CSC320 Project

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

Huakun Shen
huakun.shen@mail.utoronto.ca

April 2, 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 or highlight something we do not want.
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. Using Gaussian pyramid, we extract the main features of the image to reduce the noise. And the modified algorithm do not require user to give parameters for the Gaussian filter.
2. We divide image into different patches and set different thresholds for each patch. In this way, we can extract main features for each patch, and in this way, the algorithm will reveal the edges that are omitted by original algorithm.
3. Using reduce function multiple times will reduce the size of image by a lot, which significantly improves the efficiency.

4 Gaussian Pyramid Experiment

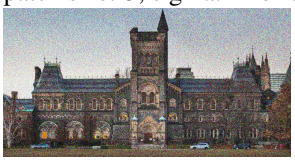
4.1 Experiment setting

Using Gaussian pyramid reduce function in the first step instead of Gaussian filter. Comparing the edges detected

from original algorithm and modified algorithm on images with different noise level and the running time of the two algorithms.

4.2 algorithm results

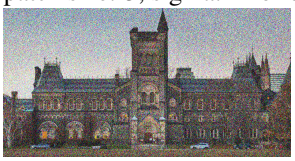
Set1:
Parameters setting:
thresholds: [0.1, 0.3] for both algorithms
patch size: 3, sigma: 1 for original cannyEdge algorithm.



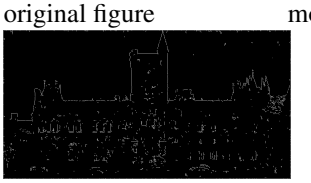

original figure modified algorithm result

original algorithm result

Set2:
Parameters setting:
thresholds: [0.1, 0.3] for both algorithms
patch size: 3, sigma: 1 for original cannyEdge algorithm.

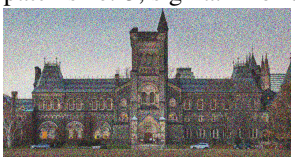



original figure modified algorithm result



original algorithm result

Set3:
Same noise level as Set2
Parameters setting:
thresholds: [0.1, 0.4] for both algorithms
patch size: 3, sigma: 1 for original cannyEdge algorithm.




original figure modified algorithm result

original algorithm result

4.3 algorithm analysis

Set1:
As we can see from the results of set1, the edge detected by modify algorithm captures more details, whereas the original algorithm only captures the outline of the building. This reveals that Gaussian filter could make some edges weak, so that those edges could not be detected by the algorithm. However, by using Gaussian pyramid reduce function, it extracts the main features from the original image and only smooth out the image by a little. This shows that if the user cannot find proper values for the parameters, our algorithm could outperform original algorithm because it doesn't affect the input image by a lot. Furthermore, finding good parameters are really time consuming because the domain of the parameters are in real number. This demonstrates that our algorithm is also better in the sense of user-friendly.

Set2:
For the second experiment, we add more noise points to the input image. From the results, we can see that the modified algorithm thinks the noisy points are also the edges due to the sharp change in intensity. However, the result of the original algorithm is pretty clear and clean, which means the original algorithm could perform well even if the images are really noisy. It doesn't mean that the modified algorithm performs bad on noisy images because how this algorithm perform also depends on the high and low thresholds that we set.

Set3:
In this experiment, we increase the high threshold to 0.4 and the result of the modified algorithm looks pretty good. This implies that even if the noise level is pretty high, we can modify threshold in order to make algorithm perform well. However, with this new threshold, the original algorithm cannot even captures the outline of the building. This experiment indicates the modified algorithm require higher high thresholds and the original algorithm require lower high thresholds. The good thing for modified algorithm is that it can captures the edge with more details, whereas for the original algorithm, it might lose those details due to the blur effect.

4.4 Conclusion

1. Modified algorithm is more user-friendly because it doesn't require user to give patch size and sigma, which is not easy to find.

2. Modified algorithm could obtain edge with more details.

3. Modified algorithm could also perform well on the images with high noise level if the thresholds are chosen properly.

4. The running time of the modified algorithm is 4 times faster than the original one because it make the image size 1/4 of the original image.

4.5 limitation

1. The edge image found by modified algorithm is only 1/4 of the original one. Even if we use Gaussian pyramid expand function to reconstruct the result, the edge is blurred.

2. The modified algorithm may perform bad if the image have a lot of sharp changes in intensity.

5 Reference