

CS 216/295: Image Understanding  
Fall 2008  
Homework 1 Due at beginning of class, 10/16

Review your class notes and Forsyth and Ponce Chapters 1,7,8,9

1. Show that convolution is associative, that is that  $(f \star g) \star h = f \star (g \star h)$
2. Show that correlation is not associative
3. Let  $g_1$  be a 1D Gaussian with variance  $\sigma_1^2$  and  $g_2$  be a 1D Gaussian with variance  $\sigma_2^2$ . Show that  $g_3 = g_1 \star g_2$  is also Gaussian. What is  $\sigma_3^2$ , the variance of  $g_3$ ? Why might this fact be useful in practice (i.e. to make Gaussian filtering run faster)?
4. Show that the 2D isotropic Gaussian filter  $g(x, y) = \frac{1}{2\pi\sigma^2} \exp \frac{-(x^2+y^2)}{2\sigma^2}$  can be written as the product of two 1D Gaussians, i.e.  $g(x, y) = g(x)g(y)$ . Why might this fact be useful in practice?
5. Play with the discrete Fourier transform (DFT) in MATLAB. Start out in 1D and make a signal which is of length 100 with a single impulse of height 1 in the middle. Compute the DFT of the signal and plot the magnitude of the spectrum. Now increase the width of the pulse from a single sample to a unit height “box function” 5 and 10 samples long and plot resulting the spectrum magnitudes. Now do the same for a Gaussian function with  $\sigma = 1$  and  $\sigma = 2$  (the Gaussian has infinite support but you should just analyze a finite chunk centered around the origin). Lastly, figure out how to do a 2D DFT and inverse DFT. Replicate the experiment we showed in class (shown as Figure 7.6 in FnP) by computing the DFT of two images, combining the phase from one and the magnitude from the other and compute the inverse DFT. Please submit the 4 plots of the magnitude spectrum and the 3 images (two inputs and their combination). Please feel free to also submit any other examples you played with which you found enlightening in understanding the DFT (e.g. DFT of your favorite function, presidential candidate, etc).

Hint: in MATLAB you will want to make use of the functions `fft`, `fft2`. You will also want to use `fftshift` in order to get frequency

domain plots like we showed in class where low frequencies are in the middle.

6. Write a gradient based edge detector in MATLAB. Your code should load in a grayscale image (use `imread` and convert to a double array using `im2double`). You can display the image using `imagesc`. Once you have loaded in the image, you should smooth the image with a Gaussian filter and then compute horizontal and vertical derivatives using a derivative filter. The amount of smoothing is determined by the  $\sigma$  of the Gaussian (which should be a parameter of your code). You can use `conv2` with the 'same' option to perform the required convolutions. Once you have computed the derivatives in the  $x$  and  $y$  directions using convolution, compute the gradient magnitude and orientation.

Please submit a printout of your MATLAB code along with example gradient magnitude and orientation images for two different settings of  $\sigma$  (see for example Figure 8.10 of FnP)

7. Implement a simple face detector based on convolution. Load in one of the two images of the class and display on the screen. Clip out a patch of the image containing a face (you may find `ginput` or `impixelinfo` useful to get the coordinates). Use this patch as a template in order to try and detect other faces in the image by convolution. Visualize the result of the convolution as an image.

You can try thresholding the result of convolution but you will note that around detected faces there will be many pixels with high values. In order to suppress non-maximal locations, zero out any pixels which are smaller than one of their 8 neighbors. (You can do this easily in MATLAB using array indexing tricks, e.g. in 1D

```
L = (x(2:end-1) > x(1:end-2)) % bigger than our neighbor to the left?
R = (x(2:end-1) > x(3:end))   % bigger than our neighbor to the right?
T = x(2:end-1) > threshold %above detection threshold?
maxima = R & L & T
```

Plot the locations of above threshold local maxima on top of the original photograph using the `plot` or `rect` command. Play around with the threshold to try to find a good tradeoff between detecting all the faces and not too many background detections. Submit a printout of

your code along with a visualization of the template you used and your final detection results on all three images.

Hints: Remember to flip the template before doing the convolution (use `fliplr/flipud`). Also try blurring your face template slightly before doing the convolution. This will tend to help it find faces which don't match exactly your original template.

8. Modify your code from the previous experiment to perform normalized convolution. If before you had the simple inner product computation:

```
response = conv2(I,T);
```

instead try out the squared difference between template T and the image I:

```
IT = conv2(I,T);  
Tsquared = sum(T.^2);  
Isquared = conv2(I.^2,ones(size(T)));  
squarediff = Isquared - 2*IT + Tsquared;
```

Also try “normalized convolution”:

```
response = conv2(I,T);  
energy = conv2(I.^2,ones(size(T))).^0.5;  
normalizedresponse = response ./ energy;
```

Which of these three variants work the best? Explain what (if any) problems they solve for your particular images (Note that you may need to adjust your choice of threshold differently for each).

Save the MATLAB figure showing the test image with your best detection results overlaid and email it to me. The best performing homework solution will be announced in class and win a prize!