CS 216: Image Understanding

Spring 2018

Homework 2

Due: writeup as a PDF file and a zip file containing your code should be uploaded to the EEE dropbox by 4/30/2018 at 11:59pm.

Reading: Review your class notes, Klette Chapter 1.2-1.3, Szeliski Chapter 2,3 and/or Forsyth and Ponce Chapters 1,4,8,9

1. Prove that *convolution* is associative, that is that $(f \star g) \star h = f \star (g \star h)$. Show by example that *correlation* is not associative. You only need to work this out for 1D signals.

2. 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 (i.e. how many multiplications)? How about if we use the FFT "trick"?

3. 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. Is there a way to use this idea to speed up convlution with a 2D isotropic Gaussian filter $g(x, y) = \frac{1}{2\pi\sigma^2} \exp^{\frac{-(x^2+y^2)}{2\sigma^2}}$?

4. Experiment with the discrete Fourier transform (DFT) in MATLAB.

   (a) 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.

   (b) 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.

   (c) 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).

   (d) 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 Forsyth

and Ponce) of swapping phase and magnitude by (1) computing the DFT of two images of the same size, (2) computing a new set of coefficents which have the phase from the first image and the mangitude from the second image, (3) taking the inverse DFT of this combined spectrum to produce a new image.

In your writeup, please include the 4 plots of the magnitude spectrums (part a-c) and the 3 images (two inputs and their combination) in part d. 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.

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.

5. 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` and `colormap gray`. 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 include your MATLAB code in a script called `detect-edge.m` along with example gradient magnitude and orientation images for two different settings of σ in your writeup (see for example Figure 8.10 of Forsyth and Ponce)

6. 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` useful to get pixel coordinates for user clicks). 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 `colormap`

`jet`. 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 scores which are smaller than one of their 8 neighbors. You can do this easily in MATLAB using array indexing, e.g. in 1D this would look like:

```
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 in a script `detect-template-correlation.m` 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. Alternately you can use `imfilter` which directly performs the correlation computation.

7. Modify your code from the previous experiment to compute the sum of squared differences (SSD) of pixel values. 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);
   ```

Note that the squared-difference will be small where there is a good match, so you will probably want to add a negative sign in order to re-use the same maxima detection code you used for correlation-based detector.

Please submit a script `detect-template-ssd.m` containing your implementation. 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 or plot function).

Does SSD work better than correlation? Why? Explain what (if any) problems they solve for your particular images (Note that you may need to adjust your choice of threshold differently from the one you used for cross-correlation.