. *E.g.* `J = uint8(ConvSpace(I,h));`)

   (a) Write a program to perform the convolution in the *spatial domain*. Use the shift-multiply-accumulate approach described in the lecture notes.

   (b) Write a program to perform the convolution in the *frequency domain* by multiplying the Fourier Transforms of the image and the weight matrix and taking the inverse transform of the result. Use Matlab's `fft2` function.

   Recall that within your program you must allocate a temporary image the same size as I, and copy h into it so that the center of h coincides with the center

of the temporary image. You must apply `ifftshift` to the image that contains h but not to I prior to computing the Fourier transforms. Why is that? See `EECE253_06_FourierTransform.ppt`, slides 63-65, for the definition of the Matlab image center and for examples of `fftshift` and `ifftshift`.

Remember to take only the real part of the inverse transformed result. (The imaginary part may exist but it will contain only extremely low amplitude noise when filtering a real image with a real convolution mask.)

(c) Which is faster, spatial or frequency domain convolution? Write a simple test program to compare the computation times of your two programs. I suggest that you create a small constant image such as `I=ones(512,512)`. Within the main loop of the test program, generate convolution masks of size $n \times n$ (*e.g.*, `h=ones(n,n)`). Convolve `I` with `h` using both programs over a range of integers such as $n = 1, 2, 3, \ldots, 33$ and record the time it takes each one to complete. For example, to compare the computation times of programs named `ConvSpace` and `ConvFreq` over convolution masks ranging in size from 1×1 to 33×33 one could define an array `t=zeros(32,2)`. Then within the main loop:

```
tic;
J = ConvSpace(I,h);
t(n,1) = toc;
tic;
J = ConvFreq(I,h);
t(n,2) = toc;
```

Plot the two time sequences on the same graph. Describe how time required by the two algorithms changes as a function of the weight matrix dimensions. If the curves cross each other, indicate the values of n where that happens. What function should you use use for a given convolution mask size?

2. **Sinusoids and FTs of impulses.**

(a) Find and load an $R \times C \times 3$ (truecolor) image, I. Create an $R \times C \times 3$ zero image, h. Place an impulse – a 1 – in all three bands at some location, (r_1, c_1) in the interior of h, at neither the center nor close to a corner. Convolve I with h using your Fourier transform convolution program. Display the result. Describe as precisely as possible what has happened to the image.

(b) Create a $512 \times 512 \times 1$ image of all zeros. Place a unit impulse (a one) at the center of the image. Apply `ifftshift` to the image, take its Fourier transform, and apply `fftshift` to the result. Find the maximum and minimum values of the FT of the image. [Hint: Find the maximum and minimum values of the four components, the real, imaginary, magnitude, and phase of the FFT using `real(F)`, `imag(F)`, `abs(F)`, and `angle(F)`.] Describe the results – what are the real, imaginary, magnitude and phase of the FT of an impulse at the origin? It should not be necessary to display these to understand what has happened. Do the same thing for an image of size $512 \times 512 \times 1$ with value one at each location.

Be sure to look at the numbers in the vicinity of the origin of the FT. Comment on the relationship between the impulse and the constant image in both the spatial and Fourier domains.

(c) Now create another 512^2 image of zeros and place an impulse at some location, (r_0, c_0), other than the center. Find the maximum and minimum values of the FT of the image as in part (b). Contrast the results – how do the results differ? In particular, how do the power spectra differ and how do the phases differ? What mathematical property of the Fourier transform does this illustrate? Include the corresponding equation in your report. Hint: Try displaying the four components using, for example,

```
figure,imagesc(real(F2c)),colormap(gray(256)),truesize;
```

If it is difficult to see what is going on try placing the impulse at location (r, c) where $r = c = p + 1$ where $p = 512 \cdot 2^{-n}$ for some integer $n \geq 1$.

(d) Create a rectangular, one-band image, I, such that I has dimensions $R \times C \times 1$, where $R \geq 384$ and $C = 3R/2$. Let r_0 be any single integer from the set $\{8, 9, \ldots, 64\}$ and set $c_0 = r_0$. Place an impulse in I at locations (r_0, c_0) and $(-r_0, -c_0)$ with respect to the *centered* origin of I. Calculate the frequency, wavelength, and orientation of the sinusoid represented by the location (r_0, c_0).

Compute the inverse Fourier transform of I (after applying `ifftshift`) and display its real component. Verify through direct measurement that the sinusoid has the predicted frequency, wavelength, and orientation.

(e) Create another image like the one in problem 2d. Use the same values of r_0 and c_0. Place four impulses in the image, one each at at locations (r_0, c_0), $(r_0, -c_0)$, $(-r_0, -c_0)$, and $(-r_0, c_0)$ with respect to the *centered* origin of I. Compute the inverse Fourier transform of I (after applying `ifftshift`) and display its real component.

What is the angle in degrees between the two sinusoidal components?

For a given r_0 what must c_0 be so that the angle between the two components is $90°$?

Generate that image and display its real component.

3. **Gaussian lowpass, highpass, and bandpass filters.**

(a) Find a highly textured 24-bit, 3-band color image with dimensions of at least 512×512 to use throughout this problem. Display it in your report.

(b) Write a program to generate a space-domain (SD) symmetric Gaussian filter with arbitrary variance centered on a type double image of arbitrary size. Generate it from the equation at the bottom of slide 33 in Lecture 8. The inputs to the program should be the variance of the gaussian and the size of the image to be output. Be sure that the Gaussian is centered on the Matlab `fftshift` center Normalize the result so that the output image sums to one.

(c) Use the program you wrote for the previous part (problem 3b) to generate three SD Gaussian filters, each with a different variance and size $R \times C$. Space the variances so that one is small, one is medium, and large, The largest one should be on the order of 1/4 the size of the smaller of the two dimensions R and C. Apply each one to the image you selected in part and describe the results.

4. Generate a highpass and a bandpass filter in the FD from the Gaussians in 3c. It is most straightforward to do so in the FD and then to take the inverse FFT of the result. Compute the sums of each filter in the SD. What should the sums be? If they are not what they should be, adjust the filters to make them so. Take the real part of the inverse FFT of each and find the support of of each one in the space domain. The support is the set of pixels on which the value of the filter is greater than or equal to $(1/(3.5\sigma))(1/256)$. If can find that in Matlab by displaying the image

```
S = 255*(abs(h)>(1/(3.5*sigma*256)));,
```

where h is the space domain filter. Given the results of problem 1 determine the best domain in which to perform the filtering.

Apply each filter to the image you selected in part and describe the results.

5. **Image Sharpening**

Use the image from problem 4 in this exercise.

(a) *Direct Sharpening.* Convolve the image with a 3×3 Laplacian (See p. 59 of Lecture 7)to create a highpass image. Sharpen the original image by adding intensity-scaled versions of the highpass image to it. Use small, intermediate, and large scale factors between 0 and 1 and display the results. What scale factor results in a sharper image that looks natural and is not very noisy? How can you modify the HPF to get the same results with a single convolution?

(b) *Unsharp Masking.* Unsharp masking (USM) sharpens an image, I, by adding to it a scaled highpass image created by subtracting a Gaussian lowpass image from the original. There are two control parameters involved: the standard deviation, σ, of the Gaussian, and a scale factor, α. Let $G(\sigma)$ be a Gaussian filter. Let

$$B(\sigma) = I * G(\sigma)$$

be the lowpass filtered I. Let

$$D(\sigma) = I - B(\sigma)$$

be the difference image. Then the unsharp masked image is

$$\begin{aligned} J &= (1+\alpha) D(\sigma) + B(\sigma) \\ &= D(\sigma) + B(\sigma) + \alpha D(\sigma) \\ &= I + \alpha D(\sigma). \end{aligned} \tag{1}$$

Experiment with various values of α and σ to find a natural looking, but clearly sharper version of of your image. Include versions of the USM image to show the effect on it of varying the parameters.

6. Additional problem for graduate students.

   Write a function that will generate a Difference of Gaussians (DoG) filter as in problem 4. The input arguments to the program should be the inner and outer σ values, $\{\sigma_1, \sigma_2\}$ and the image dimensions, (R, C). The output should be an $R \times C$ array of class `double` with the DoG centered.

   Filter (convolve) a single truecolor image, I, with three different DoG filters, one within the range $\sigma_1, \sigma_2 \in (1, 8)$, one with $\sigma_1, \sigma_2 \in (16, 32)$, and one with $\sigma_1, \sigma_2 \in (64, 128)$. That is σ_1 and σ_2 should be taken from within the given interval with $\sigma_1 < \sigma_2$. Leave the resultant image as class `double` since it will have both positive and negative values.

   Write a function that will input the the filtered image, $J = I * DoG(\sigma_1, \sigma_2)$ and return a two-value image that has value 255 at every pixel except for pixels that lie on positive to negative *zero-crossings*, which should have value 0. Pixel (r, c) is on a positive to negative zero-crossing if $J(r, c) > 0$ and one of its 8-neighbors is zero or negative.

   Describe the results and the differences in characteristics of the three zero-crossing images. How might such images be useful?

Rules for laboratory assignments

1. Perform all the tasks listed in the instructions.
2. Explain the tasks you performed in detail.
3. Answer in writing in your report all the questions asked in the instructions.
4. Include in the report the original images you used and those resultant images that were specified in the instructions.
5. Include all computer code that you wrote and used, clearly documented, in an appendix.
6. All work must be yours and yours alone. Collaboration on the laboratory assignments is forbidden, with the following exceptions:
   (a) You may obtain help on any aspect of the homework from either Prof. Peters or the TA for this course.
   (b) You may obtain technical help on `matlab` from anyone you wish. However *you may not get direct help on the implementation of the specific algorithm* from another person except as noted in (a) above.
   (c) You may get help in obtaining the *input* images for the assignments from anyone you wish.

(d) You may get help in the formatting or storing or transmission of your reports, but *not the content*, from anyone you wish.

7. Write your results in a clear laboratory report format using MS Word, WordPerfect, LaTeX, or any other word processor with which you can embed images in text. I prefer that the reports be submitted in .pdf format, but that is not required. Submit your report to me as a file on Blackboard. If for some reason this does not work, you may submit your report on a CD-ROM.

8. Assignments are due at midnight on the day specified in the instructions or in class. The grade on a laboratory report will be reduced by 10 points (out of 100) for every day (24 hours) that it is late.