EECS 490

Project 1

Image Quantization and Sampling

Edward Venator
Due: 9/8/2011
Handed In: 9/8/2011

This report examines methods of subsampling a grayscale image and their comparative effects. Methods tested include nearest neighbor, bilinear, and bicubic subsampling. The subsampled images are compared side-by-side and used to construct an image pyramid. In addition, this report examines the effects of reducing the number of gray levels in an image.

Technical Discussion

Subsampling—Three Methods

There are many methods of subsampling an image, but there are three that are the most common. Nearest neighbor is the simplest and least computationally intensive, but bilinear and bicubic interpolation give smoother results that are more similar to the original (higher resolution) image. Bicubic interpolation is used by most image editing software. This task is greatly simplified by MATLAB's built in imresize() function, which can use a whole host of algorithms to perform image resizing.

To increase the resolution using the nearest neighbor method, new pixels are inserted between the existing pixels, and these new pixels are given the same value as the nearest existing neighbor pixel. This results in large, square blocks of color, but is the simplest subsampling method.

A more complex method of increasing the resolution is bilinear interpolation. Bilinear interpolation uses the four nearest neighboring pixels to determine the value v(x,y) using equation 2.4-6 in Gonzalez and Woods's *Digital Image Processing*:

$$v(x,y) = ax + by + cxy + d \qquad \text{GW 2.4-6}$$

Bilinear interpolation gives a "smoother" appearance than nearest neighbor. This is because the algorithm is inherently a low-pass filter.

Bicubic interpolation is similar to bilinear interpolation, but instead of using the four nearest pixels, it uses the sixteen nearest pixels and calculates the value according to Gonzalez and Woods's equation 2.4-7:

$$v(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij}x^i y^i \qquad \text{GW 2.4-7}$$

Whereas bilinear interpolation is a first-order equation, bicubic is a third-order equation. This allows it to more accurately "guess" what the missing pixels should be. The result is an image that is much closer to the original, despite having the majority of the original pixel data discarded.

Reducing Number of Gray Levels

Most grayscale images are represented using 8-bit values for each pixel, meaning there are 256 possible values, from 0 (black) to 255 (white). Reducing the number of possible values can reduce the amount of space needed to store the image. In many cases, reducing the number of gray values has little to no impact on the recognizability of the image. This project examines the effect of reducing the number of gray levels by factors of two. This is performed by dividing each pixel's value by the reduction factor, rounding to the nearest integer, and then multiplying by the reduction factor, as shown in Equation 1 below.

$$v'^{(x,y)} = round\left(\frac{v(x,y)}{2^i}\right) * 2^i \qquad \text{GW 2.4-7}$$

Image Pyramids

An image pyramid, also known as a mipmap, is a method of storing the same image as multiple resolutions to aid in rapid rendering for computer graphics. This project uses MATLAB's imresize() function with bicubic sampling to generate smaller images, then combines them in a single image.

Results

Subsampling

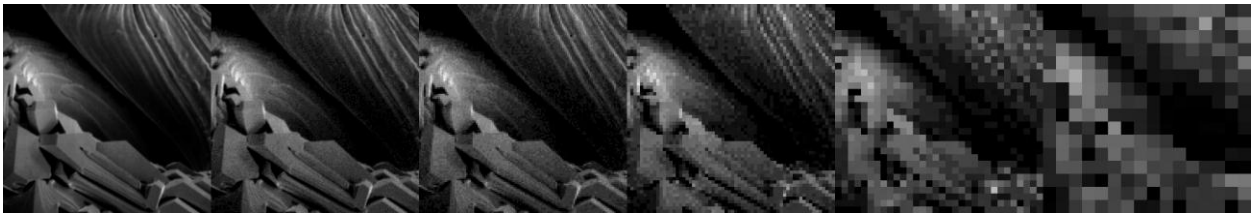

Figure 1: Image resized by nearest neighbor. Left-most image is the 512x512 pixel original. Each subsequent image has been downsampled by a factor of two, then enlarged back to the original size.

As discussed above, the nearest neighbor method results in a very blocky final image. The image becomes unrecognizable after it has been shrunk to about one 16^{th} of its original size. This algorithm may be fast and simple, but it is not very effective at reconstructing the original image.

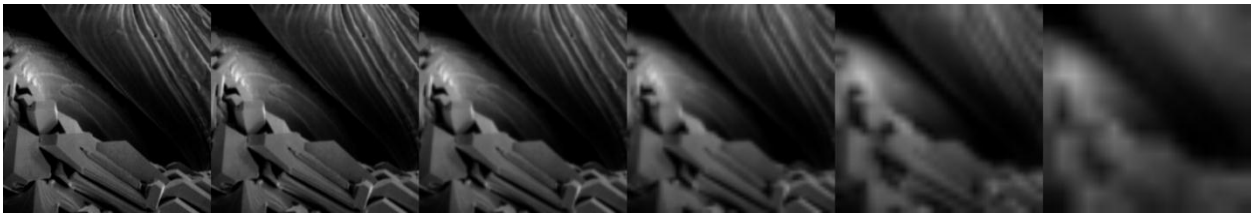

Figure 2: Image resized using bilinear interpolation. Left-most image is the 512x512 pixel original. Each subsequent image has been downsampled by a factor of two, then enlarged back to the original size.

The bilinear method has a smoothing effect on the image. Even though the image furthest to the right has been reduced to only $1/32^{nd}$ of the original image data, bilinear interpolation can still reconstruct something recognizably similar to the full size image. Of course, the fine textures and details of the original image are lost, and the image has a decidedly "blurry" appearance.

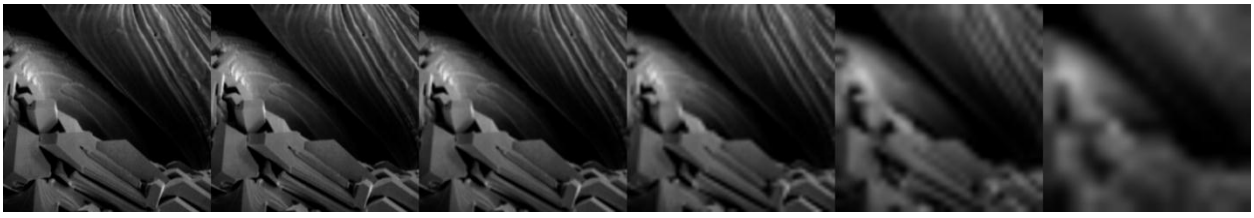

Figure 3: Image resized using bicubic interpolation. Left-most image is the 512x512 pixel original. Each subsequent image has been downsampled by a factor of two, then enlarged back to the original size.

The bicubic method yields very similar results to the bilinear method for this image, but retains slightly more detail. Notably, the edges in the lowest resolution images retain more sharpness than the images created using bilinear interpolation.

Decreasing Number of Gray Values

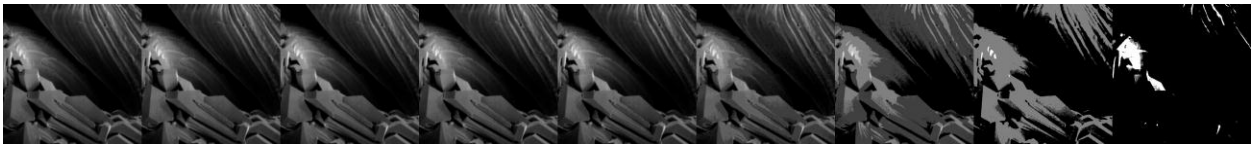

Figure 4: Reduced number of values. The left-most image is the 8-bit original. Each subsequent image has had the value depth reduced by a factor of two. The rightmost image has only two possible values, black and white.

Reducing the number of possible values has a much less pronounced effect on this image than subsampling it. Even with only 16 colors, the image fifth from the left is still almost indistinguishable from the original. This is consistent with the results shown in *Digital Image Processing*. This explains why digital cameras are digital imaging systems still use 8 bits to store pixel values, despite the low cost of digital storage—more color depth provides no benefit.

Image Pyramid

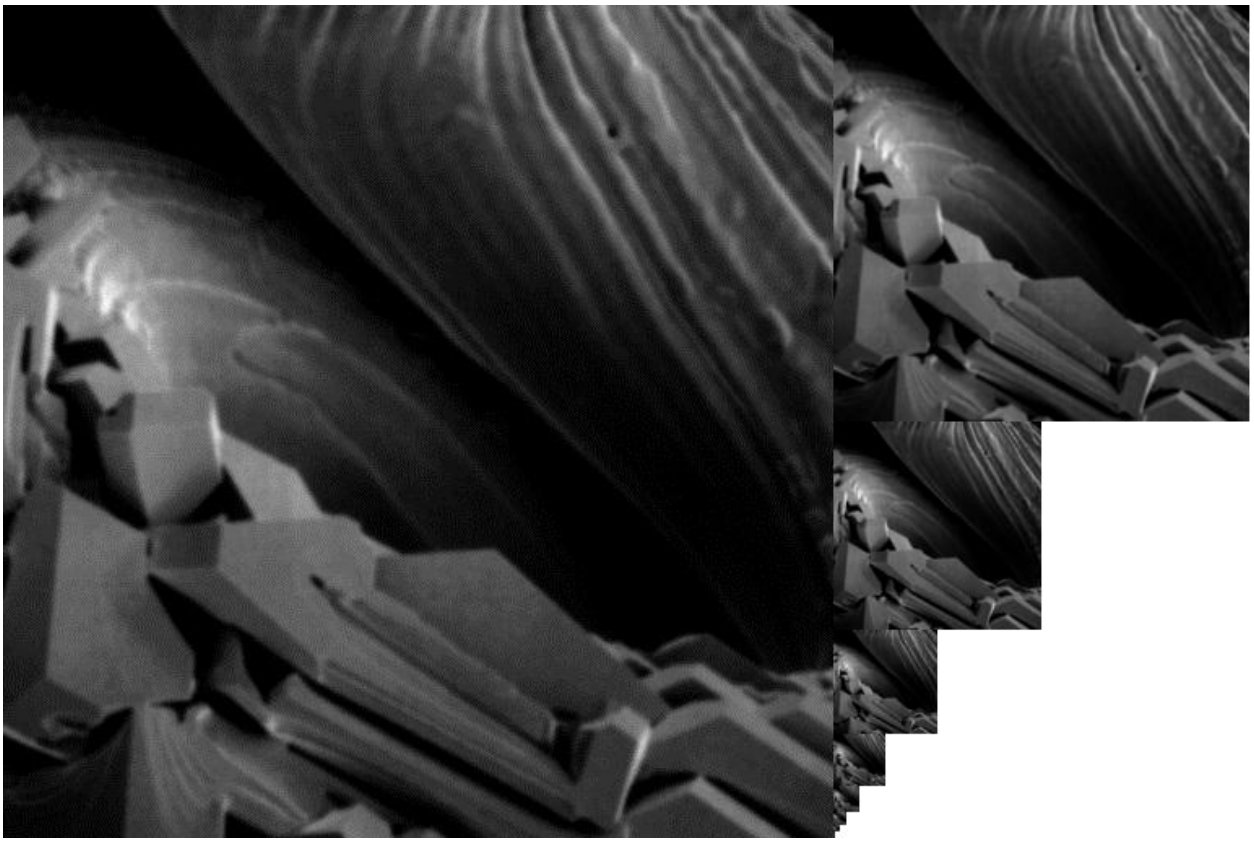

Figure 5: Image pyramid created using MATLAB

Using MATLAB's imresize() function, I was able to generate an image pyramid containing 8 images, including the original.

Appendix

All MATLAB scripts and images are submitted digitally with this report.

Scripts

load_image.m Loads the image “test.jpg” into MATLAB as global variable img_in. Also displays this image in figure 1 and stores the row and column counts in global variables rows and cols.

subsample_image.m Loads an image using load_image.m, then performs subsampling using the three methods described above. Outputs the results to figures on the screen and saves them to the hard drive.

subscale_image.m Loads an image using load_image.m, then reduces the number of gray values by factors of two. Outputs the result to a figure on the screen and saves it to the hard drive.

create_pyramid.m Loads an image using load_image.m, then uses it to create an image pyramid. Outputs the result to a figure on the screen and saves it to the hard drive.

Images

test.jpg The input image used for all scripts.

img_out_nn.png The result produced by subsample_image.m using nearest neighbor (Figure 1).

img_out_bl.png The result produced by subsample_image.m using bilinear interpolation (Figure 2).

img_out_bc.png The result produced by subsample_image.m using bicubic interpolation (Figure 3).

img_out_grays.png The result produced by subscale_image.m (Figure 4).

img_pyramid.png The result produced by create_pyramid.m (Figure 5).