Name: Caleb Yearsley

A#:

Class: CS 6680 (Computer Vision)

**Steps to run code:**

1. You can just run **main.m** for a high level summary of my project (compares images and such). All images displayed in this summary are also found in the powerpoint presentation.
2. From the Matlab console (both return images):
   1. **applyImprovedAdaptiveThresh**((<image variable>, 110); %for general use…
   2. **detectConcreteCracks**(<image variable>); %for use of concrete images
      1. Uses the **applyImprovedAdaptiveThresh** function

**Time to run code:**

**~ 0.041107** secondsfor **applyImprovedAdaptiveThresh**

**Detailed documentation for each of the components of your system:**

1. Input: either accept both an image (rgb or grayscale) and a threshold, or just an image (see commented code.)
2. Output: the functions listed above in ‘steps to run code’ will always return an image (grayscale uint8 or logical)
3. Some operations perform are: imerode, imdilate, imopen, morphological thinning, morphological clean… etc. Generally morphological operations (at least in the improved proposed approach.)
4. I measured the time, and compared the results to other methods. Compared the image results with other edge detection techniques.
5. The original proposed technique can leave noisy results. My improved proposed approach takes a little bit longer than the original proposed approach.
6. No external package was used.

**Improvements made:**

1. Much less noise was produced in the improved version

In the improved version, to clean up a lot of the noise I used a larger mask in conjunction with some morphological operations. Using a larger mask worked well as it produced thicker edge lines allowing me more flexibility with morphological operations. Performing any morphological operations (other than ‘thinning’) would dilute the image, making edges unrecognizable.

**Changes made:**

1. Can detect cracks in concrete structures a adequately than the original version

In order to make my method more suited to cracks in concrete structures, I ultimately attempted to locate the regions of concrete in the image. After locating the regions, I applied adaptive thresholding (new and improved) with a more sensitive threshold. In order to determine the regions, I calculated the probability density function to determine what was the dominate intensity level in the image. With the regions and edges queried, I then took the intersection of the edges detected and the regions.

**Any Special data structure that you have designed for solving the problem:**

Instead of using a 3x3 masked as indicated in the original proposal, I decided to use a 5x5 mask.

**Difficulties:**

I had some issues in determining the correct thresholds for the concrete detection. Also, cleaning up the workflow from the original proposal involved some work.

**Comparison:**

Ultimately, the quality of my approach contained far less noise, and adapted to concrete structures much better.

**Summary:**

**Part I:**

* The results from the published paper and my improved solution are comparable in that: they both detect edges very well, and in a timely manner. However, my improved solution cleans up noise far more affectively. To test this visually: run the following and compare the results side-by-side:



Then use **tic** and **toc** to see time efficiency.

**Part II:**

* Other experimental results would be detecting cracks in concrete structures. I tested this visually comparing it to other edges detection techniques (including but not limited to: the proposed method in the paper, and the Canny edge detection technique.) To test visually try:



and compare the results side-by-side. Then use **tic** and **toc** to see time efficiency.