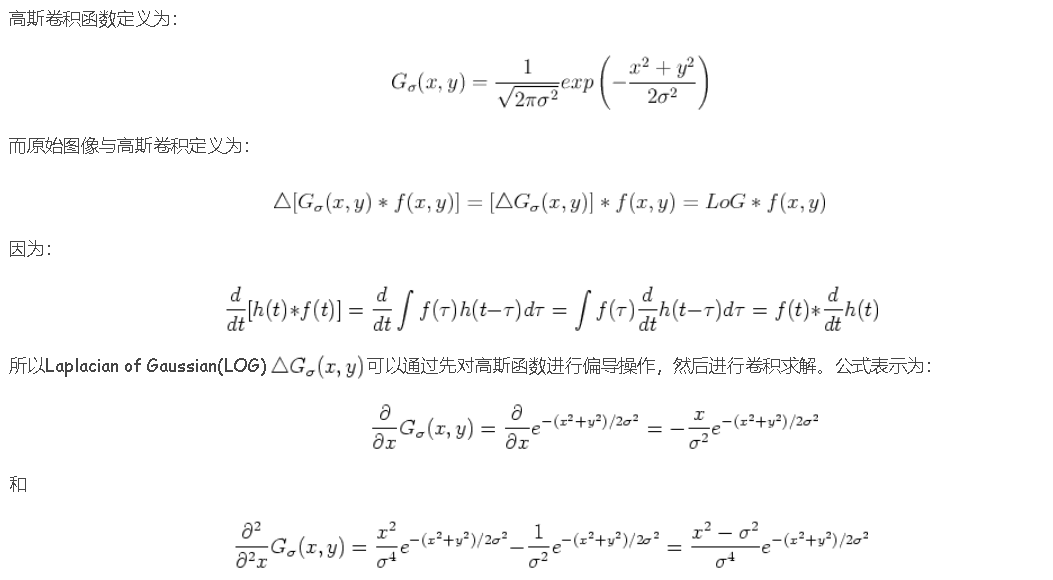
图像处理作业三

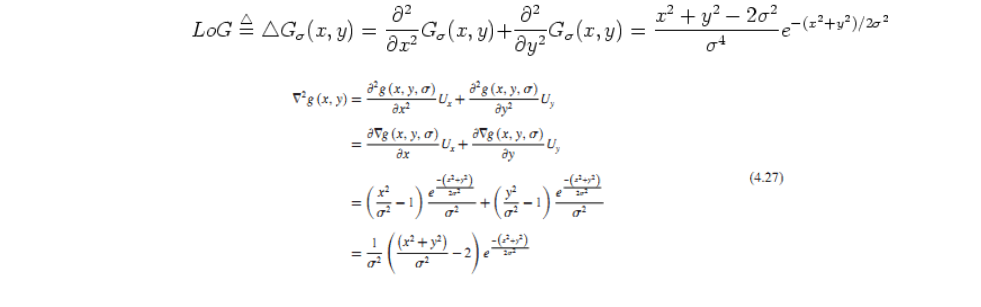
吴一汶 3017218110

其中霍夫曼拟合直线，最小二乘法，RANSAC处理边缘点用的是matlab，剩下三种用的是python

1. 对LoG的数学形式进行数学推导



因此LOG定义为：



2、实现最小二乘法、RANSAC法、霍夫变换法

2.1对于直线方程y=ax+b，生成一系列纵坐标符合高斯分布的点，再人工加入一系列的outlier，使用上述三种方法拟合一条直线

·最小二乘法拟合直线（**leastMul.py**）

代码实现：

import numpy.matlib

import random

import numpy as np

import matplotlib.pyplot as plt

# 直线为y=x+1 x=y-1

from sympy import flatten

# 生成50个符合高斯分布的点作为直线纵坐标

Y=np.matlib.randn(1, 50).A.flatten()

X = Y-1

fig = plt.figure()

# 画图区域分成1行1列。选择第一块区域。

ax1 = fig.add\_subplot(1,1, 1)

# 添加一些噪声。

random\_x = []

random\_y = []

for i in range(50):

random\_x.append(X[i])

random\_y.append(Y[i])

# 添加随机噪声

for i in range(10):

random\_x.append(random.uniform(-2,2))

random\_y.append(random.uniform(-2,2))

RANDOM\_X = np.array(random\_x) # 散点图的横轴。

RANDOM\_Y = np.array(random\_y) # 散点图的纵轴。

# 画散点图。

ax1.scatter(RANDOM\_X, RANDOM\_Y)

# 横轴名称。

ax1.set\_xlabel("x")

# 纵轴名称。

ax1.set\_ylabel("y")

#计算最小二乘的各个系数

def calcAB(x,y):

n = len(x)

sumX, sumY, sumXY, sumXX = 0, 0, 0, 0

for i in range(0, n):

sumX += x[i]

sumY += y[i]

sumXX += x[i] \* x[i]

sumXY += x[i] \* y[i]

a = (n \* sumXY - sumX \* sumY) / (n \* sumXX - sumX \* sumX)

b = (sumXX \* sumY - sumX \* sumXY) / (n \* sumXX - sumX \* sumX)

return a,b

#得到a b

a,b=calcAB(RANDOM\_X,RANDOM\_Y)

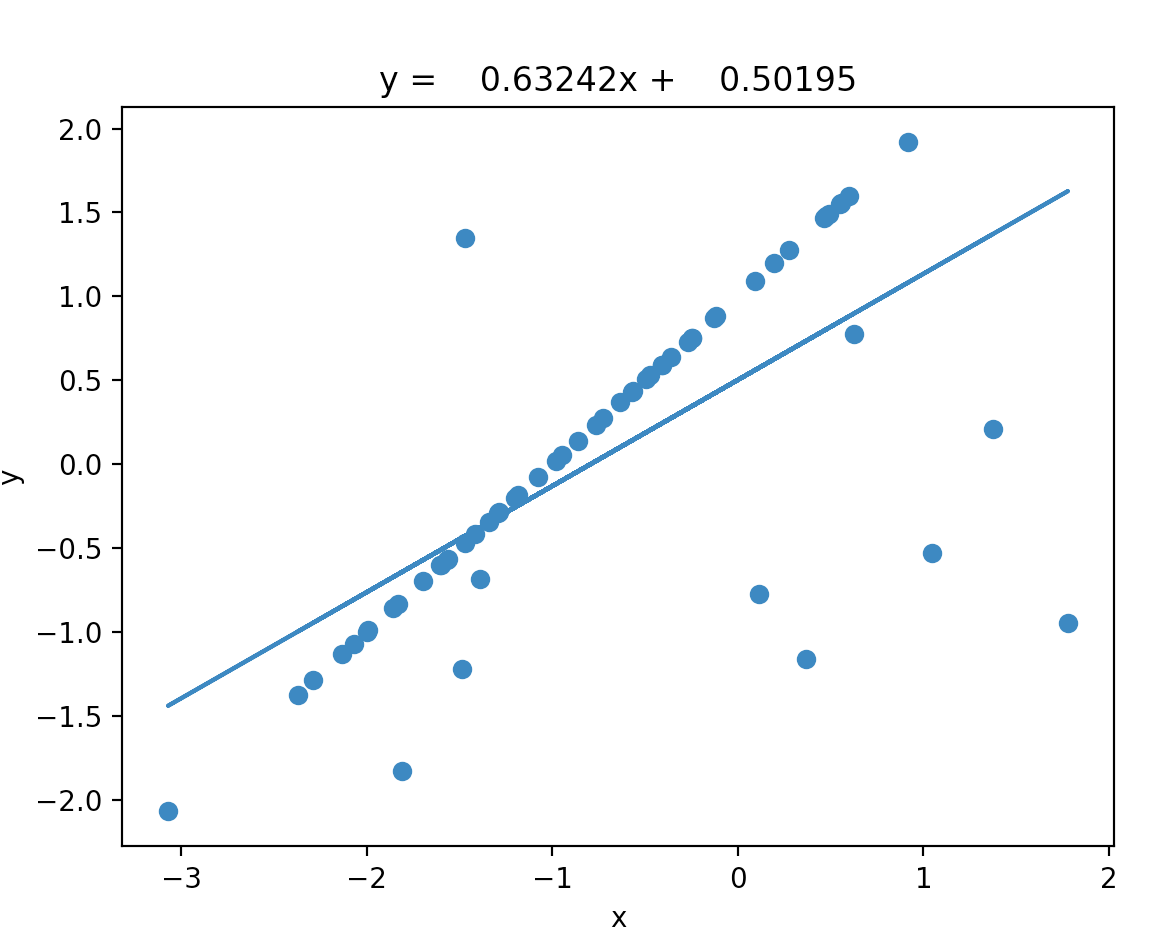
y1 = a\*RANDOM\_X+b

plt.plot(RANDOM\_X,y1)

plt.title("y = %10.5fx + %10.5f" %(a,b))

plt.show()

运行结果：



·RANSAC法拟合直线 （ransac.py）

import numpy as np

import matplotlib.pyplot as plt

import random

import math

import numpy.matlib

# 数据量。

SIZE = 50

# 产生数据

Y=np.matlib.randn(1, 50).A.flatten()

X = Y-1

fig = plt.figure()

# 画图区域分成1行1列。选择第一块区域。

ax1 = fig.add\_subplot(1,1, 1)

# 让散点图的数据更加随机并且添加一些噪声。

random\_x = []

random\_y = []

# 添加直线随机噪声

for i in range(SIZE):

random\_x.append(X[i] + random.uniform(-0.5, 0.5))

random\_y.append(Y[i] + random.uniform(-0.5, 0.5))

# 添加随机噪声

for i in range(SIZE):

random\_x.append(random.uniform(-10,10))

random\_y.append(random.uniform(-10,10))

RANDOM\_X = np.array(random\_x) # 散点图的横轴。

RANDOM\_Y = np.array(random\_y) # 散点图的纵轴。

# 画散点图。

ax1.scatter(RANDOM\_X, RANDOM\_Y)

# 横轴名称。

ax1.set\_xlabel("x")

# 纵轴名称。

ax1.set\_ylabel("y")

# 使用RANSAC算法估算模型

# 迭代最大次数，每次得到更好的估计会优化iters的数值

iters = 100000

# 数据和模型之间可接受的差值

sigma = 0.25

# 最好模型的参数估计和内点数目

best\_a = 0

best\_b = 0

pretotal = 0

# 希望的得到正确模型的概率

P = 0.99

for i in range(iters):

# 随机在数据中红选出两个点去求解模型

sample\_index = random.sample(range(SIZE \* 2),2)

x\_1 = RANDOM\_X[sample\_index[0]]

x\_2 = RANDOM\_X[sample\_index[1]]

y\_1 = RANDOM\_Y[sample\_index[0]]

y\_2 = RANDOM\_Y[sample\_index[1]]

# y = ax + b 求解出a，b

a = (y\_2 - y\_1) / (x\_2 - x\_1)

b = y\_1 - a \* x\_1

# 算出内点数目

total\_inlier = 0

for index in range(SIZE \* 2):

y\_estimate = a \* RANDOM\_X[index] + b

if abs(y\_estimate - RANDOM\_Y[index]) < sigma:

total\_inlier = total\_inlier + 1

# 判断当前的模型是否比之前估算的模型好

if total\_inlier > pretotal:

iters = math.log(1 - P) / math.log(1 - pow(total\_inlier / (SIZE \* 2), 2))

pretotal = total\_inlier

best\_a = a

best\_b = b

# 判断是否当前模型已经符合超过一半的点

if total\_inlier > SIZE:

break

# 用我们得到的最佳估计画图

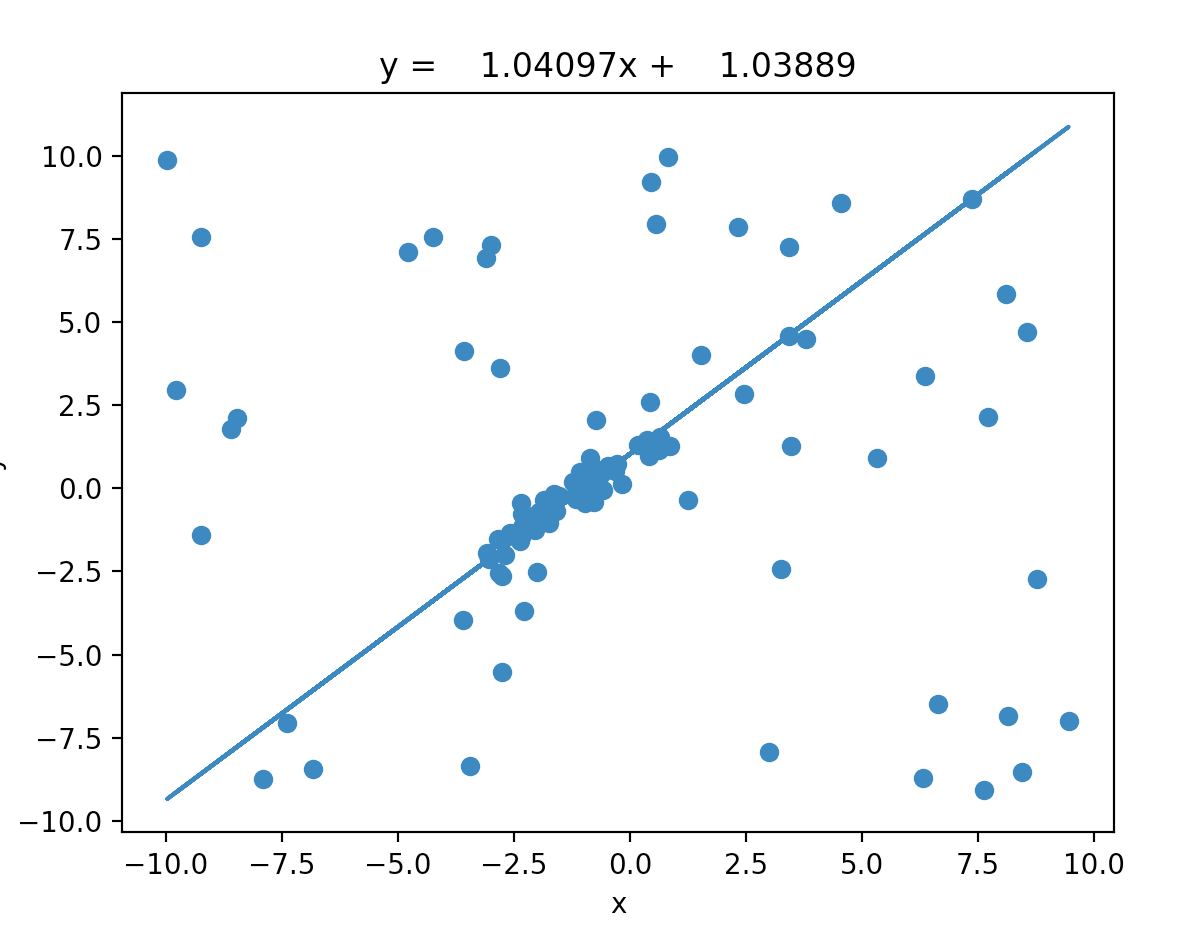
Y = best\_a \* RANDOM\_X + best\_b

# 直线图

ax1.plot(RANDOM\_X, Y)

plt.title("y = %10.5fx + %10.5f" %(best\_a,best\_b))

plt.show()

运行结果：

·霍夫变换法拟合直线

代码实现：

%生成五条带噪声直线用于直线拟合%

x1=sort(100.\*rand(1,100));

y1=x1+2+2.\*rand(1,100);

data1=[x1;y1];

x2=sort(100.\*rand(1,100));

y2=-4\*x2+4+4.\*randn(1,100);

data2=[x2;y2];

x3=sort(100.\*rand(1,100));

y3=16\*x3+6+6.\*rand(1,100);

data3=[x3;y3];

x4=sort(100.\*rand(1,100));

y4=-7\*x4+8+8.\*rand(1,100);

data4=[x4;y4];

x5=sort(100.\*rand(1,100));

y5=6\*x5+10+10.\*randn(1,100);

data5=[x5;y5];

data=[data1,data2,data3,data4,data5];%构建点集

[m,n]=size(data);%统计点数

%构建霍夫空间

n\_max=300;%霍夫空间的纵轴最大值

h=zeros(315,2\*n\_max);

theta\_i=1;

sigma=70;%设置拟合阈值

i=0;

%直线公式推导

%y=sin(theta)/cos(theta)\*x+b

%->p=b\*cos(theta)=-sin(theta)\*x+cos(theta)\*y

for theta=0:0.01:3.14

p=[-sin(theta),cos(theta)];

d=p\*data;

for i=1:n

%由于霍夫空间中d比较大，对d值进行了缩放

h(theta\_i,round(d(i)/10+n\_max))=h(theta\_i,round(d(i)/10+n\_max))+1;

end

theta\_i=theta\_i+1;

end

[theta\_x,p]=find(h>sigma);%查找投票数大于sigma的位置

l\_number=size(theta\_x);%符合直线条数

r=(p-n\_max)\*10;%将还原回距离R

theta\_x=0.01\*theta\_x;%将theta还原

figure('color','w');

plot(data(1,:),data(2,:),'\*');

hold on

x\_line=0:20:100;

for i=1:l\_number

if(abs(cos(theta\_x(i)))<0.01)%斜率不存在的情况

x=r(i);y=1:100;

plot(x,y,'r');

else

y=tan(theta\_x(i))\*x\_line+r(i)/cos(theta\_x(i));%画出拟合曲线

plot(x\_line,y,'r');

end

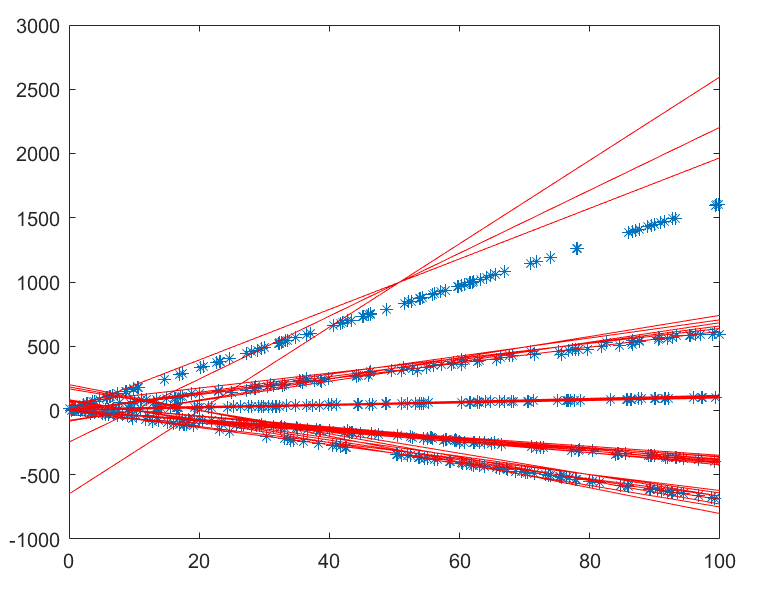
end

hold off

figure('color','w');

imshow(uint8(10\*h));%展示霍夫空间结果

运行结果：



2.2找到一幅实际图像，使用一阶导数或二阶导数找出边缘点，使用上述三种方法，找到其中的直线

·最小二乘法（**ImgLeast.py**）

代码实现：

im=imread('D:\test.jpg'); %读取图片

im=rgb2gray(im); %如果是rgb图片则转为灰度图

[h,w]=size(im); %获取图片高(h)、宽(w)

%扫描每一个像素,并记录白点（值为1）坐标及个数

n=0;

for y0=1:1:h

for x0=1:1:w

if(im(y0,x0)==255)

n=n+1;

y(n)=y0;

x(n)=x0;

end

end

end

%最小二乘法拟合直线

A = 0.0;

B = 0.0;

C = 0.0;

D = 0.0;

for i=1:1:n

A=A+x(i)\*x(i);

B=B+x(i);

C=C+x(i)\*y(i);

D=D+y(i);

end

a = (C\*n - B\*D) / (A\*n - B\*B);

b = (A\*D - C\*B) / (A\*n - B\*B);

%灰度图转rgb彩色图片

imrgb=repmat(im,[1,1,3]);

%画线，线宽3，红色

for i=1:1:w

y=a\*i+b;

y2=int32(y); %把y转换成整数

%把线上的通道1（红色）置为255

imrgb(y2-1,i,1)=255;

imrgb(y2,i,1)=255;

imrgb(y2+1,i,1)=255;

%把线上的其他通道（绿色和蓝色）置为0

imrgb(y2-1,i,2)=0;

imrgb(y2,i,2)=0;

imrgb(y2+1,i,2)=0;

imrgb(y2-1,i,3)=0;

imrgb(y2,i,3)=0;

imrgb(y2+1,i,3)=0;

end

%显示图片

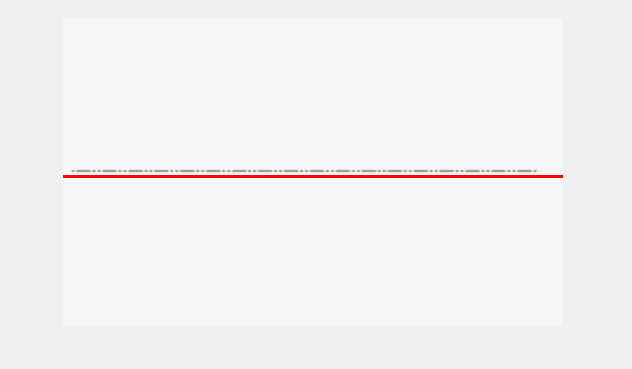
imshow(imrgb);

运行结果：

原图：



拟合结果：



·RANSAC法

代码实现：

img=imread('D:\test1.jpg');

% 拉普拉斯算子

filter=[0,1,0;1,-4,1;0,1,0];

% 算子大小

fsize=3;

flength = (fsize-1)/2;

% 图像灰度转换

bwImg = double (rgb2gray(img));

[imgH,imgW]=size(bwImg);

p=1;

imshow(bwImg);

% 处理图像，结果保存在gNewImg

for i=1+flength:imgH-flength

for j=1+flength:imgW-flength

temp = bwImg(i-flength:i+flength,j-flength:j+flength);

newImg(i,j)=sum(sum(temp.\*filter));

% 记录边缘点坐标

if newImg(i,j) ~= 0

x(p)=i;

y(p)=j;

p=p+1;

end

end

end

imshow(newImg);

% 对于边缘点拟合曲线

data = [x' y']';

% 显示数据点

% figure;

% scatter(data(1,:),data(2,:));

hold on;

number = size(data,2);

k=0;

b=0;

% 最佳匹配的参数

sigma=1;

for i=1:100

% 随机选择两个点

idx = randperm(number,2);

sample = data(:,idx)

% 拟合直线方程 y=kx+b

x = sample(1, :)

y = sample(2, :);

% 直线斜率

k=(y(1)-y(2))/(x(1)-x(2));

b = y(1) - k\*x(1);

line = [k -1 b];

% 求每个数据到拟合直线的距离

mask=abs(line\*[data; ones(1,size(data,2))]);

% 计算数据距离直线小于一定阈值的数据的个数

total=sum(mask<sigma);

% 找到符合拟合直线数据最多的拟合直线

if total>25

pretotal=total;

bestline=line;

% 最佳拟合的数据

mask=abs(bestline\*[data; ones(1,size(data,2))])<sigma;

k=1;

for i=1:length(mask)

if mask(i)

inliers(1,k) = data(1,i);

k=k+1;

end

end

% 绘制最佳匹配曲线

k = -bestline(1)/bestline(2);

b = -bestline(3)/bestline(2);

x = min(inliers(1,:)):0.1:max(inliers(1,:));

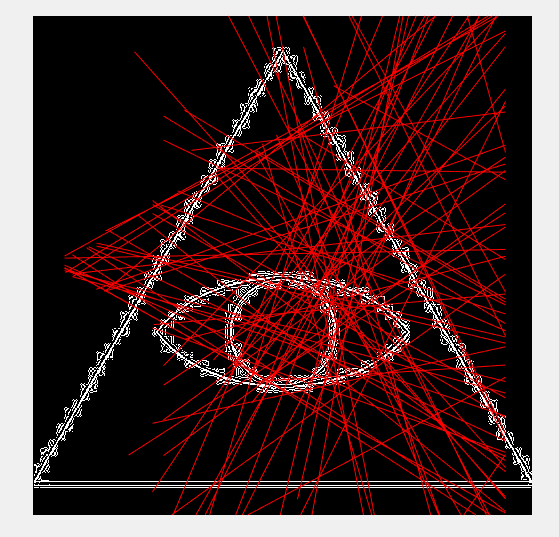
y = k\*x + b;

plot(x,y,'r');

end

end

运行结果：



·霍夫变换法（imgHough.py）

代码实现：

import cv2 as cv

import numpy as np

def line\_detection(image):

# 变换为灰度图

gray = cv.cvtColor(image, cv.COLOR\_BGR2GRAY)

# 进行Canny边缘检测

edge = cv.Canny(gray, 50, 100, apertureSize=3)

# 进行霍夫直线运算

lines = cv.HoughLines(edge, 1, np.pi/180, 200)

# 对检测到的每一条线段

for line in lines:

# 霍夫变换返回的是 r 和 theta 值

rho, theta = line[0]

a = np.cos(theta)

b = np.sin(theta)

# 确定x0 和 y0

x0 = a \* rho

y0 = b \* rho

# 认为构建（x1,y1）,(x2, y2)

x1 = int(x0 + 1000 \* (-b))

y1 = int(y0 + 1000 \* a)

x2 = int(x0 - 1000 \* (-b))

y2 = int(y0 - 1000 \* a)

# 用cv2.line( )函数在image上画直线

cv.line(image, (x1, y1), (x2, y2), (0, 0, 255), 2)

cv.imshow("line\_detection", image)

src = cv.imread("image.jpeg")

line\_detection(src)

cv.waitKey(0)

cv.destroyAllWindows()

运行结果

原图：

拟合结果：