图像处理第三次作业

郑万杰

2019 年 12 月 10 日

对 LoG 的数学形式进行数学推导

$$h(r) = -e^{-\frac{r2}{2\sigma^2}}$$
$$h'(r) = \frac{r}{\sigma^2}e^{-\frac{r2}{2\sigma^2}}$$
$$h''(r) = (\frac{1}{\sigma^2} + \frac{r}{\sigma^2} \cdot \frac{-r}{\sigma^2}) \cdot e^{-\frac{r2}{2\sigma^2}}$$

最小二乘法

最小二乘法就是通过给定许多样本点，求一条直线，使得我们测得的每个数据，到这条直线的偏离量的总和最小。如果有一组 a,b 使得对 a 求偏导为 0，对 b 求偏导数为 0，那么 f(a,b)就是极值点，如果 a,b 只有一对，那么它就是最小值点。根据数学式的推导，

a*Σxi^2 + b*Σxi = Σ(xi*yi)

a*Σxi + b*N = Σyi

实现如下，其中输入(X,Y,n)分别表示样本 x,y 和样本数目

```
function test01(X,Y,n)
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=(n*x1y1-x1*y1)/(n*x2-x1*x1);          %解出直线斜率 b=(y1-a*x1)/n
b=(y1-a*x1)/n;                           %解出直线截距
figure
plot(X,Y,'+');
hold on;
px=X;
py=a*px+b;
plot(px,py,'r');
```

RANSAC 法

RANSAC 算法思想：从总体数据中随机抽取一部分数据，进行拟合，这里选取最小二乘法拟合，设置迭代次数，每次拟合之后计算误差在某个阈值（这里设置为 1）内的个数，如果数量足够多，那么就说它拟合效果比较好，停止迭代，否则继续迭代。

```
function test01(X,Y,n)
%迭代次数为 10 次
K=10;
%随机选取 m 个点
m=floor((n/5)*4);
sigma=1;             %设置拟合直线与数据距离的偏差
pretotal=0;        %符合拟合模型的数据的个数
k=1;
a=0,b=0;
while k<K && pretotal< floor((n/5)*3)
    c=randperm(numel(X));
    randomx1=[ones(m,1),X(c(1:m))];
    randomy1=Y(c(1:m));
    [bans,bint,r,rint,stats]=regress(randomy1 ,randomx1);
    total=sum(r<sigma);
    if total>pretotal                    %找到符合拟合直线数据最多的拟合直线
        pretotal=total;
        b=bans(1);a=bans(2);
    end
    k=k+1;
end
plot(X,Y,'+');
hold on;
px=X;
py=a*X+b;
plot(px,py,'r');
```

霍夫变换

霍夫变换是通过检测到的边缘点构建直线，然后把图像空间中的直线变换到坐标空间中的一个点，在参数空间相交于同一点的所有直线，在图像坐标空间都有共线的点与之对应。根据这个特性，给定图像坐标空间的一些边缘点，就可以通过 Hough 变换确定连接这些点的直线方程。使用 matlab 实现主要使用到以下三个函数：

通过 Hough 在二值图像中检测直线需要以下 3 个步骤。

利用 hough()函数执行霍夫变换，得到霍夫矩阵。

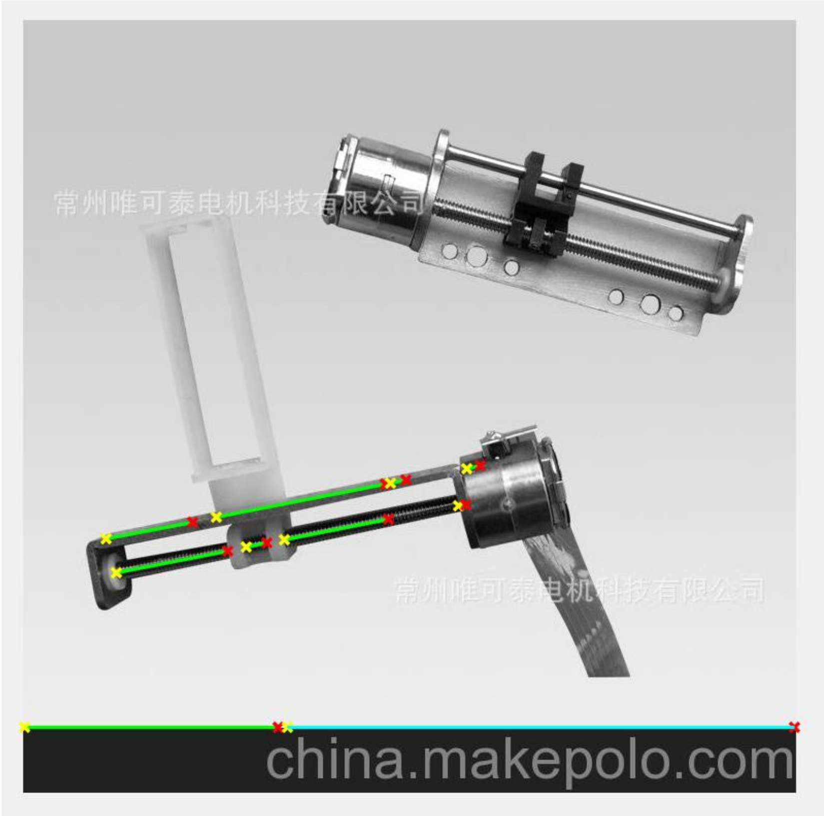
利用 houghpeaks()函数在霍夫矩阵中寻找峰值点。

houghlines()函数在之前 2 步结果的基础上得到原二值图像中的直线信息。

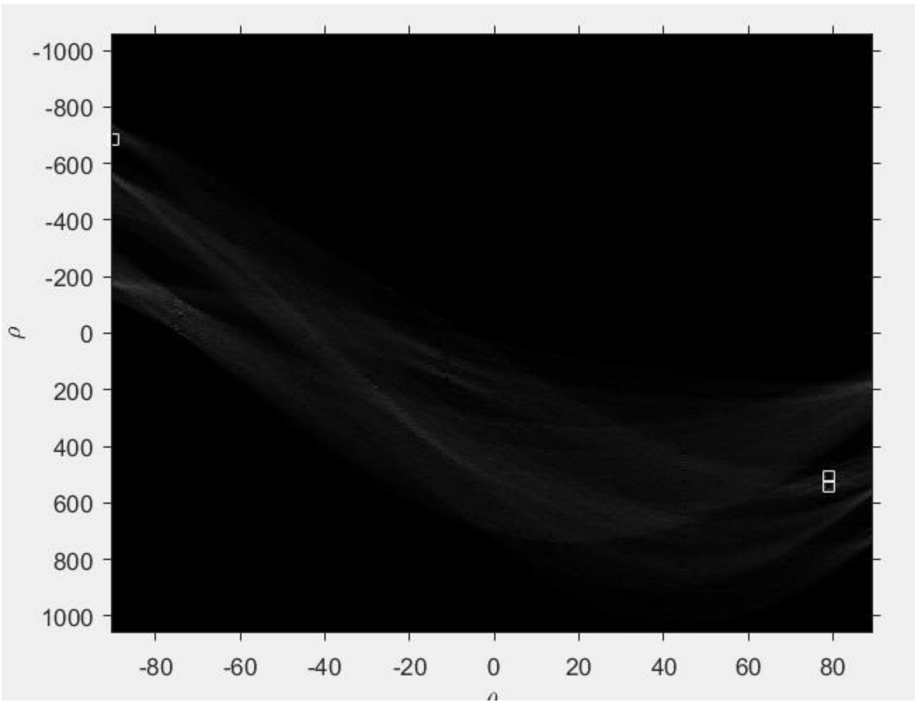
以下是具体实现

```
I    = rgb2gray(imread('line2.jpg'));
rotI = I;
BW = edge(rotI,'canny');
[H,T,R] = hough(BW);
imshow(H,[],'XData',T,'YData',R,'InitialMagnification','fit');
xlabel('\theta'), ylabel('\rho');
axis on, axis normal, hold on;
P    = houghpeaks(H,20,'threshold',ceil(0.3*max(H(:))));
x = T(P(:,2));
y = R(P(:,1));
plot(x,y,'s','color','white');
lines = houghlines(BW,T,R,P,'FillGap',5,'MinLength',7);
figure, imshow(rotI), hold on
max_len = 0;
for k = 1:length(lines)
        xy = [lines(k).point1; lines(k).point2];
        plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','green');
        plot(xy(1,1),xy(1,2),'x','LineWidth',2,'Color','yellow');
        plot(xy(2,1),xy(2,2),'x','LineWidth',2,'Color','red');
        len = norm(lines(k).point1 - lines(k).point2);
        if ( len > max_len)
            max_len = len;
            xy_long = xy;
        end
end
plot(xy_long(:,1),xy_long(:,2),'LineWidth',2,'Color','cyan');
```

直线拟合



如图所示，在图中绿色的是检测到的线，高亮显示的是最长的线。下图是参数空间中的点，每个点表示图像坐标中的一条直线。

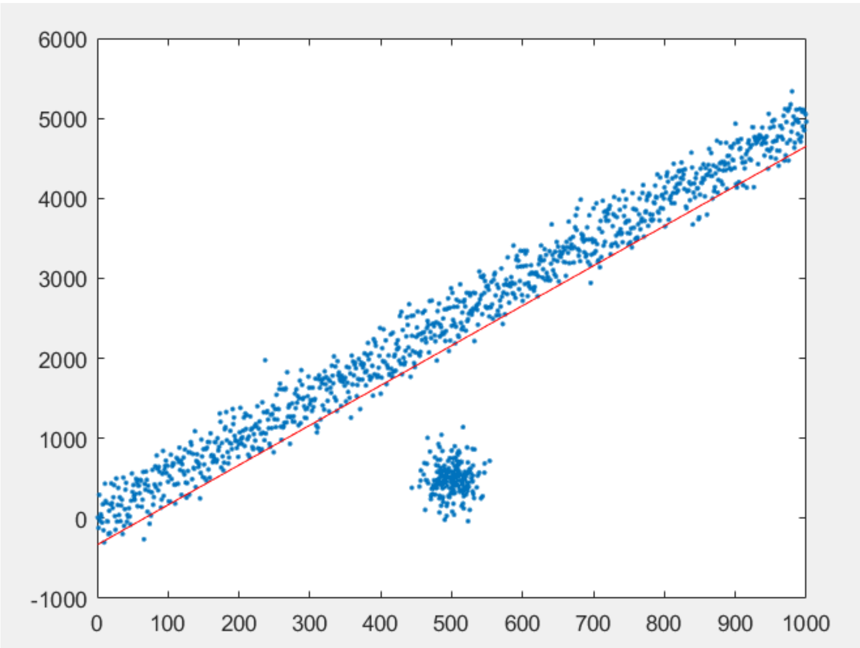


直线拟合

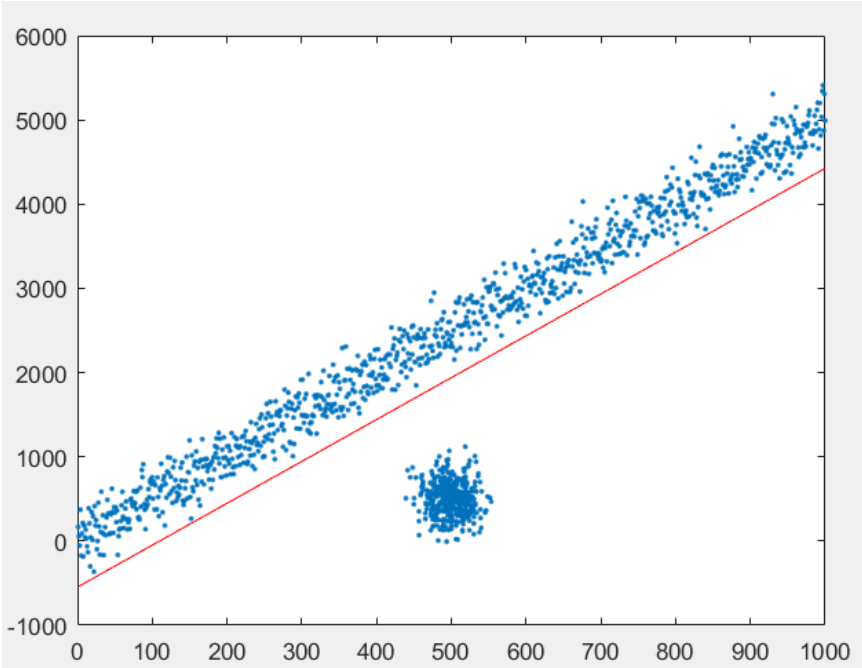
随机生成满足纵坐标为正态分布的数据，并加入一系列的离群点，实现方法如下图所示

```
function [x,y]=GenerRand_Linear()
a=5;
b=10;
x=[];
for i=1:1000
    x(i,1)=i;
    y(i,1)=a*i+b+randn(1,1);
end
for i=1001:1150
    x(i,1)=randn(1,1)*200+500;
    y(i,1)=randn(1,1)*200+500;
end
```

接下来，采用最小二乘法进行拟合，运行结果如下图所示，可以观察到它的拟合效果不是很好，因为最小二乘法兼顾了全局，把每一个点的误差都算进去了，如果我增加离群点的个数之后，它的拟合效果就会更差。



这是有 200 个离群点的效果，下面是有 400 个离群点的效果，可见离群点越多，拟合效果就越差。



接下来，采用 RANSAC 算法进行直线拟合观察结果，下图是有 200 个离群点、迭代次数为 5000 次的时候的拟合效果，它的效果比最小二乘法好。