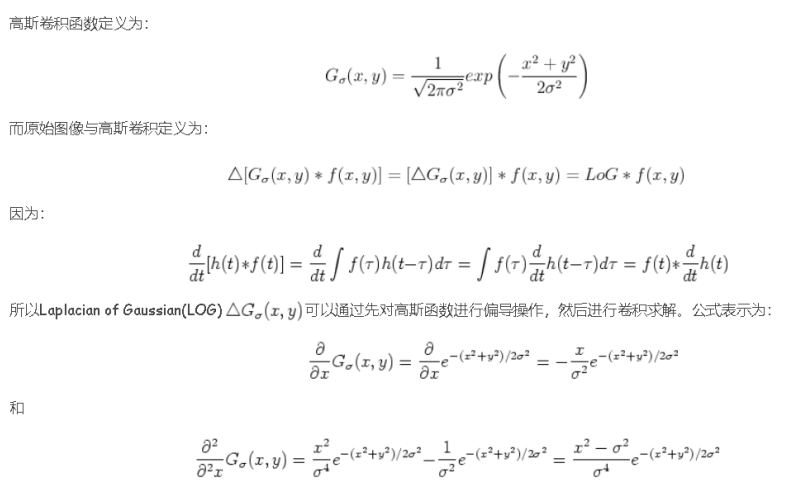
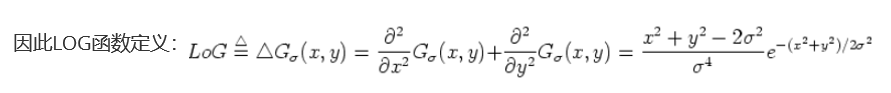
**图像处理第三次作业**

**3017218105 田承霖**

1. 对LoG的数学形式进行数学推导





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：

y=normrnd(0,1,[1,30]);

x=y+3;

%向直线中添加随机噪声

y=y+wgn(1,30,0);

x=x+wgn(1,30,0);

%计算最小二乘的各个系数

A = x\*x';

B = sum(x);

C = x\*y';

D = sum(y);

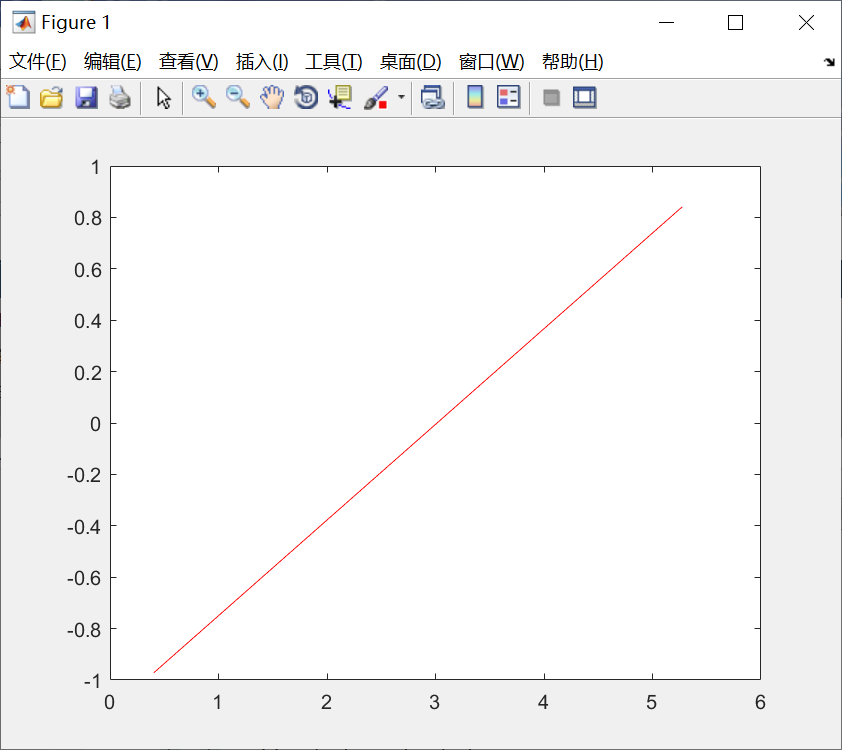
n=length(x)

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

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

y=x\*a+b;

plot(x,y,'r');

****

**RANSAC法拟合直线**

用RANSAC方法拟合原理：

RANSAC算法的输入是一组观测数据，一个可以解释或者适应于观测数据的参数化模型，一些可信的参数。RANSAC通过反复选择数据中的一组随机子集来达成目标。

%%%生成随机数据

x = normrnd(0,0.1,1,200);

y = x;

%生成outlier

tou1 = min(x)+rand(1,100)\*(max(x)-min(x)); %生成x定义域内的outlier

x = [x, tou1];

tou2 = min(y)+rand(1,100)\*(max(y)-min(y)); %生成y值域内的outlier

y = [y, tou2];

scatter(x,y); hold on;

data=[x',y'];

iter = 100;

%%% 绘制数据点

number = size(data,2); % 总点数

sigma = 1;

pretotal=0; %符合拟合模型的数据的个数

for i=1:iter

%%% 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, :);

k=(y(1)-y(2))/(x(1)-x(2)); %直线斜率

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

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

%b= x(2)-k\*x(1);

line = [k -1 b];

mask=abs(line\*[data; ones(1,size(data,2))]); %求每个数据到拟合直线的距离

total=sum(mask<sigma); %计算数据距离直线小于一定阈值的数据的个数

if total>pretotal %找到符合拟合直线数据最多的拟合直线

pretotal=total;

bestline=line; %找到最好的拟合直线

end

end

%显示符合最佳拟合的数据

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

hold on;

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

bestParameter1 = -bestline(1)/bestline(2);

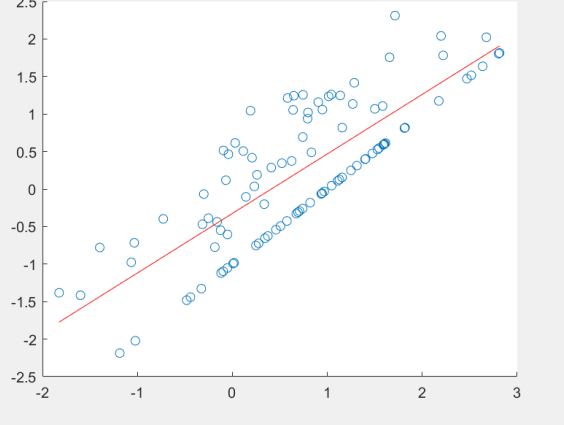
bestParameter2 = -bestline(3)/bestline(2);

xAxis = min(inliers(1,:)):max(inliers(1,:));

yAxis = bestParameter1\*xAxis + bestParameter2;

plot(xAxis,yAxis,'r','LineWidth',2);

title(['bestLine: y = ',num2str(bestParameter1),'x + ',num2str(bestParameter2)])



**霍夫变换法拟合直线**

clc

clear

%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');

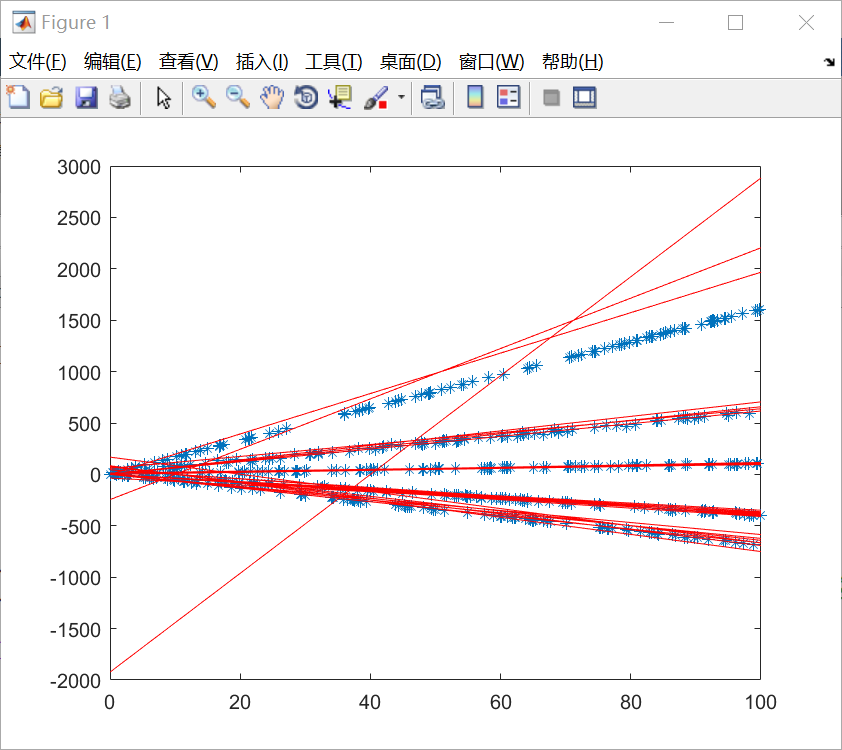
end

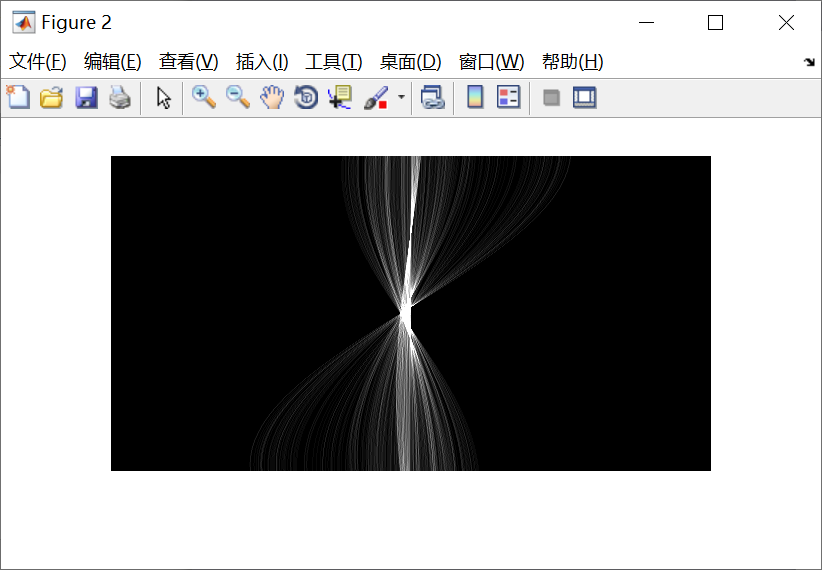
end

hold off

figure('color','w');

imshow(uint8(10\*h));%展示霍夫空间结果

****

****

**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**

img=imread('test.jpg');

[m,n]=size(img);

count = 1;

%Roberts算子一阶导数求边缘点

array=zeros(m,n);

for i=2:m-1

for j=2:n-1

array(i,j)=abs(img(i+1,j+1)-img(i,j))+abs(img(i,j+1)-img(i+1,j));

if array(i,j)<1

array(i,j)=0;

else

array(i,j)=255;

x(count)=i;

y(count)=j;

count = count + 1;

end

end

end

imshow('test.jpg');hold on;

point\_size = size(x,2);

%最小二乘法系数

x2=sum(x.^2); % 求Σ(xi^2)

x1=sum(x); % 求Σ(xi)

x1y1=sum(x.\*y); % 求Σ(xi\*yi)

y1=sum(y); % 求Σ(yi)

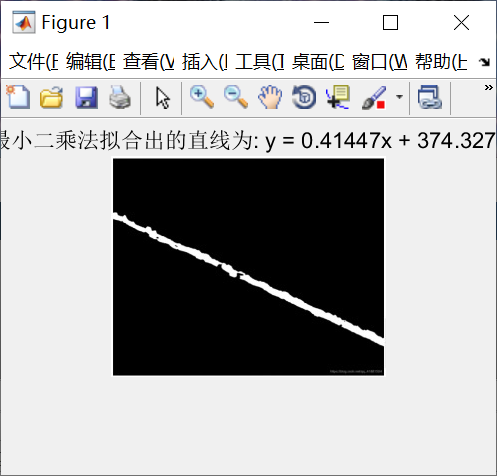
a=(point\_size \* x1y1 - x1\*y1)/(point\_size\*x2 - x1\*x1); %解出直线斜率b=(y1-a\*x1)/n

b=(y1 - a\*x1)/ point\_size; %解出直线截距

new\_y = a\*x+b;

plot(x,new\_y,'r');

title(['最小二乘法拟合出的直线为: y = ',num2str(a),'x + ',num2str(b)]);

****

**RANSAC法**

img=imread('test.jpg');

% 拉普拉斯算子

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

% 算子大小

fsize=3;

flength = 1;

% 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);

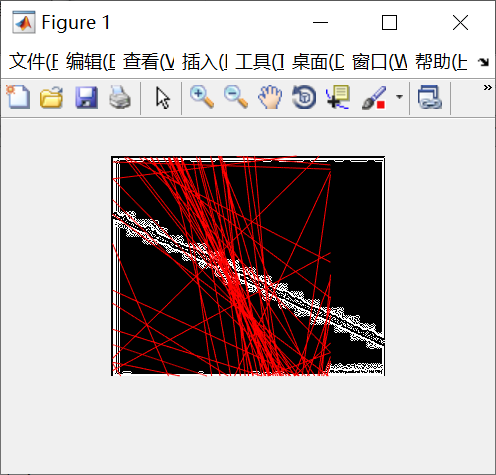
x = min(inliers(1,:)):0.1:max(inliers(1,:));

y = k\*x + b;

plot(x,y,'r');

end

end

****

**霍夫变换法**

img=imread('test.jpg');

[m,n]=size(img);

count = 1;

%Roberts算子一阶导数求边缘点

array=zeros(m,n);

for i=2:m-1

for j=2:n-1

array(i,j)=abs(img(i+1,j+1)-img(i,j))+abs(img(i,j+1)-img(i+1,j));

if array(i,j)<1

array(i,j)=0;

else

array(i,j)=255;

x(count)=i;

y(count)=j;

count = count + 1;

end

end

end

imshow('test.jpg');hold on;

point\_size = size(x,2);

data = [x;y];

%霍夫变换过程

n\_max=100;%霍夫空间的纵轴最大值

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

theta\_i=1;

sigma=70;%拟合阈值

for theta = 0:0.1:pi

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

d=p\*data;

for i=1:point\_size

%对霍夫空间中的d值进行缩放

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

line\_size=size(theta\_x);%符合直线条数

r=(p-n\_max)\*10;%还原距离R

x\_line = min(data(:)):0.01:max(data(:));

for i=1:40:line\_size

%斜率不存在的情况

if(abs(cos(theta\_x(i)))<0.01)

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

plot(x,y,'r');

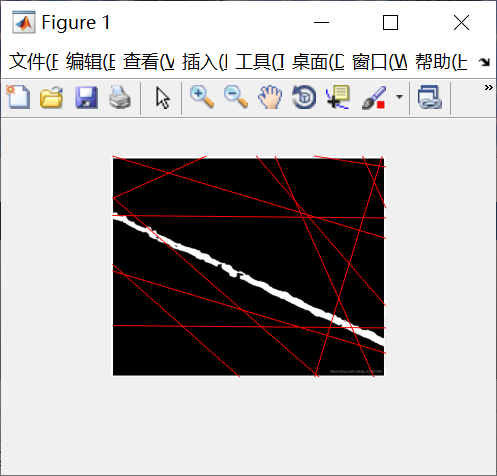
else %斜率存在的情况

y=tan(theta\_x(i))\*x\_line+r(i)/cos(theta\_x(i));

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

end

end

****