Optional Project Image Processing

1)

I used a second order derivative stencil to compute the edge map of the image; a matrix consisting of the stencil along the diagonal, for a e.g.

Then, simply multiply the image by the derivative matrix, and truncate the excess pixels in the image. The rest of the code is just image processing toolbox applications.

Code and pictures attached.

2)

I used the 2nd order accurate 2nd derivative stencil from the Numerical Analysis lab, and generalized it for non-square matrices (i.e. changed the values of and such that they reflected the size of the matrix instead of a matrix that was the largest of the two.) The initial condition became the image we wished to smooth, and the diffusion constant was changed to where the edge map is obtained from the previous problem.

One of the interesting things I encountered in the beginning was that, when one used the assumption of a square matrix, didn’t convert the image to gray scale and used a low enough time step/final time, one computed a false coloring of the image based upon intensity, which gave quite the striking difference for noisy cell pictures.

The diffusion seems to add more noise to the areas around the edges, and, with the structure of the diffusion function, turns the edges to as opposed to seen in the edge detection.