Lab Assignment #2

Due Monday, October 8 by 11:59pm
(upload to CatCourses)

Title: Image resizing

In this lab, you will implement a function that resizes a grayscale image. It will perform either nearest neighbor or bilinear interpolation to generate the resized image. Resizing an image to a smaller size is called downsampling. Resizing an image to a larger size is called upsampling.

a) Write the function myimresize that takes as input

- A grayscale image (a matrix, not a filename)
- The size (number of rows and columns) of the resized image
- A string with values 'nearest' or 'bilinear'

and outputs the resized image (a matrix, not a filename). This function should use either nearest neighbor or bilinear interpolation to determine the values in the output image. Turn in the code for your myimresize function.

b) The nearest neighbor interpolation can be performed in the myimresize function. The bilinear interpolation should be performed by the function mybilinear which you must write (and which should be called by your myimresize function). This function takes as input

- Four pixels locations (these should be integer values)
- The pixel values at those locations
- The location of the interpolated pixel (this need not be integer valued)

and provides, as output, the value of the interpolated pixel. You can implement either of the two approaches to bilinear interpolation that we covered in lecture. Turn in the code for your mybilinear function.

c) Use the image Lab_02_image1.tif to evaluate your resizing function. This image should have size 300x300 pixels. Use your function to create the following 4 images:

- A 40x75 downsampled image using nearest neighbor interpolation
- A 40x75 downsampled image using bilinear interpolation
- A 425x600 upsampled image using nearest neighbor interpolation
- A 425x600 upsampled image using bilinear interpolation

Turn in printouts of these four images.

d) Now, use your image resizing function to resize each of the above images to its original size (300x300). That is, at the end of this step, you should have the following 4 images:

- A 300x300 downsampled then upsampled image using nearest neighbor interpolation
- A 300x300 downsampled then upsampled image using bilinear interpolation
- A 300x300 upsampled then downsampled image using nearest neighbor interpolation
- A 300x300 upsampled then downsampled image using bilinear interpolation

These are our "reconstructed" images. You don't need to turn in these images.

e) Write the function myRMSE that computes the root mean squared error between two images. This function should take as input two grayscale images of the same size and output a single floating point value, the RMSE. For two images of size MxN, the RMSE can be computed as:

$$RMSE = \sqrt{\frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\left(I1(m,n) - I2(m,n)\right)^2}\ .$$

RMSE is a numeric method for computing the difference between two images. It is commonly used to evaluate the effectiveness of image reconstruction algorithms by computing the pixelwise difference between the original and the reconstructed images. We will use RMSE to compare our original image with the upsampled/downsampled and downsampled/upsampled versions. Turn in the code for your myRMSE function.

Your project write-up should include both qualitative and quantitative evaluation of the two kinds of interpolation. Your qualitative evaluation can consist of visually comparing the reconstructed images with the original (and with each other). Which interpolation method produces the best results visually? This is probably best done on a computer monitor, zooming if necessary. For your quantitative evaluation, compute and compare the RMSE values between the original image and the four reconstructed images. (Include your RMSE values in your report.) Which interpolation method results in the best RMSE values? Is a higher or lower RMSE value better? Does RMSE always agree with subjective evaluation (that is, can we do away with cumbersome subjective evaluations and just use RMSE to evaluate reconstruction algorithms)?

Hints:

1. You might want to focus on getting your image resizing function to work using nearest neighbor interpolation before considering the more challenging bilinear interpolation case.
2. My upsampled/downsampled nearest neighbor interpolation reconstructed image had a RMSE value of 0 when compared to the original image. This is also the case when using the built-in imresize function.
3. The Image Processing Toolbox contains the function imresize which is very similar to the function you will implement. It has a few extra options such as performing bicubic interpolation. You can experiment with this function but don't expect your function to provide the exact same results (mine doesn't). If you want to compare your results against those computed using imresize, make sure to set the (optional) parameter 'Antialiasing' to false: `output=imresize(input,[300 300],'bilinear','Antialiasing',false)`

   This forces imresize to skip a prefiltering step when downsampling an image. We didn't implement this. The RMSE values of my reconstructed images and those reconstructed using imresize were similar but not the same.