Canny Edge Detector

Objective:

To implement the Canny edge detector and test on test images

Language used: Python

Running the code (mp05.py):

In a terminal, enter:

```
./mp05.py [image name] [sigma] [factor_to_scale_threshold]
```

Example usage:

```
./mp05.py lena.bmp 1.4 0.2
```

Description of Canny edge detector:

1. **Gaussian Smoothing:**
   - First the image is filtered using Gaussian filter with a sigma value (input by user)

2. **Calculate Image Gradient:**
   - Using the smoothed image, two edge maps are obtained using the 'Sobel' operator, namely, gradx (horizontal) and grady (vertical)
   - The magnitude of gradient is then calculated using these edge maps (hypotenuse)
   - Also, the direction, theta, of the gradient is calculated using 'arctan'

3. **Selecting Thresholds:**
   - A specific percentage of the non-edge area in magnitude of gradient is used to determine the high threshold
   - This percentage is input by user as it varies for different images
   - The low threshold is obtained as 0.5 * [high_threshold]

4. **Suppressing Nonmaxima:**
   - A non-maximum suppression algorithm is applied to the magnitude obtained in step 2
   - There are four possible directions for a given pixel – 0 deg., 45 deg., 90 deg., and 135 deg.
   - Before applying the algorithm, the directions obtained in step 2 are quantized into the above 4 angles
   - The algorithm checks the neighboring pixels' gradient (magnitude) values and if the magnitude is greater than its neighbors, it is considered an edge and marked, else, it is neglected

- The suppressed magnitude data is used together with the thresholds from step 3 in the last step below

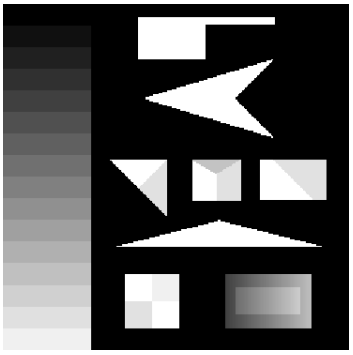
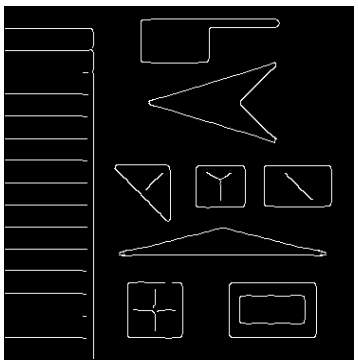
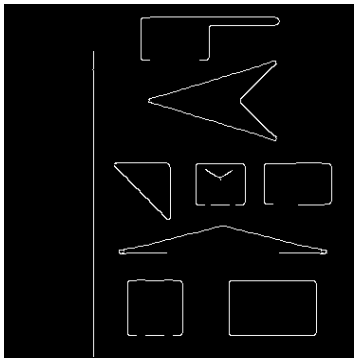
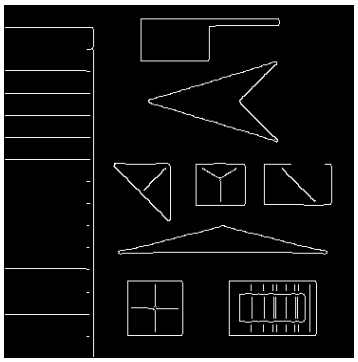
5. Edge Linking:

- In this step, using the output from step 4 and step 3 (thresholds), a recursive scanning is implemented
- The high threshold determines the start of the edge while the low threshold determines the ending of the edge, until a binary edge map is obtained
- The edges are set as white color and rest are set black

Results Analysis:

The following table summarizes the results obtained using two test images:

Image No.	Input Image	Output Image	Sigma	Threshold Factor
1 (lena)			1.4	0.2
			2.4	0.1
			1.4	0.04

2 (test1)			2.6	0.04
			2.2	0.2
			1.4	0.02
3 (joy1)			1.2	0.2
			1.2	0.04

4 (gun1)			1.8	0.12
			2.6	0.2
			1.4	0.04

All the input images produced the desired output images. It is possible to make the output images more optimal by tuning the sigma and threshold factor.

It is observed from the above results that selecting an appropriate sigma for Gaussian blur and the thresholds are critical in obtaining a ‘meaningful’ result image.