CS 216: Image Understanding

Spring 2013

Homework 2

Due: writeup as a PDF file and a zip file containing your code should be uploaded to the EEE dropbox by 4/25/2014 by 11:59pm

Reading: Review your class notes and Forsyth and Ponce Chapters 1,7,8,9 and/or Szeliski Chapter 1,2,3.

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. 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. $g(x,y) = g_1(x)g_2(y)$.
4. If we have an image I of dimension HxW and we convolve it with a filter f of size MxN using the spatial domain formula given in class, what will the complexity be? How about if we use an FFT? Starting with the formula for 2D convolution, show that if our filter is the product of two functions, $f(x,y) = f_1(x)f_2(y)$, we can compute the convolution more efficiently.
5. Let g_1 be a 1D Gaussian with variance σ_1^2 and g_2 be a 1D Gaussian with variance σ_2^2. Using the facts mentioned in class about the continuous Fourier transform, show that $g_3 = g_1 \star g_2$ is also Gaussian. What is σ_3^2, the variance of g_3? Why might this fact be useful in practice (i.e. to make Gaussian filtering run faster)?
6. Experiment 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 submit at least one other example you experimented with which you found interesting, enlightening or suprising (e.g. DFT of your favorite function) and provide a brief explanation of why.

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.

7. 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 the derivative filter described in lecture. The amount of smoothing is determined by the σ 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 σ (see for example Figure 8.10 of Forsyth and Ponce)

8. Implement a simple object detector based on correlation. Load in one of the test images provided and display it on the screen. Clip out a patch of the image containing an object you want to detect (you may find `ginput` or `impixelinfo` useful to get the coordinates). Use this patch as a template in order to try and detect other instances of the object in the image by correlation. Visualize the result of the cross-correlation as an image using the `imagesc` command and the jet colormap. Initially experiment with choosing a relatively small "object" such as Dilbert's nose or a single letter.

You can try thresholding the result of correlation but you will note that around detected objects 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 image using the `plot` or `rect` command. Experiment with the threshold to try to find a good tradeoff between detecting all the instances of an object and but not too many background detections. Submit your code along with a visualization of the template you used, the convolution output, and your final detection results.

Hints: If you use the conv2 function then remember to flip the template before doing the convolution (use `fliplr/flipud`) so that you are performing correlation. You may want to try blurring your template slightly before doing the convolution or averaging together several object instances to make the template. This will tend to help it find version which don't match exactly your original template.

9. Modify your code from the previous experiment to compute the sum of squared differences (SSD). 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(sum(T.^2));
   Isquared = conv2(I.^2,ones(size(T)));
   squareddiff = Isquared - 2*IT + Tsquared;
   ```

   Which of these variants works 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 variant).

In your homework writeup, include an image showing the MATLAB figure with the test image with your best detection results overlayed (e.g. using the MATLAB rectangle function).