EECS332 Digital Image Analysis
MP5 – Mikhail Todes

My coding is done is c++. I complied it in a bash terminal using g++. The code I used to compile and create the executable was "g++ cannyedge.cpp -o cannyedge.e `pkg-config --cflags --libs opencv`". I then ran the executable from the terminal using ./cannyedge.e.

The functions in my code are:

Mat GaussSmoothing (Mat I, int N, int Sigma);
Mat ImageGradient (Mat S);
long int FindThreshold (Mat Mag, int percentageOfNonEdge);
Mat NonmaximaSupress (Mat MagAndTheta);
float rad2deg(float radians);
Mat EdgeLinking(long int T_low, long int T_high, Mat NMS);
Mat RecursionCheck (int i, int j, Mat pixels, Mat NMS, int T_low);

The GaussSmoothing functions is used to filter to a smoother image using an NxN array and the value of Sigma. This is then convoluted over the image to make it blurred.

Once I got the smoothed version of the input image, I used the ImageGradient function to get the Magnitude and Theta for each pixel. I used Sobel to get this. The values were stored in an OpenCV Mat were the first "colour" represented the magnitude and the second "colour" represented the theta. This OpenCV Mat was a signed 16 bit format so big and negative values could be stored.

Next the Threshold was calculated using the specified percentOfNonEdges from the user in the function FindThresold. A histogram with a cumulative distribution were created to find t_high. T_low was just taken as half the value of t_low.
The function NonmaximaSupress was used to reduce the image down. It did this by checking to see which region the gradient headed to and compared the two pixels next to it in that direction to see if it was a local maxima.

Finally the edge linking function was called which calls the RecursionCheck function. The RecusionCheck function calls itself recursively until all the weak edge values between the thresholds have been linked to a high point. This leaves just a clear edge image.

The program was tested on the given images with a few different values for the parameters such as N, Sigma and percentageOfNonEdges.

The premade matlab functions were also tested against the outputs from my program.

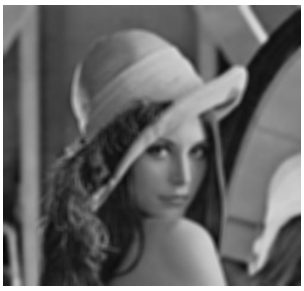

Illustration 4: GaussSmooth Lena

Illustration 2: Magnitude Lena

Illustration 3: Non Maxima Suppres Lena

Illustration 1: EdgeLinked Lena

Different N, Sigma and percentageOfNonEdges values used:

Illustration 5: N = 5, Sigma = 10, percent = 80

Illustration 6: N = 11, Sigma = 20, percent = 80

Illustration 7: N = 3, Sigma = 2, percent = 60

Edge Linking Tested on the other images:

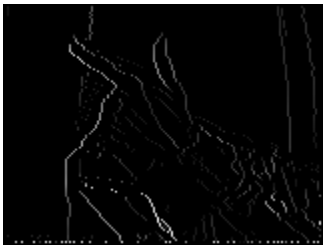

Illustration 9: Gun1

Illustration 10: Joy1

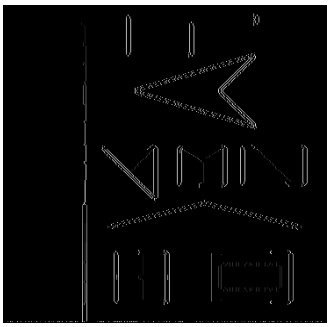

Illustration 8: test1

My program worked the worst on the test1 image.

Comparison to MATLAB functions:

Illustration 14: My program Lena

Illustration 13: Matlab Lena Roberts

Illustration 12: Matlab Lena Sobel

Illustration 11: Matlab Lena zerocross