Anthony Poerio (adp59@pitt.edu)

CS1674: Homework 2 - Written

Due: 9/12/2016, 11:59pm

QUESTIONS

1) **What is the effect of the sigma parameter of a Gaussian, on the appearance of the image that has been convolved with a Gaussian?**

   The sigma parameter of a Gaussian distribution refers to the **variance**. Higher sigma values mean there is more variance in the distribution. A Gaussian image filter is used to 'smooth' an image by taking samples from the area **around** each pixel, so that we can remove noise from the image. (Noise will be an outlier value, and we can normalize the image by averaging with the Gaussian.)

   When we convolve images using a filter whose values are drawn from the Gaussian distribution—then a higher sigma means that our filter covers a wider area of the image for each discrete calculation. For this reason, a large sigma (or variance) leads to more averaging across the entirety of the image, and thus it creates a blurrier image.

   Therefore: Higher sigma yields → Greater Blur.

2) **Explain how sharpening an image with filters works, in English (not in math). Intuitively, what steps are we performing?**

   Sharpening an image requires the application of two separate filters to the image.

   With the first filter, we double the underlying numerical value of individual each pixel—but this, in and of itself, won't sharpen anything: because the *relative values* haven't changed it all. Next, we apply a blur filter to image, averaging the values of a larger—but still local—area of the image.

   Finally, we subtract the values obtained by the second blurred filter from the doubled values from our first filter. In doing this, we obtain a sharpened image where we are essentially 'subtracting' a multiple of the local average from each pixel's doubled value. This accentuates the each pixel's own values in relation to the area around it—and we call this 'sharpening'. Intuitively, we're exaggerating the difference between pixel and those around it by some constant factor.

3) **Say I am at a pixel (r, c), where r is the row index and c is the column index. How can you find the difference between pixels to the right of me and pixels to the left of me, i.e. between pixels (r, c-1) and (r, c+1), using a filter? What about the difference between pixels below/above me, i.e. (r+1, c) and (r-1, c)?**

For differences between right/left, you can use this filter → [0, 0, 0; 1, 0, -1; 0, 0, 0]

For differences between top/bottom, you can use this filter → [0,1,0; 0, 0, 0; 0, -1, 0]

4) **How can you highlight edges in an image using filters?**

To highlight edges between columns (vertical edges), we can apply a filter using this matrix:

[1, 0, -1; 2, 0, -2; 1, 0, -1]

To highlight edges between rows (horizontal), we can apply a filter using this matrix:

[-1, -2, -1; 0, 0, 0; 1, 2, 1]

Intuitively: We want to look at the pixels either Left/Right or Above/Below our target column, and computer their relative differences. A filter (at least in Matlab) is an efficient way to do this.

5) **Say you have an image of a tree, and an image of a sky. You compute the dI/dx and dI/dy image gradients for all pixels, using the answer to Question 3 above. Which image would have a larger number of high-magnitude gradients? Why?**

I expect that ***the image of the tree would have a larger number of high-magnitude gradients***. This is because an image of sky should be roughly uniform in color and light/darkness across the entire image (assuming it's a clear day, and not a lightning storm, tornado, etc.).

Whereas, the image of a tree will be dark/brown where the tree's trunk is, green where the leaves are, and so on. Moreover, trees often have complex patterns in their foliage, with leaves, branches and the like all interweaving. Such complex patterns will create a large number of high-magnitude gradients, especially in comparison to a more uniform image like the sky (presumably on a clear day).

6) **How can you find patterns in an image (e.g. let's say you're looking for a plus sign in images) in an image using a filter? Would a 3x3 filter work for any image, i.e. where the plus signs appear at different sizes and orientations? Explain why it would/wouldn't.**

To find patterns in an image using a filter, you need to create a matrix which can highlight the differences you are looking for. Sometimes the matrix needed to do this will have to be relatively large (if you are looking for a complex pattern). Pursuing this same train of thought—I don't think it would be possible to identify plus signs successfully with just a 3x3 matrix.

For example, I know we can identify vertical edges using a matrix like this one, call it A:

1	0	-1
2	0	-2
1	0	-1

Similarly, we could identify horizontal images using the transpose of that matrix, A^T.

In both cases, we need to use values all three rows/columns to successfully ID the edges we are looking for.

So, if we need to identify both Horizontal and Vertical edges in one single matrix, it would need to be something like this: [1, 2, 1; 2, 0, 2; 1, 2, 1]. And that's where the problem arises. Using this matrix construction from A → we need one side to have positive values, and the other side to have negative values.

But here, that's impossible. Some values, like A(3,2) would need to be BOTH positive and negative, if we are identifying horizontal and vertical edges with just one filter. And that's impossible. Therefore, as a contradiction of sorts, we cannot successfully identify plus signs with just a 3x3 matrix. We'd need to either use two matrices, or a larger and more complex matrix.

7) **Say you want to illustrate both the finer and coarser details in an image, separately. Describe a process that allows you to produce 3 "detail" images (i.e. images that only show the sharp detail of the image), the first one being very fine, the second a little coarser, and the third even more coarse**?

I know we can 'sharpen' and image by first taking a filter like this:

[0 0 0; 0 2, 0; 0 0 0] → Call this A

And subtracting a blur filter like this one:

1/9 [1 1 1; 1 1 1; 1 1 1] → Call this B

So the whole operation would be: A – B.

--

Moreover, I know that this 'sharpening' process image will display the 'sharp' details of image, generally. But if I'm understanding correctly, we need to isolate the details along a continuum: very fine > fine > coarse.

I think we can do this by:

a) Perform the A – B operation on our image as outlined above. → Call the output C
b) Run a filter to find pixel values which are:
   i) **Very fine**: Greater than 3x larger than the pixels values in their local area (say 3x3)
   ii) **Fine:** Greater than 2x larger than all pixels in their local area—BUT less than 3x as large.
   iii) **Coarse:** The remaining pixels less than 2x larger than all pixels in their local area.

I'm not sure exactly what operations would do this in Matlab right now—but the general idea is that I want to compute differences for each pixel with respect to the local area around them—then place them in separate buckets based on their respective values, using a filter. I'd sharpen the image first, to make sure we had the detailed portions identified up-front, and then I'd run my segmentation algorithm.