Computer Vision: Lab 1 Report
Canny Edge Detection

Zhuoyang Jiang, 58119125

I. Experimental Objectives

In this lab we need to implement the complete steps of the function of canny edge detection. The specific objectives are as follows:

A. Gray Treatment

Turn the input color image into a gray image, in order to do further work.

B. Gaussian Filter with Derivative

Calculate image gradient with the Gaussian Derivative Filter.

C. Non-Maximum Suppression

Implement the Non-Maximum Suppression on the image, in order to tighten up the width of edgers into single pixel.

D. Hysteresis Thresholding and Linking

Set two threshold, a higher one and a lower one. Judge the edge with the higher one and link them with the lower one.

E. Finally Achieve the Goal of Canny Edge Detection

After Hysteresis Thresholding and Linking, we get the result of canny edge detection.

II. Process Analysis

A. Image Import and Gray Processing

Import a color image and gray it, in order to refine the image information. We can get a gray image comparing with the origin image as Fig.1:

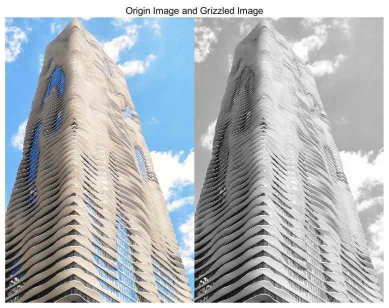
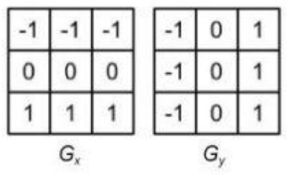


Fig. 1. Origin Image and Grizzled Image

B. Gaussian Filter with Derivative

1) Coordinate Positioning Matrix: At the very beginning, we need to construct the positioning matrix in X and Y directions.

We can use the Prewitt filters to achieve it.Here we define two 3 * 3 coordinate positioning matrix (Prewitt Maticx) in X and Y directions (Pay attention to row and column representation)



-1	-1	-1
0	0	0
1	1	1

G_x

-1	0	1
-1	0	1
-1	0	1

G_y

Fig. 2. Prewitt Matrix

2) Gaussian Derivative Filter Construction: Than we can define Gaussian Derivative Filter.

As we know, both the step of derivative and the step of Gaussian filter need to be implemented on the image, and they can be synthesized into one step.

So that, after customizing Gaussian parameters to make the window width fit Sigma, we can synthesize the two steps. The formula method is used to calculate the first derivative of each element position `(use .* ; ./ operation in matlab)` of the Gaussian kernel.

$$\frac{\partial G_\sigma(x,y)}{\partial x} = (-\frac{1}{2\pi\sigma^4})xe^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$\frac{\partial G_\sigma(x,y)}{\partial x} = (-\frac{1}{2\pi\sigma^4})ye^{-\frac{x^2+y^2}{2\sigma^2}}$$

After that, we get the Gaussian derivative filters in two direction.

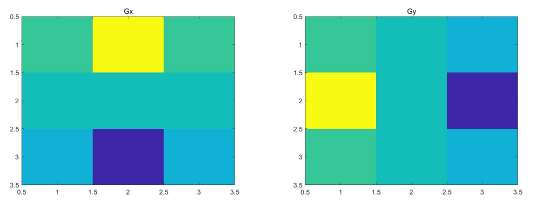


Fig. 3. Filters in Two Direction

C. Image Gradient Calculation

In this step, we use the filters we got in the previous step to calculate the gradient information of the image.

1) Correlation: We use correlation method to filter the image with our filters in two direction.

Now we have calculated the Gradient Magnitude and Gradient Direction.

2) Edge Strength: With Gradient Magnitude, we can calculate the edge strength on every pixel:

$$\|\nabla I\| = \sqrt{(\frac{\partial I}{\partial x})^2 + (\frac{\partial I}{\partial y})^2}$$

Then we can show the edge strength of the image:

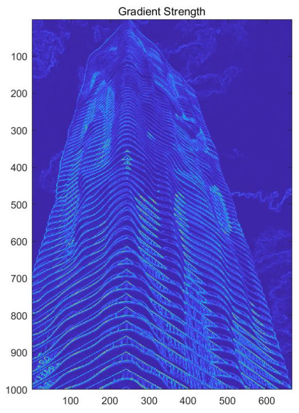


Fig. 4. Edge Strength of Image

3) Direction Angle: With Gradient, we can calculate gradient Direction on every pixel:

$$\theta = tan^{-1}(\frac{\partial I}{\partial x}/\frac{\partial I}{\partial y})$$

In matlab we use atan2() function to calculate it.

4) Direction Adaptation: In particular, I use a simplified way to replace Interpolation Method.

We need to 'Adapt' the gradient direction to eight directions `(up/down/left/right/2Diagonal)`, which is convenient for non maximum suppression later. Using the rounding function, the angle in the gradient direction is changed to the nearest multiple time of $\frac{\pi}{4}$.

D. Non-Maximum Suppression

After defining the new blank image or we call it storage matrix, we can do NMS.

We need to scan every pixels of the image and compare it with its two neighbours on its gradient directions to decide if it is the Edge Strength Maximum pixel.

So our first problem should be 'How to compare with the two neighbours of a pixel'.

1) Neighbour Pixels Position: As we have done in the previous step, we adopt the gradient direction to eight directions. This is a simplified way to replace Interpolation Method.

So, according to the eight values of the normalized gradient direction, we can simply discuss the positions of the two neighbour pixels ("comparison points") to of every pixel.

TABLE I
POSITION DISSCUSSION

Distance	-1	0	1	condition
-1	$-3\pi/4$	π	$3\pi/4$	$\|\theta\| \geq 3\pi/4$
0	$-\pi/2$	/	$\pi/2$	'else'
1	$-\pi/4$	0	$\pi/4$	$\|\theta\| \leq \pi/4$
condition	$\theta \leq 0$	'else'	$\theta \geq 0$	

2) Gradient Magnitude Comparison: After the discussion of the location between the current pixel and its neighbours, we compare their Gradient Magnitude.

If the current pixel's Gradient Magnitude is larger then its both neighbours, we keep it. Or, we assign its value to zero.

3) Get NMS Result: So that, we get the NMS result of the image:

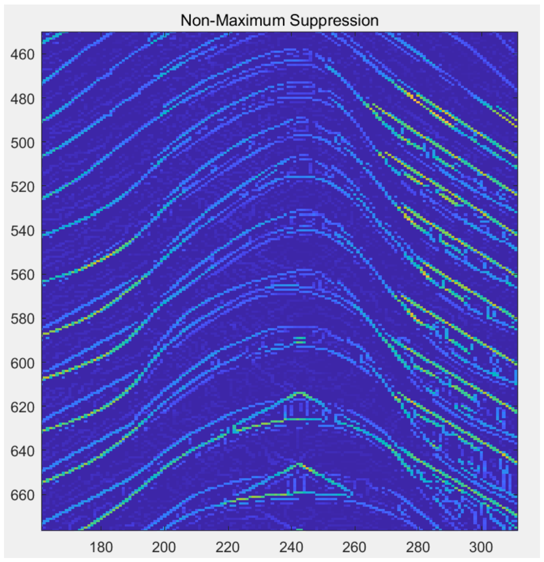


Fig. 5. Part of the NMS image result

On the NMS image, we can easily find that most edges' pixel become single.

E. Hysteresis Thresholding and Linking

Set two threshold, a higher one and a lower one. Judge the edge with the higher one and link them with the lower one.

1) Hysteresis Thresholding: Set two thresholding, which can be adjusted to reach a better effect.

2) Comparing to Find Main Edges: With the NMS result, we define two matrix to store the result of comparison of high threshold and low threshold.

The comparison results will be 1 and 0, which mean larger and not larger.

On this step, we have actually get the main edges with the storage of pixels higher than the high threshold.

3) Scanning to Link Left Edges: Than we use the result of two matrices storing the pixels higher than the threshold to link edges.

From a high-threshold pixel, we link its neighbour if this neighbour has been stored in the low-threshold matrix.

So we need solve 3 problems:

- Find neighbours
- Judge
- Link.

To find neighbour, we still use the Table.1 to get distance which can locate to the neighbours.

To judge, we can put the condition at the beginning of every scan time of every pixel. The condition is simple: 'Whether the pixel's value in high-threshold matrix is 1'. If the condition is true, than we do the link work.

To link, we assign the pixel's value in the low-threshold matrix to the same pixel in the high-threshold matrix.

F. Finally Achieve the Goal of Canny Edge Detection

Finally, the result of the high-threshold matrix after scanning and linking is the target result.

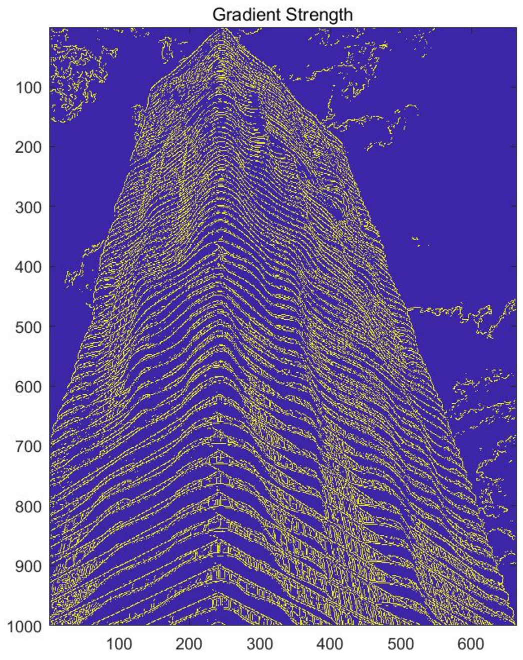


Fig. 6. My Canny Edge Detection Result

III. Results Description and Summary

A. Comparison to The Standard Canny Algorithm

We can apply the standard Canny edge detection to the same image comparing with my own canny Application.

Then, we get the result as follow:

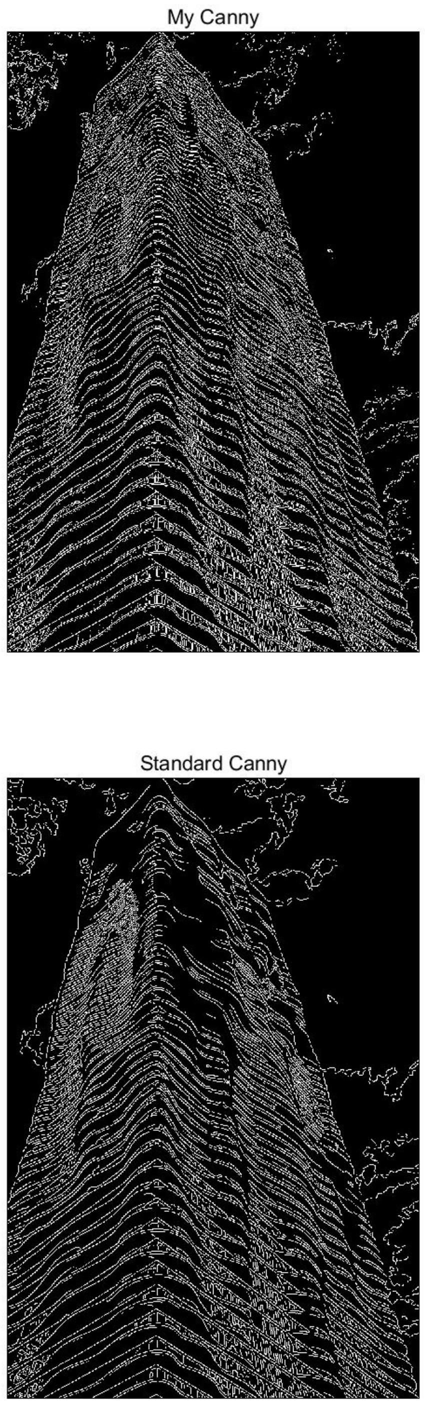


Fig. 7. Comparison of Canny Edge Detection

As we can illustrate, my Canny may detect more miscellaneous points and also more complete on some vague edges. And this is both related to the setting of Threshold and the Interpolation mode I simplified.

B. Threshold Adjustment and Effect

So I first analyse the setting of Double Threshold.

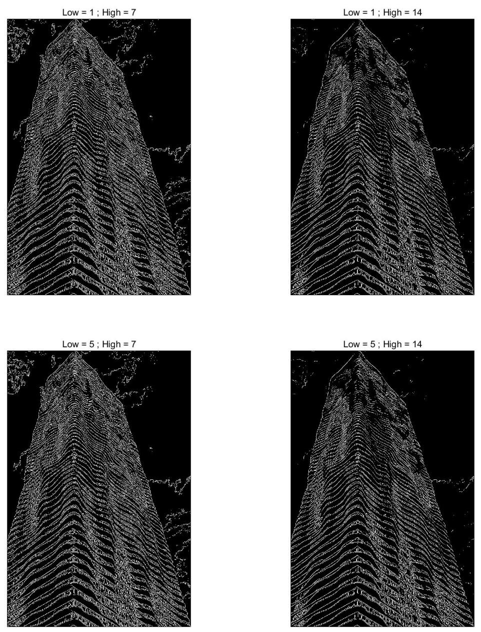


Fig. 8. Comparison of Threshold

We can draw a conclusion:

- The high threshold determine to the entirety of the edges.
- The low threshold determine to the Locality of the edges.

To show the detail, we compare a more exaggerated pair.

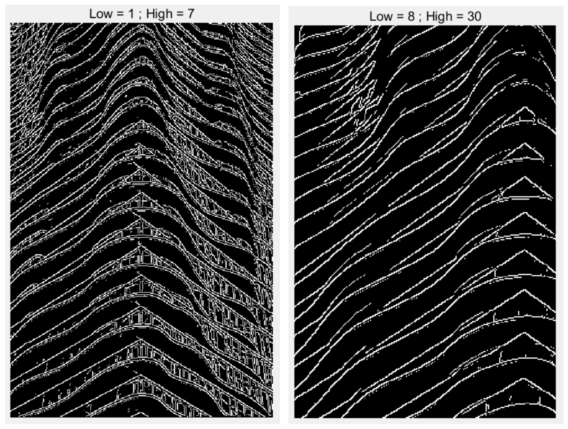


Fig. 9. Comparison of Detail

C. Different Kind of Images

The result is also determined by the complexity or we can call it ’Relative Clarity’.

The more it is complex with the same amount of pixels, the higher its Relative Clarity is, so that, it will be harder for detector to detect the edges.

The main reason of it is the difficulty of gradient processing.

What’s more it is also dealing with my simplification of the interpolation method.

Let’s see the image drawn by line.

Obviously it is more easy to detect edges for my Canny detector.

Fig. 10. Simpler Image

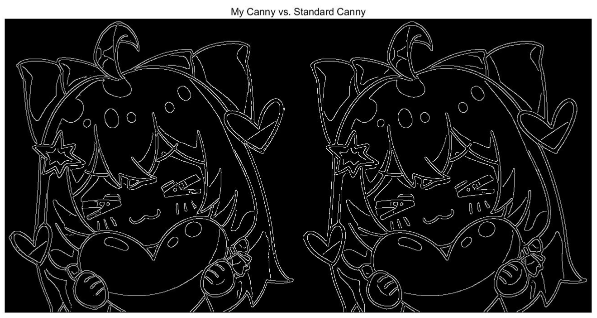


Fig. 11. ED of Simpler Image

D. Improvement Ideas

As we concluded, my simplification of the interpolation method may influence the quality of edge detection.

We can do Refinement on the way of interpolation, so that we can get a more fine gradient calculation which determines a more fine result.

E. Summary

In this Experiment, I have learned a lot:

- Learn and apply the whole process of Canny Edge Detection.
- Familiar with the application of matrix transformation in image processing.
- Understand the relevant knowledge of image smoothing and image gradient calculation.
- Proficient in computer vision code debugging ideas.

All in all, the experiment is finally completed.

Supporting Material: My Code in Matlab

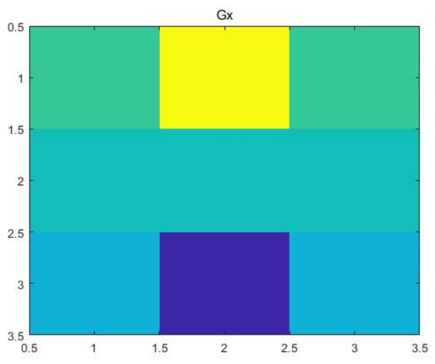
1.图像引入与灰度处理

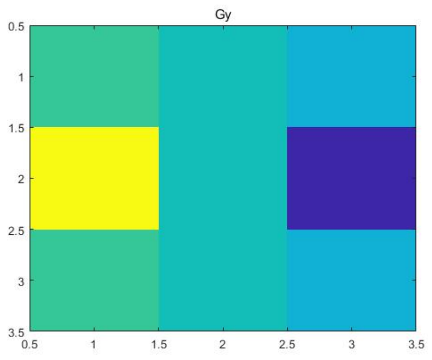
```
% 读入

img_origin = imread('Aqua.jpg');

% 灰度处理

img_gray = double(rgb2gray(img_origin));



figure('name','Origin Image and Grizzled Image');

imshowpair(img_origin, img_gray, 'montage');

title('Origin Image and Grizzled Image');
```

Origin Image and Grizzled Image

2.高斯一阶导数滤波器（Derivative of Gaussian Filter）

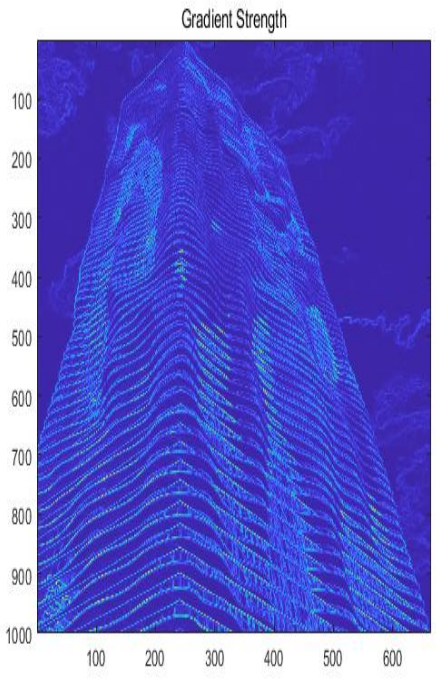
```
% 定义 x,y 方向的两个 3*3 的坐标定位矩阵(注意行列表示)

x_loc = [-1,-1,-1;

        0,0,0;

        1,1,1];

y_loc = [-1,0,1;

        -1,0,1;

        -1,0,1];


% 自定义高斯参数，使得窗宽与 sigma 尽量适配

sigma = 0.5;

% 将偏导步骤与高斯核构建步骤合并，用公式法计算高斯核每个元素位置（.*;./运算实现）的一阶导数

x_gdt = (-x_loc/(2*pi*(sigma^4))) .* exp(-(x_loc.*x_loc + y_loc.*y_loc)./(2*(sigma^2)));

y_gdt = (-y_loc/(2*pi*(sigma^4))) .* exp(-(x_loc.*x_loc + y_loc.*y_loc)./(2*(sigma^2)));


figure('name','Gx');

imagesc(x_gdt)

title('Gx');

figure('name','Gy');

imagesc(y_gdt)

title('Gy');
```





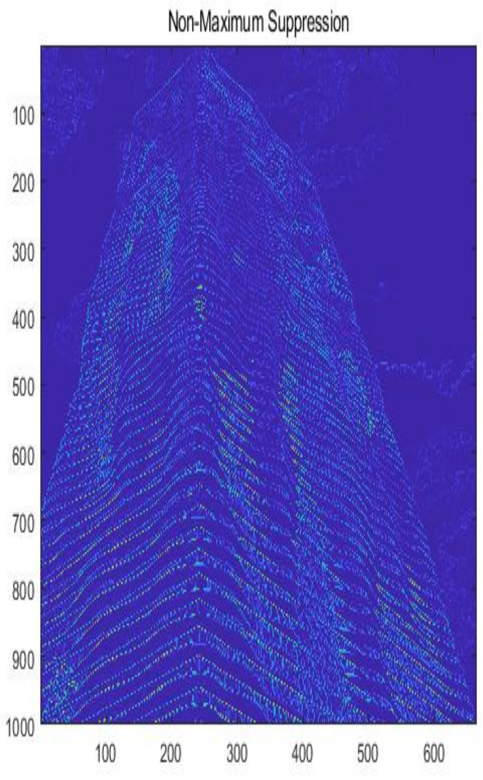
3.图像梯度计算

```
% 用相关对图像进行滤波得到图像的梯度

img_gd_x = imfilter(double(img_gray),x_gdt,'corr');

img_gd_y = imfilter(double(img_gray),y_gdt,'corr');

% 得到各像素点梯度幅值，构建为一个矩阵

img_gd_Strength = sqrt(img_gd_x.^2 + img_gd_y.^2);



figure('name','GS');

imagesc(img_gd_Strength)

title('Gradient Strength');



% 得到各像素点梯度方向角度，构建为一个矩阵（4 象限反正切）

img_gd_Direction = atan2(img_gd_y,img_gd_x);

% 将梯度方向归化至上下对角八个个方向，便于之后进行非极大值抑制操作

img_gd_Direction = round(img_gd_Direction ./ (pi/4)) .* (pi/4); %利用取整函数，将梯度方向角度大小变为最接近的 pi/4 的倍数
```



4.非极大值抑制算法实现

```
% 定义非极大值抑制像素存储矩阵

[img_m,img_n] = size(img_gray);

NMS = zeros(img_m,img_n);

% 扫描计算得到 NMS 结果

for img_x = 2:(img_m - 1)

    for img_y = 2:(img_n - 1)

        % 根据归化后的梯度方向的八个取值，讨论每个像素点处，与该像素点比较的两个像素（“比较点”）的位置

        % 水平方向“比较点”位置分类讨论（比较条件简化呈现）

        if ( img_gd_Direction(img_x,img_y) >= (pi/4) && img_gd_Direction(img_x,img_y) <= (3*pi/4) )

            move_y = 1;

        elseif ( img_gd_Direction(img_x,img_y) >= (-3*pi/4) && img_gd_Direction(img_x,img_y) <= (-pi/4) )

            move_y = -1;

        else

            move_y = 0;

        end

        % 竖直方向“比较点”位置分类讨论（比较条件简化呈现）

        if( abs(img_gd_Direction(img_x,img_y)) <= (pi/4) )

            move_x = 1;

        elseif ( abs(img_gd_Direction(img_x,img_y)) >= (3*pi/4) )

            move_x = -1;

        else

            move_x = 0;

        end

        % 将本像素点与两个“比较点”进行比较

        if      (img_gd_Strength(img_x,img_y)       >=       img_gd_Strength(img_x+move_x,img_y+move_y)       &&
img_gd_Strength(img_x,img_y) >= img_gd_Strength(img_x-move_x,img_y-move_y) )

            NMS(img_x,img_y) = img_gd_Strength(img_x,img_y); % 若比两个比较点都大，则保留
            else
                  NMS(img_x,img_y) = 0;
            end
      end
end


figure('name','NMS');
imagesc(NMS)
title('Non-Maximum Suppression');
```



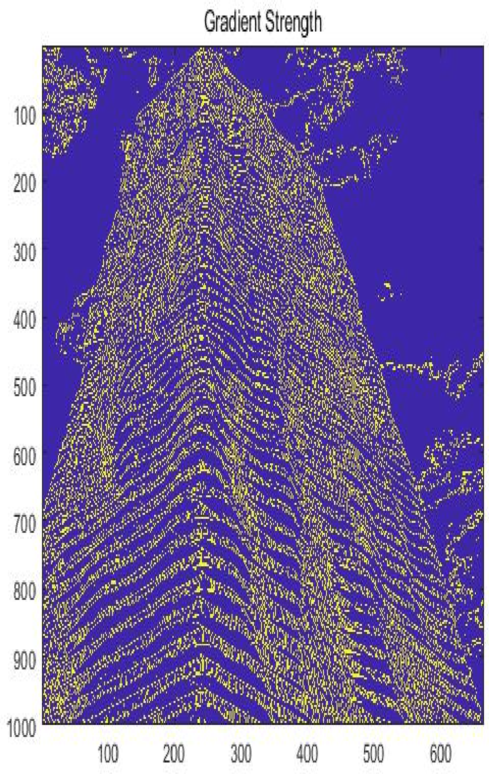
5.阈值操作及连接

```
% 定义双阈值
threshould_high = 5;
threshould_low = 1;
% 构建双阈值选择矩阵（图像）
```

```
img_high = double(NMS > threshould_high);

img_low = double(NMS > threshould_low);

% 在高阈值图像上根据低阈值图像进行连接

for x = 2:(img_m - 1)

    for y = 2:(img_n - 1)

        % 若像素点与已经判定为边的高阈值点相邻，则连接

        if (img_high(x,y) == 1)

            % 根据归化后的梯度方向的八个取值，讨论每个像素点的梯度邻接点位置

            % 水平方向

            if ( img_gd_Direction(img_x,img_y) >= (pi/4) && img_gd_Direction(img_x,img_y) <= (3*pi/4) )

                move_y = 1;

            elseif ( img_gd_Direction(img_x,img_y) >= (-3*pi/4) && img_gd_Direction(img_x,img_y) <= (-pi/4) )

                move_y = -1;

            else

                move_y = 0;

            end

            % 竖直方向

            if( abs(img_gd_Direction(img_x,img_y)) <= (pi/4) )

                move_x = 1;

            elseif ( abs(img_gd_Direction(img_x,img_y)) >= (3*pi/4) )

                move_x = -1;

            else

                move_x = 0;

            end

            % 邻接像素点连接

            img_high(x+move_x,y+move_y) = img_low(x+move_x,y+move_y);

        end

    end

end
```

6.最终生成边缘检测

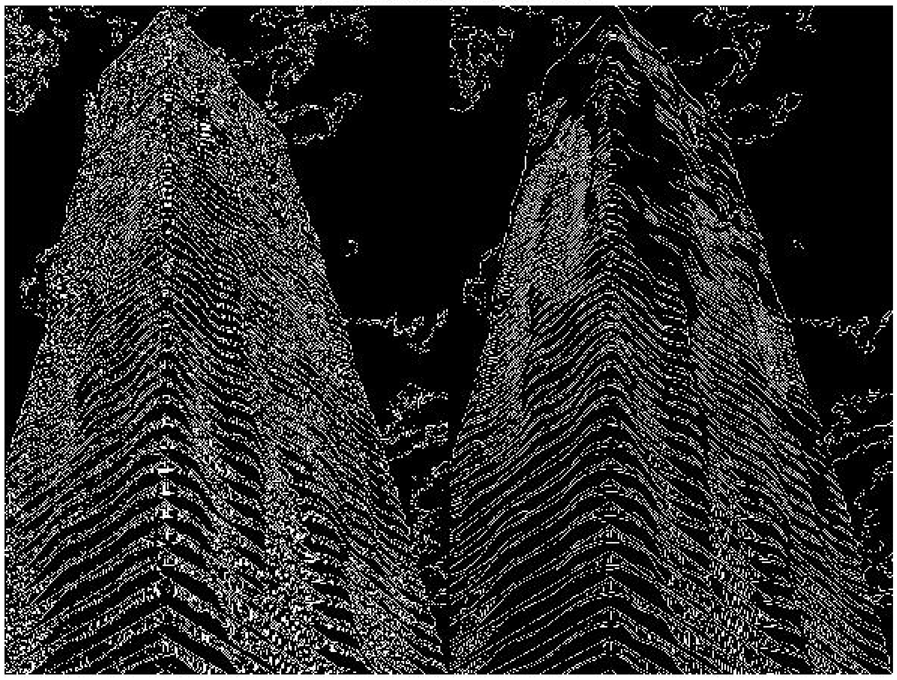
```
my_Canny = img_high;


figure('name','GS');

imagesc(my_Canny)

title('Gradient Strength');
```



7.直接 canny 边缘检测并对比

```
standard_Canny = edge(img_gray,'canny');


figure('name','canny 算子比较');

imshowpair(my_Canny,standard_Canny,'montage');

title('My Canny vs. Standard Canny');
```

My Canny vs. Standard Canny



Published with MATLAB® R2020b