CIS581: Computer Vision and Computational Photography
Project 1 Challenges: Canny Edge Detection
Due: Sept. 26, 2017 at 3:00 pm

Instructions

- This is the additional challenge for project 1, Canny Edge Detection. Total three challenge problems and are **optional** that for the extra credit out of the project grade. We truly recommend you to review those questions and try to implement them if you have time.
- Please put all functions that contribute to challenges in the file **helper.m/py**, submit that along with the whole project zip file. It will be better to include a **README** file to tell us what you did for the challenge problems.
- **Enjoy Challenges!** If you get stuck, please post your questions on Piazza or come to office hours!

1 Local Threshold

In the edge linking section, it's necessary to set suitable low and high thresholds to figure out starter (strong) and link (weak) edge respectively. However, directly applying two threshold values to the whole image doesn't work reasonable since most of images don't have uniform gradient distribution. Therefore, setting local threshold is necessary to make your Canny Edge Detector more general to most cases.

Goal: Implement Local Threshold in the Edge Linking Section to improve the detection result.

Tips:

- Separate the whole images into several cells (windows), extract the gradient features within each unit, e.g. the sum of gradient magnitude, the max gradient magnitude, the mean gradient orientation.
- Try to build a relationship between the local gradient feature and the global whole image.
- Based on the above connection, come up with equations to compute optimal local high and low thresholds automatically for each cell.

2 Line Segment

The Line Segment detection is aimed at detect locally straight contours on images. Those contours are zones of the image where the gray level is changing fast enough from dark to light or the opposite. The object in the line segment map should contain more line information with less small or strange edge shape. That is, we focus more on the contour (global) information and filter out detail edges.

Goal: Implement the Line Segment based on the edge map obtained from designed Canny Edge Detector.

Tips: For the line segment, the basic algorithm is 'region growing' applied to line edges. The cost would be the 'mean edge orientation'. Edge orientation is computed by the filtering output (no need to round value). The region growing process is similar to a minimum spanning tree algorithm. The **pipeline** shows below:

1. Starts at a randomly picked seeds (edge pixel) to form a new seed group;

2. For each group, initialize the ’mean edge orientation’ to orientation at itself, also compute the initial standard deviation as zero;
3. Greedily ’grows’ by picking a neighboring edge (not in any groups) with the most similar edge orientation;
4. Stop growing at current direction if the new edge has very different orientation from ’mean’ or new standard deviation is beyond certain threshold.
5. Once find valid neighbors, update the ’mean edge orientation’ as well as the standard deviation, and repeat from **step 3**;
6. Finally, all edge pixel on the map should belong to one of groups, filter out those groups with less points, and use some built-in line fitting function to form line segments for the rest edge seeds groups.

3 Color Map

The binary output is kind of boring, you are recommended to generate a colorful map to represent edge detection or line segment result. The idea is pretty straightforward, extract R, G and B channels from the raw image, apply the almost same pipeline, finally, sum three results together.

Goal: Implement color map to build a colorful result, if your algorithm works well, you are supposed to get the results below that are pretty fancy than binary one.

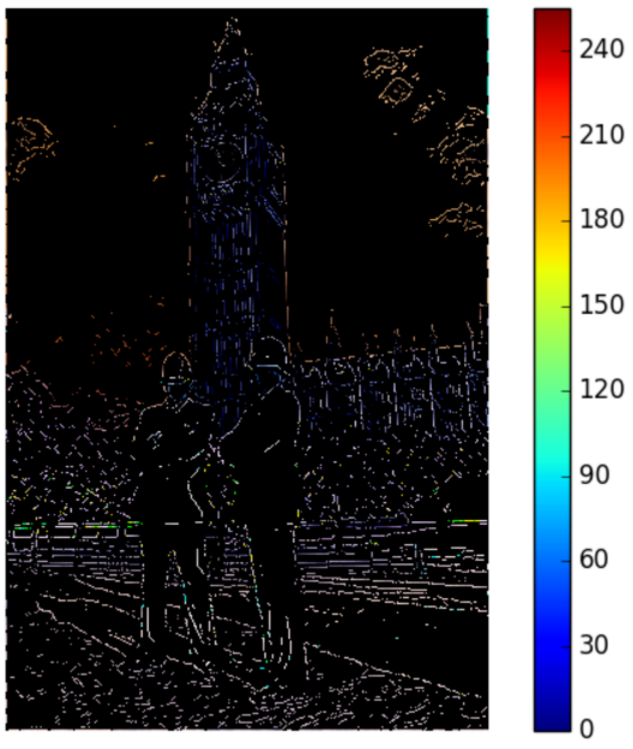

Figure 1: Color Line Segment Demo 1

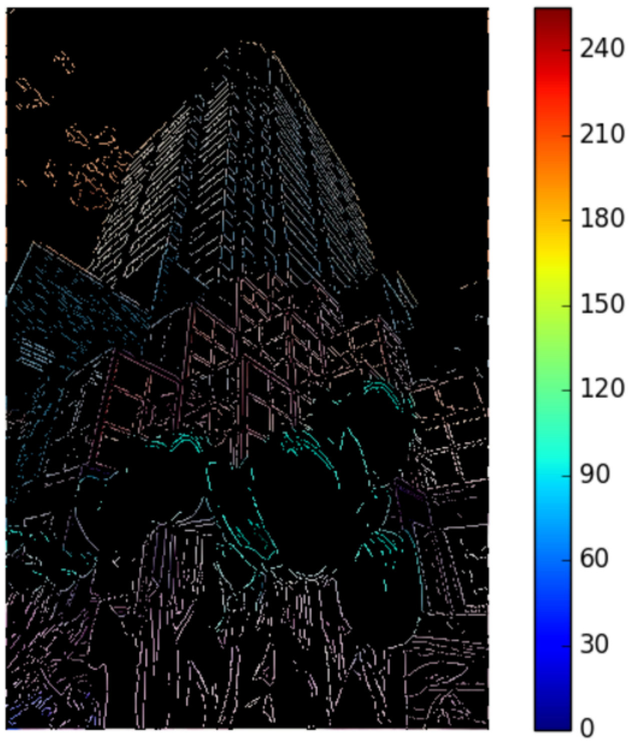

Figure 2: Color Line Segment Demo 2