Lecture Note for Edge Detection

18340013 Conghao Chen

Abstract

This lecture note discusses the related content of edge detection. At first, it introduces the basic mathematical knowledge that the edge detection needs to use. Then this note makes clear the concept of edge detection and when to use it, and puts emphasis on analyzing several methods in edge detection. After discussing the theoretical knowledge, I carry out a comparative experiment, analyze the experimental results for verifying the theorem whether it is correct or not. Finally I put forward some my ideas for improvement.

Figure 1. Original Picture

Figure 2. After Edge Detection

1. Theoretical Introduction

1.1. Brief

Edge detection is a basic problem in image processing and computer vision. The purpose of edge detection is to identify the points with obvious brightness change in digital image. The significant changes usually reflect the important feature of images. Edge detection greatly reduces the amount of data, eliminates the irrelevant information, and retains the important structural information of the image. It is important to note that edge detection is only for grayscale images, so we need to convert the picture to grayscale image first. Here are an original picture and the picture after edge detection. By visualizing it, we can see what edge detection is doing intuitively.

1.2. Mathematical Tools

In order to better understand the algorithm about edge detection, I need to introduce the

concept of derivative and gradient on the next.

As we all know, the derivative of a function f in 1D is as follows:

$$\frac{df}{dx} = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x)}{\epsilon}$$

Edge detection is generally based on grayscale image which is two-dimensional at this time, so let's take a look at the partial derivatives of a function f in 2D:

$$\frac{\partial f}{\partial x} = \lim_{\epsilon \to 0} \frac{f(x+\epsilon, y) - f(x, y)}{\epsilon}$$

$$\frac{\partial f}{\partial y} = \lim_{\epsilon \to 0} \frac{f(x, y+\epsilon) - f(x, y)}{\epsilon}$$

Based on derivative, we can give the concept of gradient. Gradient is a vector which consists of partial derivatives. That is:

$$\bigtriangledown f(x, y) = (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}) = (f_x, f_y)$$

Meanwhile, gradient magnitude and gradient direction are as follows:

$$|\bigtriangledown f(x, y)| = \sqrt{(f_x)^2 + (f_y)^2}$$

$$arctan \frac{\partial y}{\partial x} = arctan \frac{f_x}{f_y}$$

A function changes the fastest along the gradient direction and has the highest rate of change. This is the meaning of gradient. In practical application of edge detection, we use difference to approximate derivative, so we have the following formulas approximately:

$$\frac{\partial I}{\partial x} = I(x+1, y) - I(x, y)(\epsilon = 1)$$

$$\frac{\partial I}{\partial y} = I(x, y+1) - I(x, y)(\epsilon = 1)$$

1.3. Related Algorithm

The key of edge detection is convolution. In my opinion, convolution is similar to inner-product of vector, both of which multiply the corresponding element and sum them up. When convoluting a pixel of an image, place the center of the convolution kernel on the pixel, calculate the product of each element and the overlapping pixel, and sum them up to get new pixel values.

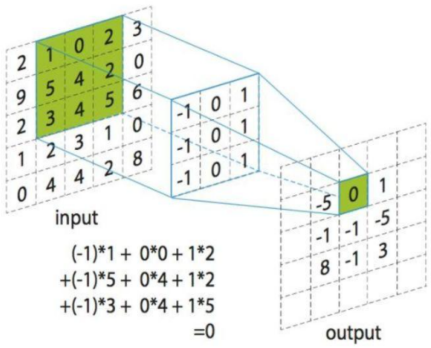

Figure 3. Convolution

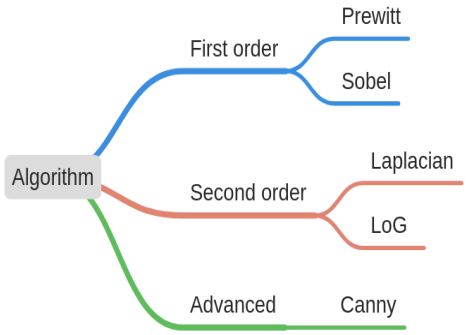

Figure 4. Related Algorithm

We call convolution kernel as operator in edge detection. Different convolution kernels correspond to different algorithms. Most of them can be divided into two categories: the first order or the second order. The first order method detects the edge by finding the extremum values of the first derivative of the image. As for the second order, obviously the edge is detected by zero-crossing of the second derivative. Commonly used algorithms are

listed in Figure 4. I will introduce these algorithms one by one.

1.3.1 Robert

Robert operator is a 2x2 convolutional kernel, which uses the two adjacent pixels in diagonal direction of the middle point. In Robert operator, we have the following formulas:

$$\frac{\partial I}{\partial x} = I(i+1, j+1) - I(i, j)$$

$$\frac{\partial I}{\partial y} = I(i+1, j) - I(i, j+1)$$

From above, we can get Robert operator:

$$G_x = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

Then, we need to overlay in two directions and get the final result. Robert operator locates more accurately, but it's sensitive to noise.

1.3.2 Prewitt

Prewitt operator is a kind of first-order differential operator for edge detection. It estimates the gradient of the middle pixel using the pixel of eight neighboring points. It can detect edges, remove some pseudo-edges, also has smooth effect on noise compared to Robert operator. In Prewitt operator, we have:

$\frac{\partial I}{\partial x} = I(i-1, j-1) + I(i, j-1) + I(i+1, j-1) - I(i-1, j+1) - I(i, j+1) - I(i+1, j+1)$

$\frac{\partial I}{\partial y} = I(i-1, j-1) + I(i-1, j) + I(i-1, j+1) - I(i+1, j-1) - I(i+1, j) - I(i+1, j+1)$

From above, we can get Prewitt operator:

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

1.3.3 Sobel

Compared with Prewitt operator, Sobel also uses the surrounding 8 pixels to estimate the gradient of the middle pixel, but Sobel operator thinks that the points close to the middle points should be given higher weight, so Sobel operator sets the weight of the 4 pixels which adjacent to the middle pixel to 2 or -2. So the Sobel operator is as follows:

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

For Prewitt operator and Sobel operator, the former is an average filter, the latter is a weighted average filter. Both of them have a good effect on the image with gradual change of gray level and low noise, but for images with complex noise, the processing effect is not ideal. When the accuracy requirement is not very high, we can use Prewitt operator or Sobel operator.

1.3.4 Laplacian

From above, we can conclude that the edge is the local maximum of the first derivative of the image, which means that the second derivative of the point is zero according to the knowledge of calculus. This is the principle of Laplacian operator. The formulas of Laplacian operator are as follows:

$$\frac{\partial^2 I}{\partial x^2} = I(x, y+1) - 2I(x, y) + I(x, y-1)$$

$$\frac{\partial^2 I}{\partial y^2} = I(x+1, y) - 2I(x, y) + I(x-1, y)$$

From above we can get Laplacian operator for fourfield:

$$G = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

By extending it, we can get Laplacian operator for eightfield:

$$G = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Because zero-crossing, isolations or endpoints are more sensitive, so Laplacian operator is especially suitable for highlighting the isolations, lines, or line endpoints of an image. However, it responds more to noise. Under this circumstances, we can process the image with Gaussian smoothing at first, then carry out convolution with Laplacian operator. This method is called Laplacian of Gaussian(LoG).

Gauss smoothing is a 2D convolution operation applied to blurred images to remove detail and noise. We know Gauss function and its second order gradient(i.e. divergence) according to probability theory and calculus(note: omitting coefficient):

$$G_\sigma(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$\begin{aligned} LoG &= \frac{\partial^2 G_\sigma(x,y)}{\partial x^2} + \frac{\partial^2 G_\sigma(x,y)}{\partial y^2} \\ &= \frac{x^2+y^2-2\sigma^2}{\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}} \end{aligned}$$

Figure 5. The divergence of Gauss function

It is clear that the edge of the image can be obtained by detecting the zero-crossing of the filtering results from Figure 5.

After calculation we can get the convolution kernel:

$$G = \begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

We can use the convolution kernel above to convolute the image:

$$g(x, y) = [\bigtriangledown^2 G(x, y)] I(x, y)$$

However, the calculation quantity of this method is large and the efficiency is low, so the second method is generally adopted. Because convolution is a linear operation, it can also be written as follows:

$$g(x, y) = \bigtriangledown^2 [G(x, y) I(x, y)]$$

Because of Gaussian smoothing, this algorithm can overcome the influence of noise to some extent. But false edges may occur and the precise may decrease for some curved edges.

Although the LoG algorithm still has the above shortcomings, it plays a positive role in image feature extraction. In particular, the idea of Gaussian smoothing before gradient calculation, was later adopted by Canny method.

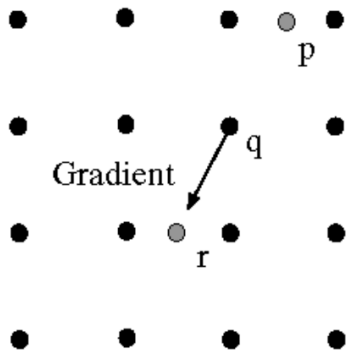

Figure 6. Example

1.3.5 Canny

Canny puts forward a set of standards to evaluate the edge detection operators:

First, high SNR. The higher SNR value, the better the edges detected.

Second, precise positioning. The distance between the detected edge and the actual edge should be as small as possible.

Third, a clear response. There is only one response to each edge and only one point is obtained.

Canny operator has two more steps than other algorithms: non-maximum suppression and hysteresis thresholding. I will briefly describe these two steps below.

(i)Non-maximum suppression:

The key of this algorithm is to find the local maximum of a pixel point and set the gray value corresponding to the non-maximum point to 0, it will remove a large part of the non-edge points. Here I'll explain how this algorithm works with Figure 6.

In this figure, the gradient direction of q is marked. From the knowledge of calculus, we know that the local maximum must located in this line, that is, p or r may be a point that reaches the local maximum in addition to q. So We need to judge whether the gradient of q is greater than p and r or not. If not, q is not the local maximum point, that is to say, q is not edge. This is how non-maximum suppression works.

(ii)Hysteresis thresholding:

Due to the influence of noise, interruption often occurs at the edge which should be continuous. Hysteresis thresholding sets two thresholds: one is high threshold T_H, and the other is low threshold T_L. The whole process is described as follows:

If the gradient value of the pixel is less than T_L, the pixel is a non-edge pixel;

If the gradient value of the pixel is greater than T_H, the pixel is an edge pixel;

If the gradient value of the pixel is between T_L and T_H, we need to further detect its eight-field, if more than one point has the greater gradient value than T_H, the pixel is an edge pixel, otherwise it is not an edge pixel.

Canny method is not easy to be disturbed by noise and have the ability to detect real weak edges. The advantage is that two different thresholds are used to detect the strong edge and the weak edge respectively, and when the

weak edge and the strong edge are connected, the weak edge is included in the results.

2. Comparative Experiment

2.1. Experimental Configuration

In this experiment, I use opencv-python and PIL package to implement above algorithms. For Robert, Sobel, Prewitt and Laplacian, the key step which is convolution is implemented by myself. I also compare the Canny algorithm in opencv package with the Canny algorithm which implemented by myself without packages. The input picture I used is below:

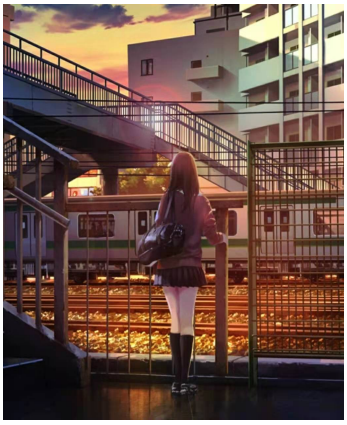

Figure 7. Input

2.2. Experimental Result

Here are results after edge detection:

Figure 8. Robert

Figure 9. Prewitt(vertical)

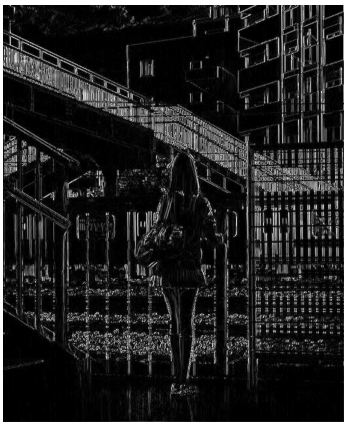

Figure 10. Sobel(horizontal)

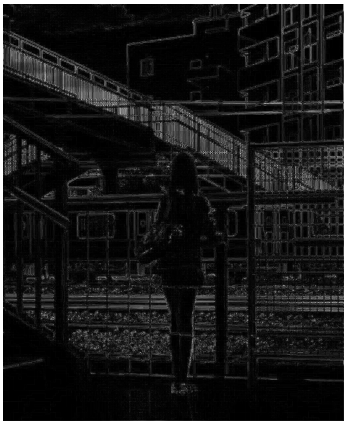

Figure 11. Laplacian(eightfield)

Figure 12. LoG of opencv

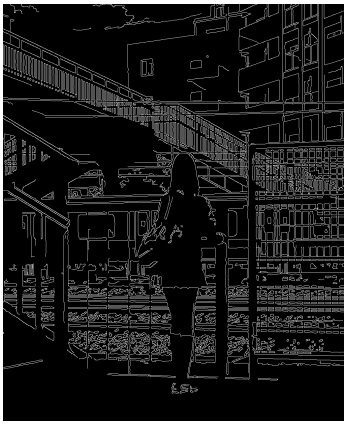

Figure 13. Canny by myself

Figure 14. Canny by opencv

I use an API which called compare_psnr of skimage package to get the PSNR(Peak Signal Noise Ratio) are as follows:

Method	PSNR
Robert	9.557
Prewitt	9.360
Sobel	9.466
Laplacian	9.663
LoG	9.993
Canny(opencv)	9.973
Canny(myself)	9.397

From the result, we can see that the Canny of OpenCV is better than the Canny implemented by myself apparently, reflected in the clearer edges. The result of Sobel is better than Prewitt, we can see that the picture of Prewitt is vague and hasn't hierarchy visually, which comply with theory we mentioned. The PSNR of LoG is the greatest, so we can conclude that LoG is the best method for this input picture.

2.3. Improvement

Here I give my view on how to improve Canny. In hysteresis thresholding, we need to select two thresholds T_L and T_H, but how to select them is a problem. We usually adjust these two thresholds manually. I think we can find a way to get more information about edges and then use statistical methods to pick up two

thresholds. [1] Also, we use Gaussian smoothing at the beginning of Canny. But noise is a high frequency signal and edge is also a high frequency signal. It causes that Gauss smoothing doesn't distinguish all the high frequency information, so the effect may be not very good. I think we can design a method instead of Gauss smoothing to distinguish the high frequency. [2]

3. Summary

This note mainly introduces the related content of edge detection, focuses on several algorithms of edge detection, and makes a comparative experiment to verify the related theory. As we can see, each algorithm has its advantages and disadvantages, which also determines that they have different applicable scenes. We also note that the steps of the Canny algorithm are more complex due to taking into account more factors which makes the Canny algorithm one of the most typical algorithms. Finally, I give my immature ideas on how to improve Canny algorithm in theory. Of course, edge detection is a very deep research direction. It is impossible to say that I have mastered this direction completely during this period of time, but I can say that I have learned a lot. That's all what I understand about edge detection.

References

[1] 朱秋林、石银涛、李靖. 一种改进型 canny 算子边缘检测算法. 地理空间信息, 018(001):128–133, 2020. 8

[2] 胡松、黄志远、邓磊、李壮豪、范洁滢、曾维. Canny 算子在图像处理中的优化研究. 计算机技术与发展, v.30;No.282(10):112–116+215, 2020. 8