CSC320 Project

Jiatao Xiang
1004236613
jiatao.xiang@mail.utoronto.ca

Huakun Shen
huakun.shen@mail.utoronto.ca

April 1, 2020

1 Introduction

Edge detection is a important tool in many computer vision applications. The major process of finding edge is to recognize sharp discontinuities in an image and mark them white whereas other places are marked black. Canny edge detection is one of the most famous algorithms that was invented by John Canny in 1986. And in this report, we will focus on how to improve Canny edge detection by modifying some steps of the algorithm.

2 Background Information

Canny Edge detection is a simple algorithm that is only consist of 5 steps. The first step is applying Gaussian filter. This is a pre-process step that can smooth out some noisy points in the image. The second step is to use Sobel filter (Gradient based filter) to extract the major edge of the image. The edge could be pretty wide. Non-maximum suppression is the third step of the algorithm. This step can thinning the wide edge, and make them one pixel wide. Next, the algorithm use threshold to separate the edges into 3 sections. They are strong pixels, weak pixels and zero pixels respectively. The final step (tracking process) is to track the weak pixels. The algorithm will keep all weak pixels that are connected to strong pixels and discard the others and assign them with zero intensity. The thinning process and tracking process provide precise edges in the result. This algorithm can also detect the edges in noisy state by applying the threshold methods.

3 Proposal

We gain a lot benefit from the Canny edge detection algorithm. However, there are also some drawbacks. We've list our concerns and possible improvements in the subsections below.

3.1 Our concerns to the algorithm

1. In the first step, we have to choose patch size and sigma for Gaussian filter to get rid of the noisy points. If the value we choose is not property, the result will be unwilling.
2. In the threshold step, the algorithm only have one high and low thresholds for the whole image, which may miss some detail that should be the edge.
3. The algorithm require a lot of computational time, which make the efficiency low.

3.2 Possible improvements

1. In stead of using Gaussian filter with user given patch size and sigma, we plan to use Gaussian pyramid instead (using reduce function when pre-process the image, and expand them back when we get the edge).
2. We plans to identify multiple high and low thresholds by dividing image into different patches. For each patch, we calculate thresholds based on the highest intensity of pixel in that patch.
3. Using reduce function multiple times to extract only the main features and after we get the edge image,

we use expand function the same number of times to make the edge image the same size as the original image.

3.3 Reason for each improvements

1.

2.

3. This reduce the image size by a lot, which significantly improves the efficiency.

4 Modelling

5 Experiment

6 Conclusion

7 limitation

8 Application