EE 133 - Digital Image Processing
Department of Electrical and Computer Engineering
Tufts University Spring 2017
Problem Set #3

Distributed: Feb. 16, 2017 Due: March 2, 2017

Problem 3.1

Compute by hand the 2D convolution of the following two images

$$f(m,n) = \begin{cases} \max(m,-n) & |m+n| < 2 \\ 0 & \text{else} \end{cases}$$

$$h(m,n) = \begin{cases} m + (-1)^n & m,n \in \{0,1\} \\ 0 & \text{else} \end{cases}$$

Verify your results using the Matlab function `conv2`.

Problem 3.2

Implement in Matlab a 2D convolution operation on two images, $f(m,n)$ and $h(m,n)$. Suppose that f has M_f rows and N_f columns and h has M_h rows and N_h columns. The convolution should return an image that is of size $M_f \times N_f$. The idea here is that f is the image and h is going to be a (much) smaller convolution kernel used to do averaging, extract edges, perform matched filtering, etc. As we discussed in class to do this, you have to handle the edge effects in some way. Your convolution program should take as input a variable that indicates which method is to be employed. The following edge options should be supported

1. Fill in all required border pixels with zero
2. Fill in the required border pixels by assuming your image is one period of a 2D periodic signal.

You are to write this function from scratch. Do not use any built-in image processing functions from Matlab or Python such as `conv2`. The function should be able to process any f and h and you can assume that the size of f is larger than that of h and that h has an odd number of rows and columns. When it comes time to fill in the "corners," you will need to determine the right thing to do for each of the four cases. To be handed in for this problem are the following:

1. The documented code for your function
2. A formal writeup indicating how you approached each of the edge-extension options and how you handled the corners
3. Verification that your program works. Specifically, in the file `ps03s117_matlab_things.mat` are three "images" (`image1`, `image2` and `image3`) as well as two filters (`h1` and `h2`). With these, please do the following:
   (a) To verify the correctness of your zero boundary condition, convolve `image1` using the filter `h1` using (a) your code with zero boundary conditions and then (b) the built in

Matlab function `conv2`. Display the results to show that the two images are the same. Also, compute the error

$$e = \sqrt{\sum_{x,y} (g_1(x,y) - g_2(x,y))^2}$$

with g_1 the result of using your code and g_2 the image obtained using `conv2`. You should see that the error is on the order of 10^{-11}.

(b) To verify the correctness of your periodic boundary conditions, use your code to compute (a) the convolution of `image2` and `h2` and then (b) the convolution of `image3` and `h2`. Repeat these two convolutions but this time compute the results by hand. For each of the two images, plot the theoretical convolution result and the result using your code.

Problem 3.3

This problem should be done entirely in Matlab

1. Create a 256×128 image that is all white except for a 20 pixel by 30 pixel black square centered in the middle. To figure the correct pixel value for black and white, assume 8 bit grayscale. Using `imagesc` and `colormap` display the image in grayscale. Figure out how to use the `axis` command to display the image so that the height is twice the width.

2. Compare your convolution code against the `conv2` function by convolving the image you just created with the following filters:

$$h_1 = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad h_2 = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \quad h_3 = \frac{1}{9}\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Display the results and explain what each filter does

Problem 3.4

It is frequently the case that one wants to "find" specific objects in a image such as eyes for red eye removal, cells in a microscopy context, automobiles for traffic monitoring etc. Because 3D objects can appear quite differently in 2D images depending on how they are facing the camera (i.e., their *pose)*, their distance to the camera, lighting conditions, and such, the object detection problem is in general really, really hard. The basic method though upon which most of the sophisticated techniques are based is quite understandable in terms of fundamental image filtering ideas.

Suppose we denote by $t(x, y)$ the object we wish to find. The variable t in this case indicates that this is the *template* for our problem. Object detection in images is then done using a process known as *template matching* which is nothing more than convolution of the input image with a filter whose impulse response is $h(x, y) = t(-x, -y)$. For some input image f, we let $g(x, y) = f(x, y) * h(x, y)$; that is, $g(x, y)$ is the convolution of f and h.

1. Define the template matching operation as

$$g_1(x, y) = \int \int t(\alpha - x, \beta - y) f(\alpha, \beta) d\alpha\beta. \tag{3.4.1}$$

How are $g_1(x, y)$ and $g(x, y)$ related?

2. Suppose that $f(x,y) = t(x,y)$. Either 1/ prove mathematically using the Cauchy-Schwartz inequality or 2/ show with a convincing Matlab demo and argue persuasively that $g(x,y)$ obtains it maximum value at $x = y = 0$. What is that maximum value in terms of the template?

3. Suppose now that the input image is a shifted version of the template $f(x,y) = t(x-x_0, y-y_0)$. Now where will the maximum value of $g(x,y)$ be and how is this helpful in terms of finding things in images?

4. Now let's see how this works in practice. Load the file `ps02s117_text.tiff` (zipped with this PDF) into Matlab and plot the image. Your job is to use template matching to find all the locations of 'o's in the image. Based on the result of the convolution, how do you decide exactly where in the image all those o's are? Do you find them all? Do you find other extraneous stuff? Please hand in a discussion of these questions supported by images.

Problem 3.5

Here we consider a family of Gaussian filters, $h(x,y,t)$,indexed by "time", t :

$$h(x,y,t) = \frac{1}{2\pi\sigma^2(t)} e^{-\frac{x^2+y^2}{2\sigma^2(t)}} \quad \sigma(t) = \sqrt{2Dt} \tag{3.5.2}$$

where D and t are both real numbers greater than zero. We call $g(x,y,t)$ the result of filtering $f(x,y)$ by $h(x,y,t)$.

1. Using the image `ps03s117_test_image.jpg`, compute and plot the result of using these filter for a few values of t. Start with t close to zero and then proceed to find a couple of other values so you clearly see what is going on. That is, you should be able to find a "middle" t and a "large" t. Comment on what this family of filters is doing.

2. For the same values of t you used above, plot $g(x,y,t+\delta) - g(x,y,t)$. You will again need to find "good" values of $\delta > 0$. Specifically, you should be seeing edge-like images from these plots. Using the linearity of convolution, you should be able to find "equivalent" filters (differences of Gaussian) that are being used to generate these images. Plot the impulse response of these filters and explain why we are seeing edges.

3. Show that $h(x,y,t)$ satisfies the following diffusion equation:

$$\frac{\partial h}{\partial t} - D\nabla^2 h = 0 \quad \text{with} \quad \nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \tag{3.5.3}$$

   Argue that as t goes to zero, $h(x,y,z)$ becomes a delta function in time and space. Thus, you have found the *impulse response* (also known as a Green's function) to the diffusion equation.

4. From basic linear systems, you know that the output to a linear shift invariant system is the convolution of the impulse response with the input. The same is true for the diffusion equation. If we think of $f(x,y)$ as the initial condition to the diffusion equation then $g(x,y,t)$ can be written either as the convolution at the start of this problem or as the solution to the PDE:

$$\frac{\partial g(x,y,t)}{\partial t} - D\nabla^2 g(x,y,t) = 0 \quad g(x,y,0) = f(x,y) \tag{3.5.4}$$

More specifically in linear systems-speak, g is the zero input response. The question for this part of the problem is to explain using the PDE why taking differences of g in time gives you edge information.

Problem 3.6

Problem 5.7 on page 268 of the Birchfield text

Problem 3.7

Read Section 5.2.4 of the Birchfield text and do Problems 5.13 and 5.14 on page 269

Problem 3.8

Problem 5.18 on page 269 of the Birchfield text