The final report of Digital Image Processing and segmentation short course

Cheng Xinhao

Spring 2019

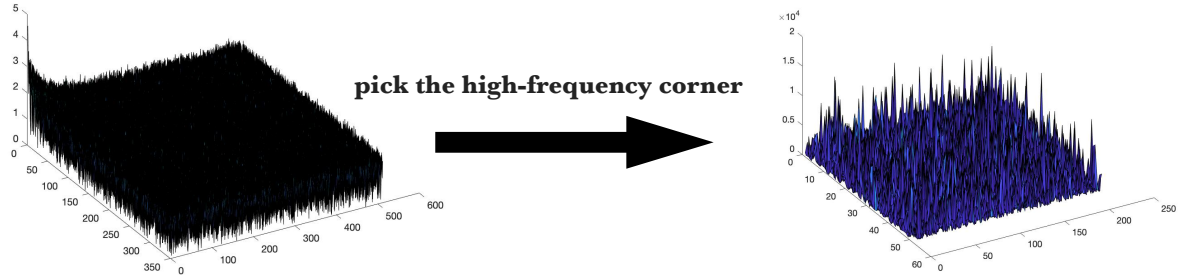
1.the DCT wiener filter

Basic idea: noise is concentrated in the high frequency region, and effective information is in the low frequency region.

1. Pick the Y channel of the image

2.use dct2(img) to do DCT convert to the image

3.he **Valid value** is always in the low frequency, the **Noise** is always in the high frequency. So take the high corner as a sample of the noise.



4. estimate the noise: $NoiseVariance = mean(mean(sample))$

5.choose a $beta$ to multiply the NoiseVariance coefficient.

6.$WienerFilter = 1 + (NoiseVariance ./ SignalVariance)$;

7.use the WienerFilter to multiply the image, and reverse it to the RGB, perform the result

2.the edge detection

Edge detection is a basic problem in image processing and computer vision. The purpose of edge detection is to identify points in the digital image where the brightness changes significantly, which often reflect some information people are interested in. It can greatly reduces the amount of data and eliminates information that is considered to be irrelevant, preserving the important structural properties of the image. So it is always an important area in the image processing and computer vision. There are many ways to perform edge detection. However, the majority of different methods may be grouped into two categories: the gradient based edge detection and laplacian based edge detection.

2.1. the gradient based edge detection:

The image gradient is the partial derivative of the current pixel point for the X-axis and the Y-axis, so it can be understood as the speed of the pixel gray value change in the image processing field.

the gradient $f'(x) = f(x+1) - f(x)$, if we use a 3*3 kernel, it is like :

$$f'(x) = f(x+1) - f(x-1)$$

or

$$f'(x) = 1 * f(x+1) + 0 * f(x) - 1 * f(x-1)$$

2.1.1 the Sobel operator:

The Sobel operator is a typical edge detection operator based on first derivative, it focus on the the horizontal and vertical direction, use two kernels to convolute with the image to obtain the horizontal and vertical **brightness difference approximations**.

base:

$$\frac{\partial f}{\partial x} = f(Z7,2Z8,Z9) - f(Z1,2Z2,Z3)$$

$$\frac{\partial f}{\partial y} = f(Z3,2Z6,Z9) - f(Z1,2Z4,Z7)$$

here is the two kernels Sobel operator always uses:

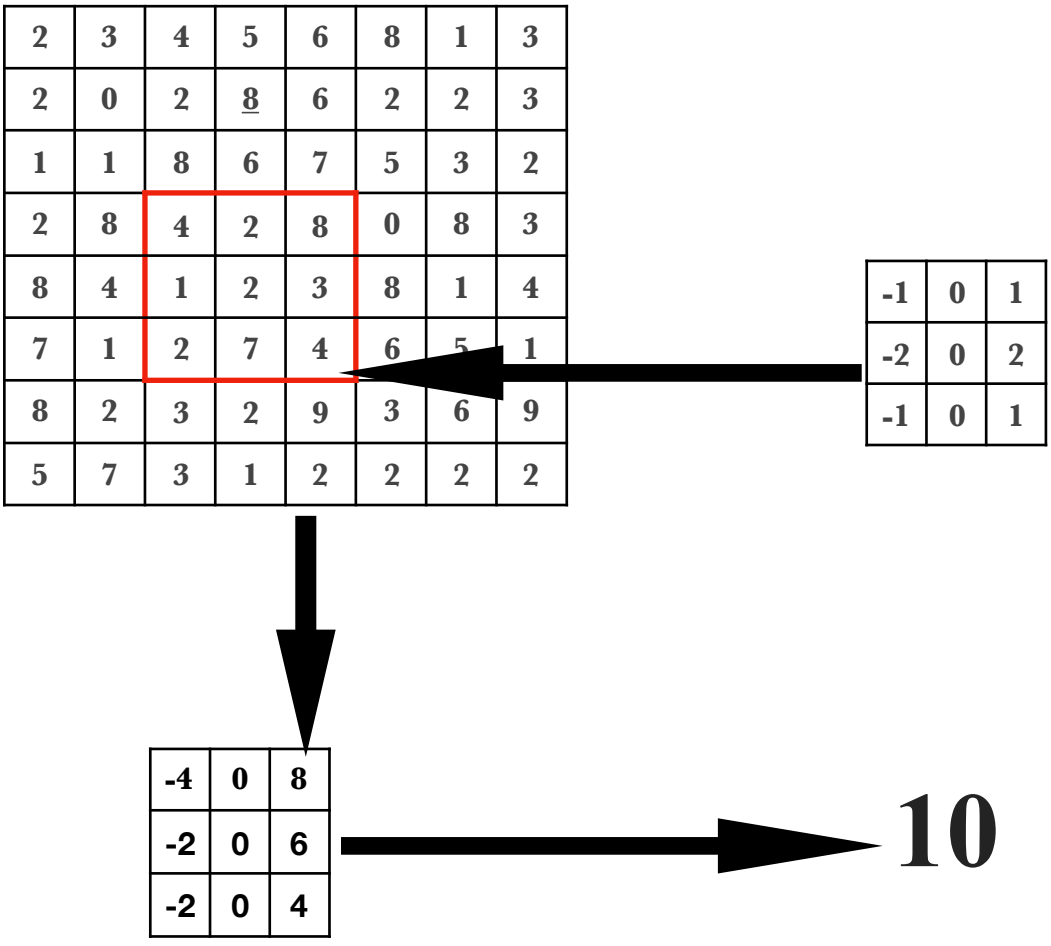
Gx

-1	0	1
-2	0	2
1	0	1

Gy

1	2	1
0	0	0
-1	-2	-1

The Gx kernel can determine the difference in X-axis direction , the Gy kernel can be used to determine the difference in Y-axis direction.The steps are shown below:



We can see that by using the two kernels to do convolution in the image, every pixel has a Gx value and Gy value, which can indicate the gradient in the two directions. Then we can use the formula below two calculate the gray-level of the pixel.

$$G = \sqrt{(Gx^2 + Gy^2)}$$

And the direction of gradient can be calculated by:

$$\theta = arctan(\frac{Gx}{Gy})$$

(it can be used in the canny operator)

After we have the gray-level(G) or edge strength of each pixel, we can choose a threshold to determine the lowest G in the image, to determine which pixels are remained, which pixels are thrown away. Smaller threshold, more information.

with threshold = 0.6

with threshold = 1.5

Demo algorithm code:

```
1.  Gx = [-1,-2,-1;0,0,0;1,2,1];
2.  Gy = Gx';
3.  sobel_threshold = ***;
```

set the Gx and Gy matrix and set the threshold

```
1.  for i = 1:H - 2
2.      for j = 1:W - 2
3.          temp = f(i:i+2,j:j+2);
4.          px = sum(sum(Gx.*temp));
5.          py = sum(sum(Gy.*temp));
6.          G = sqrt(px.^2+py.^2);
7.          if G > sobel_threshold
8.              im2(i,j) = 255;
9.          else
10.             im2(i,j) = 0;
11.         end
12.     end
13. end
```

make the convolution and calculate the gray-level

Features: Sobel operator introduces an **average** operation, so it has a smoothing effect on noise. But edge accuracy is not high, and there are some breakpoints.

2.1.2 the Prewitt operator: It is similar with the Sobel operator, but it doesn't use a weight of 2 on the center coefficient, so it is not as effective as the Sobel when dealing with the smooth noise

base:

$$\frac{\partial f}{\partial y} = f(Z3,Z6,Z9) - f(Z1,Z4,Z7)$$

$$\frac{\partial f}{\partial x} = f(Z7,Z8,Z9) - f(Z1,Z2,Z3)$$

the kernel:

Gx-2

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

Gy-2

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

The gradient direction:

$$\theta = arctan(\frac{Gx}{Gy})$$

Demo algorithm code:

```
1.  Gx = [-1,0,1;-1,0,1;-1,0,1];
2.  Gy = Gx';
3.  sobel_threshold = ***;
```

set the Gx and Gy matrix and set the threshold

The convolution process is same as Sobel operator

Features: Use threshold to determine the edge, but many noise points have large gray values, and for edge points with smaller amplitudes, the edges are lost.

2.1.3 the Roberts operator:The Roberts operator is one of the simplest operators. It uses the difference between two pixels in the diagonal direction to approximate the edge amplitude to detect the edge. It does have method to reduce the noise, so this operator is sensitive to noise.

base:

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

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

kernel:

Gx-3

0	**1**
-1	**0**

Gy-3

1	**0**
0	**-1**

the direction of gradient:

$$\theta = arctan(\frac{Gx}{Gy}) - 3\pi/4$$

Demo algorithm code:

```
1. Gx = [-1,0,1;-1,0,1;-1,0,1];
2. Gy = Gx';
3. sobel_threshold = ***;
```

set the Gx and Gy matrix and set the threshold

Feature: The effect of detecting the vertical edge is better than the oblique edge, the positioning accuracy is high, it is sensitive to noise, and the influence of noise cannot be suppressed.

2.2.the laplacian based edge detection: The Laplacian is a 2-D measure of the 2nd spatial derivative of an image. The Laplacian of an image highlights regions of rapid intensity change and are often used for edge detection.

Base:

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

in 2-D dimension:

$$\frac{\partial^2 f}{\partial^2 y} = -4f(x,y) + f(x-1,y) + f(x+1,y) + f(x,y-1) + f(x,y+1)$$

the corresponding kernel is :

0	-1	0
-1	4	-1
0	-1	0

if we consider the diagonal, it is like:

-1	-1	-1
-1	8	-1
-1	-1	-1

Because the laplacian kernels are approximating a second derivative measurement on the image, they are sensitive to noise. To counter this, the image is often Gaussian Smoothed before using the Laplacian filter to do the convolution

2.2.1: the canny kernel:

The Canny Edge Detection Operator is a multi-level edge detection algorithm developed by Australian computer scientist John F. Canny in 1986.

It contains the following processes:

1.Filter out the noise in the image before edge detection(always use the Gaussian filter)

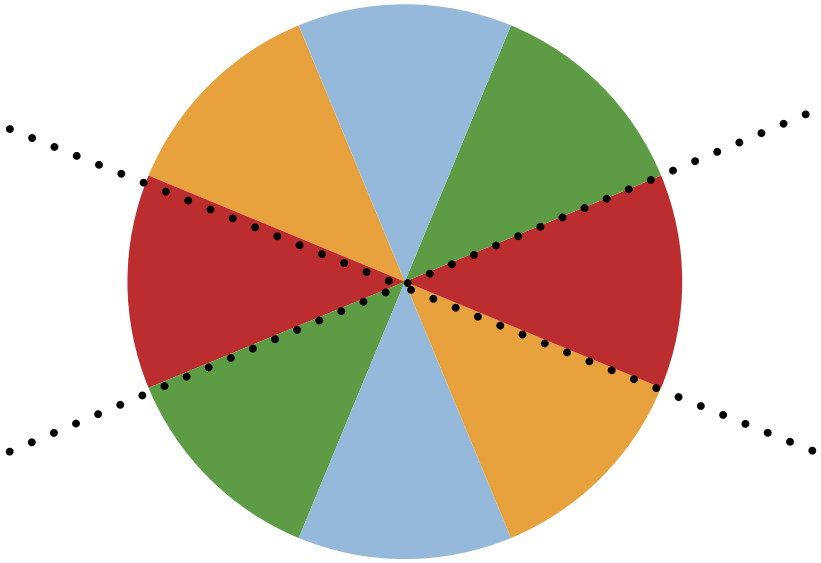
2.Find the edge strength(gray-level) of each pixel by using the gradient based operator(Sobel).

3.compute the direction of the edge by Gx and Gy(theta = arctan(Gx/Gy))

4. set 4-directions(0 degrees,45 degrees,90 degrees,135 degrees) of each pixel: use the range of theta

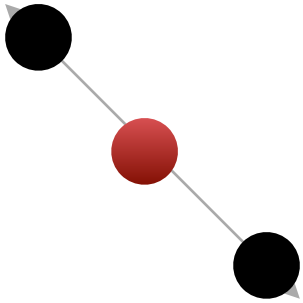
5.non-maximum suppression will be applied, trace along the edge in the edge direction and suppress any pixel value (sets it equal to 0) that is not considered to be an edge. This operate can get a thin edge and reduce some useless information

the 4 direction



6.then we can use the direction to estimate wether the pixel is the maximum one in this direction:

The values of the points at the intersection of the gradient directions may also be local maximum values. Therefore, judging the gray level of the **red** point and the gray level of the two points can determine whether the red point is the local maximum gray point in its neighborhood.



6.Use hysteresis to eliminate the streaking, which is the breaking up in the image. We use two threshold to set three kind of pixels: strong edge, weak edge and 0.

3.the Hough transform

Hough transform is a feature extraction that is widely used in image analysis, computer vision, and digital image processing. The Hough transform is used to identify features in an object, such as lines. His algorithm flow is roughly as follows. Given an object, the type of shape to be distinguished, the algorithm performs voting in the parameter space to determine the shape of the object, which is determined by the local maximum in the accumulator space(**from wiki**)

3.1Use Hough transform to detect lines in an image:

3.1.1 Bases:

The general representation of a line is:

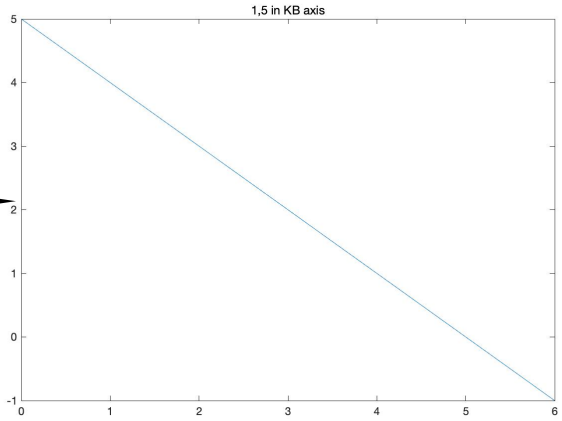
$$y = kx + b$$

k **and** *b* are two parameters, in Hough transform. in turn, the two parameters are treated as **arguments and variables** and the equation is written as:

$$b = -xk + y$$

at this time the axis has changed, the space has been converted. The original axis is xy , the new axis it is kb
a point in xy can convert to a line in kb:

(x,y) = (1,5)



Every pixels can correspond to a line in kb axis. So if the lines intersect at one point, it means the pixels have a same values of k and b**, it means there is a line in the image.**

If the angle between the straight line and the horizontal direction becomes larger, K becomes larger. If it is a vertical line, K will approach infinity. In order to simplify the calculation, we use polar coordinates in the actual experiment.

3.1.2 Procedures

1.use edge detection to filter the image

2. calculate the ρ range: $\sqrt{(rows^2 + cols^2)}$ and θ range: $(-90,90)$ in the image

3.create a matrix called Hough (the matrix is $\rho * \theta$) to save all groups of ρ and θ

4. create a loop in xy axis and calculate ρ and θ in every pixel, and save in Hough matrix.

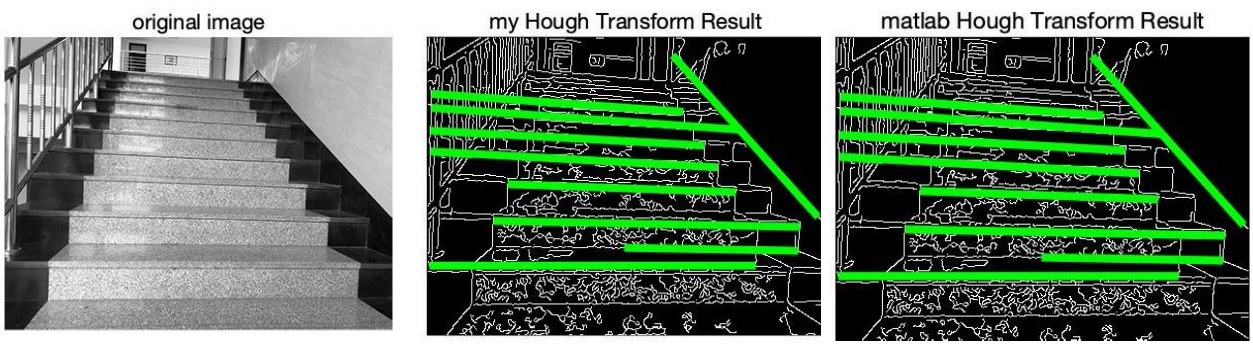
5.find the peak of group of ρ and θ

6.use houghlines to save all the line struct in the image, it contains **the start point**, the **end poin**t, the ρ and θ of the image

7.we can connect the straight lines with same ρ and θ, the ρ and θ is saved in the line struct

8.draw the line on image

3.1.3Results



4.some own work: Character recognition on the license plate

I have used image segmentation to create a small example of identifying a license plate number.

4.1 Procedure:

1.use im2bw(img) to change the image to binary

2.use bwareaopen(img,***) to reduce the small blocks that may useless information

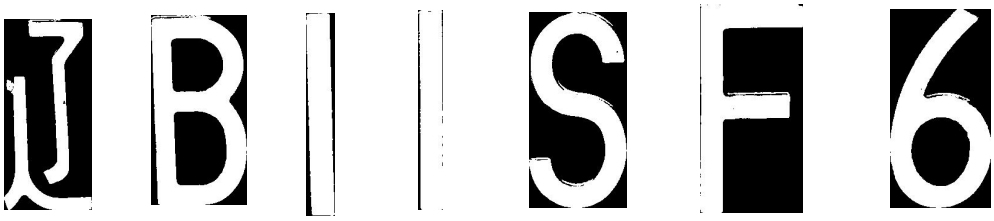
3.create a loop on the column of the image. if the sum of a column is 0, it means there is a white block, so we can divide the picture.
Use this method to divide the image into several small parts. Each part contains a number, letter or Chinese character.

4.resize the parts we have identified to make it same size with models in the database

4.compare the parts with the models which have been create.
Use all the models to minus one part one by one. And find which model has the smallest difference with the part.

5.perform the result

4.2 Result:



辽 B I I F S 6