Problem Set 5
CMSC 427
Distributed April 4, 2015
Due: Tuesday, April 21 2015

This problem set asks you to implement correlation, create a couple of filters, and use bilinear interpolation to change the size of images. To do this, you will need to have code that reads in an image, and writes an image. You can do this using any language that Zheng agrees to and is familiar with. One good option is to use C along with the C++ routines we provided for the last problem set to read and write images. Whatever language you use, you cannot use any built in functions or libraries for convolution, correlation, creating image filters, resizing, interpolation, or other image processing functions. You can use built in functions or libraries to read or write images. I would not recommend using Java, which has some issues with handling images properly. If you have any doubts, please ask.

One tip. Be sure that when you are performing computations with images, you use floating point. Your filters will not have integer values, so you want to be sure that your intermediate computations are accurate. Once you've created a new image, you could convert it to an eight bit unsigned integer if you wanted to, but doing this for intermediate calculations will cause trouble.

We are providing three images to use in testing your code. The first is an image of a dog. We provide you with the results of our code when applied to the dog image. We also provide an image of a swan and a rug, although we do not include results for these images. We will ask you to test your code using these images. In these cases, include printed versions of the result images in your hardcopy, and images saved as bmp files in the electronic version.

1. **20 points.** Write a function to convert an image from rgb to grayscale. This function takes as input an rgb image in the form of a 3D matrix. At each pixel, convert the three colors to a single grayscale value using the following formula. If a pixel has values [r,g,b], take the grayscale value to be .299*r+.587g+.114b. The function will output an image with the same dimensions as the input image, in the form of a 2D matrix. Test your code on the dog and swan images and turn in the results.
2. **20 points.** Write a function to perform correlation. This function will take an image as input, along with a 2D matrix containing the filter. Test this function using a 7x7 box filter (this is just a 7x7 array in which all values are 1/49). Try this on the dog and swan images.
3. **20 points.** Write a function to create a Gaussian filter. This function should take as input the standard deviation of the Gaussian, and return a 2D matrix containing the filter. Recall that to do this you sample the Gaussian at integer values. Make sure that the width of the Gaussian is big enough to capture values up to three standard deviations from the mean (which is 0, the center of the pixel). Also make sure the values in the filter sum to 1, and that the width and height of the

filter is odd. Use this code to create a filter with a standard deviation of 1.5, and apply it to the dog and swan images.

4. **20 points.** Write a function to create a sharpening filter. The input to this function will be a standard deviation. Suppose T is a filter that just contains a 2 in the center and 0 everywhere else. Then suppose G is a Gaussian filter with the standard deviation that is input to the function, using your function from problem (3). The sharpening filter is T-G. Apply this filter to the dog and swan images.
5. **20 points.** Write a function to resize an image. This function should take as input an image, I, and a scale value, s, and output a new image J. The width(/height) of J will be the width(/height) of I times s. To find the intensity at J(r,c), you will use the intensity at I(r/s, c/s). The tricky part of this is that r/s and c/s may not be integers. So you will need to use bilinear interpolation to find the appropriate values (don't just round off s*r and s*c). Test your code by scaling the dog image by a factor of 3 and .75. Scale the rug image by .05. You should see that the rug image looks kind of funny when shrunk so much. Try scaling it again, but first smoothing it with a Gaussian that has a standard deviation of 10. Does it look better?
6. **Challenge Problem, 30 points.** Implement a function to perform bilateral filtering. This is a kind of extension of Gaussian filtering. One way to think of filtering with a Gaussian is that we are replacing each pixel with a weighted average of its neighbors. Each neighbor pixel is weighted using its distance from the central pixel. The weight is computed by pushing this distance through a Gaussian function with some standard deviation. With the bilateral filter, the weight depends on the product of this Gaussian weight and the result of applying a Gaussian (with a different standard deviation) to the difference in intensities of the pixels.

   Let's spell this out with an equation. Let G(x; 0, s) denote the value of x using a Gaussian distribution with a mean of 0 and a standard deviation of s. Now suppose we are filtering an image, and we want to find the filtered value for pixel x (here x is a vector containing the pixel's two coordinates). The new value will be a weighted average of x's neighbors. Suppose y is one of these neighbors. Then the weight we will use for y will be:

   $G(||x-y||; 0, s1)*G(|I(x)-I(y)|; 0, s2)$.

   One more detail. These weights won't add up to 1, so we need to normalize them so they do.

   Your bilateral filtering code can contain four inputs: the image, two standard deviations, one for spatial differences and one for differences in intensity, and a width, which indicates how far away from the center pixel you'll look for neighboring pixels. Test your code using the dog and swan images, with a standard deviation of 3 for spatial distance and 32 for pixel intensities, and a neighborhood width of 8.

In doing this problem, it may help to look at the paper “Bilateral Filtering for Gray and Color Images,” by Tomasi and Manduchi, http://www.super.tka4.org/materials/lib/Articles-Books/Filters/Bilateral/tomasi98bilateral.pdf
or the Wikipedia article on the bilateral filter.