CNY5308: Introduction to Computer Vision

Report of Final Project II

Canny Edge Detection

Zheng Long	(SC11023040)
Liu Yuan	(SC11023043)
He Yangyang	(SC11023046)

Submission Date: May 15, 2012

1 Introduction

Edge detection, which detects the presence and location of edges constituted by sharp changes in colour intensity (or brightness) of an image [1], have been an essential part of many computer vision systems. The edge information is very useful for applications in 3D reconstruction, motion, recognition, image enhancement and restoration, image registration, image compression, and so on [2].

Edge detecion generally are divided into three stages [1]. First, a noise reduction process is performed. This process is usually achieved by low-pass filtering because the additive noise is normally a high-frequency signal. However, edges can possibly be removed at the same time. The second stage involves a high-pass filter such as differential operators to find the edges. In the last stage, an edge localization process is performed to distinguish genuine edges from similar responses caused by noise.

Edge detection techniques differ in their smoothing filters, differentiation operators, labeling processes, goals, computational complexity and the mathematical models used to derive them [2].

In our approach for edge detection, we implemented the classical Canny edge detector. For a given image, first convolve with a gaussian to smooth, then calculate the gradient. The direction of the gradient indicates the direction that pixel values change most abruptly, so local edge normal directions for each pixel can be estimated in this way. Then we find the edge location by searching along the edge normal directions for local maximums. For all possible locations of edges, we calculate the gradient magnitude, and through Hysteresis thresholding the edges are obtained. Finally, the results of operators with different width are synthesised via an approach called "feature synthesising".

2 Implementation Details

2.1 Main Ideas of Canny Edge Detection

The edge detection approach proposed by Canny is optimal for step edges corrupted by white noise [3]. The success of the approach depends on the definition of a comprehensive set of goals for the computation of edge points [4]. The three criterias are

1. detection criteria: important edges should not be missed and there should be no spurious responses
2. localization criteria: distance between the actual and located position of the edge should be minimal
3. one response criteria: multiple responses should be minimized to one single edge

The most important procedure is to compute the first derivative of the smoothed image in the edge normal direction $\mathbf{n}$. Non-maxium suppression and careful thresholds are designed to obtain edge pixels.

In the derivation of the optimal detector, it is found that there is an uncertainty principle relating detection and localization of noisy step edges, and that there is a direct tradeoff, which can be varied by changing the spatial width of the detector, between the two. Specifically, a filter with a broad impulse response will have better signal-to-noise ratio than a narrow filter when applied to a step edge, which means it has a better dectection performance. On the contrary, a narrow filter will give better localization than a broad on. Because of this tradeoff, image edges are detected by operators with different scales.

Edges detected by different operators are combined by a "feature synthesis" approach in the last procedure to obtain good performance in both detection and localization.

2.2 Implementation

The overview of the steps to implement classical Canny edge detector is listed below [3]. The details will be explained in the following paragraphs.

1. Convolve an image f with a Gaussian of scale σ.
2. Estimate local edge normal directions $\mathbf{n}$ using equation 2-1 for each pixel in the image.
$$\mathbf{n} = \frac{\nabla(G * f)}{|\nabla(G * f)|}. \tag{2-1}$$
where G is a 2D Gaussian.
3. Find the location of the edges using equation 2-2 (non-maximal suppression).
$$\frac{\partial^2}{\partial \mathbf{n}^2} G * f = 0. \tag{2-2}$$
4. Compute the magnitude of the edge using equation 2-3.
$$|G_n * f| = |\nabla(G * f)|. \tag{2-3}$$
where G_n is given by
$$G_n = \frac{\partial G}{\partial \mathbf{n}} = \mathbf{n}\nabla G. \tag{2-4}$$
5. Threshold edges in the image with hysteresis to eliminate spurious responses.
6. Repeate steps (1) through (5) for ascending values of the standard deviation σ.
7. Aggregate the final information about edges ate multiple scale using the 'feature synthesis' approach.

As mentioned in the previous section, operators of different scales are applied to the image to obtain good performance both in detection and localization. Our choice of the standard deviation σ and operator width is based on the tutorial in [5]. The choice is listed in the following table.

Standard Deviation σ	Size of Mask
0.5	3×3
1	5×5
2	9×9
3	13×13
4	19×19

Table 1: Choice of σ and operator width

As for non-maximal suppression in the 3rd step, we tried two approaches. The first approach uses equation 2-2. To find the edge locations by directly dectecting zeros in the resulting second derivative image will inevitably fail, since in discrete case, the position correspondes to the local maximum will not exactly have a zero derivative. In our implementation we use the method in [3], which identify a zero crossing in a moving 2×2 window. If both polarities occur in the 2×2 window, mark the upper left corner as a local maximum. While this approach generates resonable results, it's not satisfying in that the 2×2 window involves inadequate pixels, thus the resulting local maximums will be easily affected by noise. Moreover, by randomly marking the upper left corner as the local maximum actually lowers the performance of localization.

On the contrary, the second approach yields a much more satisfying result, and therefore is adopted in the system. It first quantizes edge direction in each pixel into four main directions: west and east, north and south, north west and south east, north east and south west. Then compare magnitudes with the two pixels picked from the corresponding 8-connectivity along the normal edge direction. If the pixel has a greater magnitude than both of neighbouring pixels, it is marked as local maximum. The detailed implementation can be found in [5].

In order to implement the previous non-maximal suppression approach, we need to compute the edge magnitude in step 4 before step 3. The edge magnitude is actually gradient magnitude of the smoothed image.

The following step is hysteresis thresholding. It uses two thresholds: a higher one, t_h, and a lower one, t_l. Pixels with edge magnitude greater than t_h are marked as edge immediatly. And those with magnitude lower than t_l are marked as non-edge. For those between, check for its neighbourhood, 4- or 8-connectivity, if there is any edge pixel, mark it as edge, too. The challenging part here is the selection of thresholds.

To find the thresholds, we use the self-adpative threshold based on Otsu referred in [6] for efficiency and simplicity.

After edges are found by operators with different scales, the next step is to combine the results using “feature synthesis”. The synthesis approach involved in our system is rather simple. When combine results of a narrow operator and a broad operator, the result of the narrow one are first convolved with a Gaussian in the normal edge direction. If the result of the broad operator has responses that the smoothed result of narrow operator doesn’t have, mark them as edge in the combined result.

3 System Outcome

3.1 Experiment Results

The experiment is mainly done on the test image Lenna. Figure 3.1 shows the original edge magnitude and the edge magnitude after non-maximal suppression. Figure 3.2 to Figure 3.4 shows the edges obtained by operators with different scales and the final combined edges.

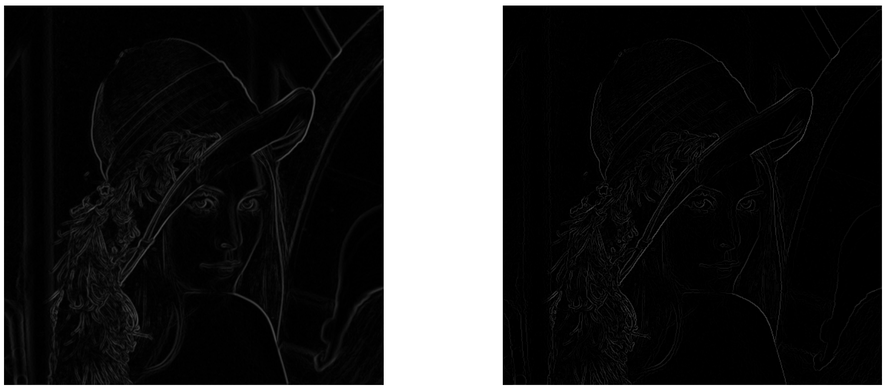

Figure 3.1: Demostration of non-maximal suppression. Left is the original edge magnitude, and right is the edge magnitude after non-maximal suppression.

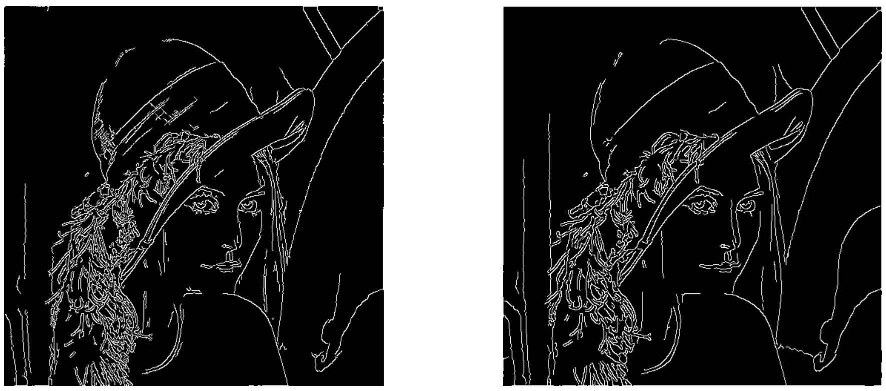

Figure 3.2: Left is the edges found by $\sigma = 0.5$, right by $\sigma = 1$

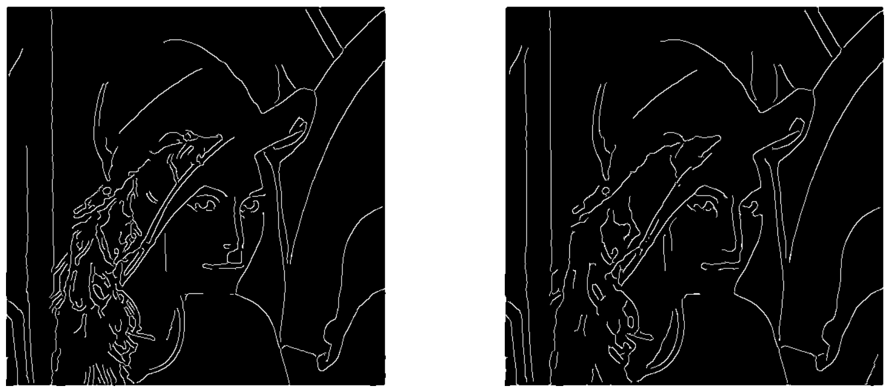
Figure 3.3: Left is the edges found by $\sigma = 3$, right by $\sigma = 4$

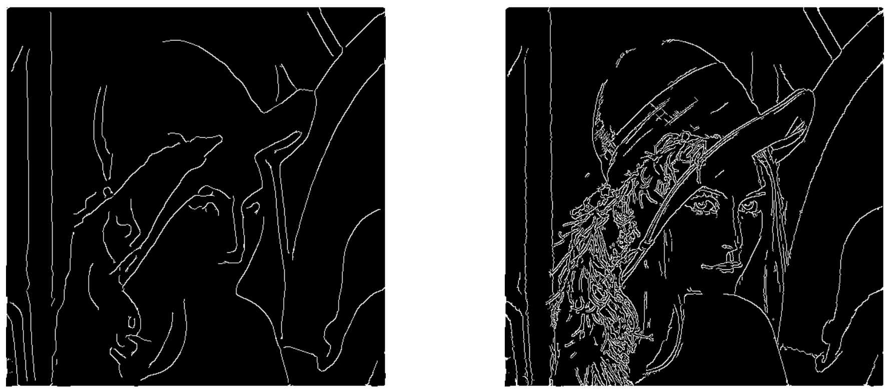
Figure 3.4: Left is the edges found by $\sigma = 5$, right is the final combined edge image

3.2 Outcome Discussion

From Figure 3.1 we can see that the edge magnitude after non-maximal suppression has been thinned to one-pixel thick, which means the approach we chosed really works well.

From Figure 3.2 to Figure 3.4 we can see that when the σ and operator width becomes larger, the resulting edge image will have less details. This is reasonable because when the operator becomes larger, the image is smoothed heavier, thus more details are lost. But there will some edges that cannot be detected by narrow operator. This is because filter with broader impulse response will have better signal-to-noise ratio [4].

The final combined result in Figure 3.4 presents both details and basic skeleton, and all the edges are one pixel thick, which is pretty satisfying.

3.3 Lessons Learned

When dealing with selecting thresholds for the hysteresis thresholding process, we first planned to set them manually. But it is fraustrating since good results only come from trying, and good thresholds in this case may not be good in another case. So it will be great to have some systematic way to solve the adaptive thresholds. The approach we adopted in this system really satisfys this requirement wonderfully. So the lesson is, a better understanding or a deeper interpretation of the problem usually yields nicer results.

4 Summary

In this project, we implemented Canny edge detector. Operators with different scales are applied to the image to cope with different signal-to-noise ratio for each edge. The edge magnitudes are computed as the magnitude of gradient. An edge direction quantization process is performed to do non-maximal suppression of the edge magnitudes. In the hysteresis thresholding, a self-adpative threshold based on Otsu is applied. Finally, results of different operators are combined using “feature synthesis”.

In retrospect, when tackling the project, we were facing with many problems: what was the role of the direction of gradient? How to select thresholds for hysteresis thresholding? How to combine results of operators with different scales? Why was it necessary to use different operators? How did the size of the operators affect the final result? Fortunately, after days of tortures, the problems are solved and a handsome edge detector come into our sight.

This course has taken us to a wonderful world of computer vision. When facing the matrices of pixel values, we are not, as in the past, that uncertain about what we can really do with them. Compute vision is sort of a magical power that can yield surprisingly nice results from those ugly pixel values.

References

[1] Mohammadreza Asghari Oskoei and Huosheng Hu. A Survey on Edge Detection Methods. Technical Report February, University of Essex, 2010.

[2] D Ziou and Salvatore Tabbone. Edge Detection Techniques - An Overview. *International Journal of Pattern Recognition and Image Analysis*, 8(4):1–41, 1998.

[3] Milan Sonka, Vaclav Hlavac, Roger Bolye. *Image processing, analysis and machine vision*. Nelson Engineering, 3 edition, April 2007.

[4] John Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[5] Canny's edge detector: Implementation. `http://suraj.lums.edu.pk/~cs436a02/CannyImplementation.htm`.

[6] Yuan-kai Huo, Gen Wei, Yu-dong Zhang, and Le-nan Wu. An Adaptive Threshold for the Canny Operator of Edge Detection. *Science*, (1):2–5, 2010.