Image Processing Homework3

3017207553 黄千慧

2019 年 12 月 11 日

实现最小二乘法、RANSAC 法、霍夫变换法对于直线方程 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找到一幅实际图像（较简单的），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

下面是最小二乘法、RANSAC 法、霍夫变换法实现的代码：

```
# least squares algorithm
# matrix implement
def least_squares(x, y):
    x = normalize_var(x)
    y = normalize_var(y)
    N = np.size(x,0) # total number of data
    x = np.vstack((x,np.ones(N))).T
    A = x.T@x
    f = x.T@y
    kb = np.linalg.solve(A, f)
    return kb

def ransac(x, y):
    x = normalize_var(x)
    y = normalize_var(y)
    N = x.shape[0]
    # the iters value will optime in each loop
    iters = 100000
    sigma = 0.25
    res_k, res_b = 0, 0
    total = 0
    P = 0.99
    for i in range(iters):
        x_1, x_2 = np.random.choice(x, size=2)
        y_1, y_2 = np.random.choice(y, size=2)
        k = (y_2 - y_1) / (x_2 - x_1)
        b = y_1 - k * x_1

```

```
        inlier_num = 0
        for i in range(N):
            y_temp = k * x[i] + b
            if abs(y_temp - y[i]) < sigma:
                inlier_num += 1

        if inlier_num > total:
            iters = math.log(1 - P) / math.log(1 - math.pow(inlier_num / N, 2))
            total = inlier_num
            res_k, res_b = k, b

        if inlier_num > N / 2:
            return [res_k, res_b]
    return [res_k, res_b]

def hough_transform(x, y):
    x = normalize_var(x)
    y = normalize_var(y)
    N = x.shape[0]
    sin_val = np.array([math.sin(i * math.pi / 180)
                for i in range(-90, 90)])
    cos_val = np.array([math.cos(i * math.pi / 180)
                for i in range(-90, 90)])
    hough = np.array([([0] * 180)
                for i in range(
                math.floor(np.max(sin_val) * np.max(x) +
                    np.max(cos_val) * np.max(y))
                )])
    total = 0
    sin_res, cos_res = 0, 0
    for i in range(N):
        for j in range(N):
            for k in range(180):
                tp = math.floor(sin_val[k] * y[j] + cos_val[k] * x[i])
                hough[tp][k] += 1
                if hough[tp][k] > total:
                    total = hough[tp][k]
                    sin_res, cos_res = sin_val[k], cos_val[k]
                    p = sin_val[k] * y[j] + cos_val[k] * x[i]
    return [sin_res, cos_res, p]
```

实验结果

1. 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
   如图 1 所示

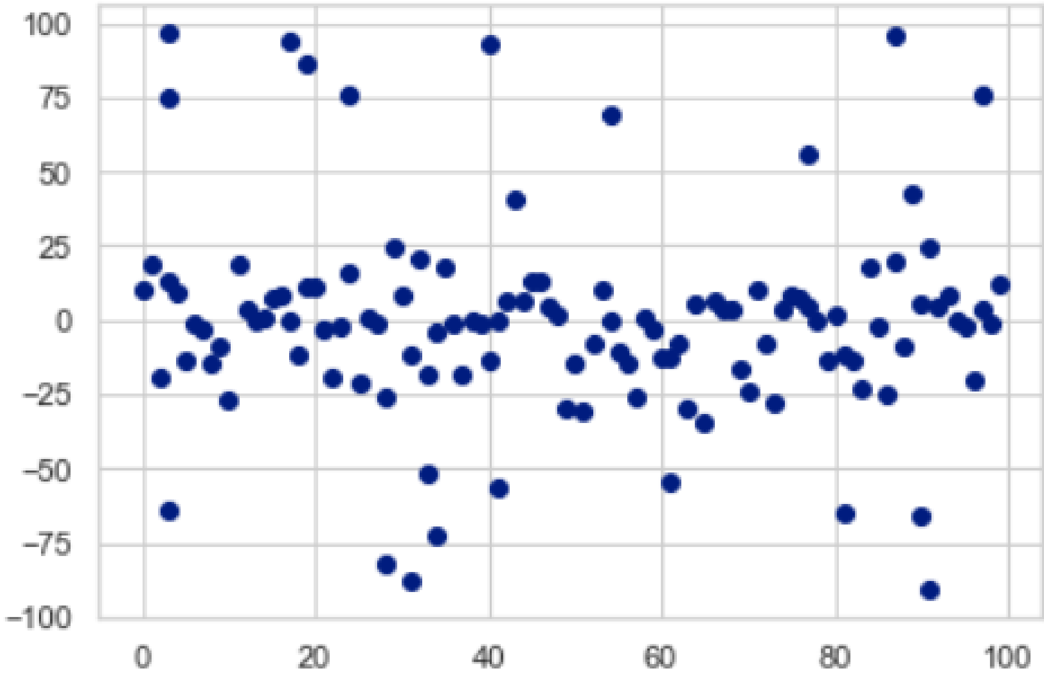

图 1: 点分布情况

2. 最小二乘法拟合
   如图 2所示
3. RANSAC 法拟合
   如图 3 所示
4. 霍夫变换法拟合
   如图 4 所示
5. 对图片使用 Canny 算子提取边缘点，再进行 Hough 直线检测
   如图 5 所示

运行结果在文件目录的 report/image 目录下

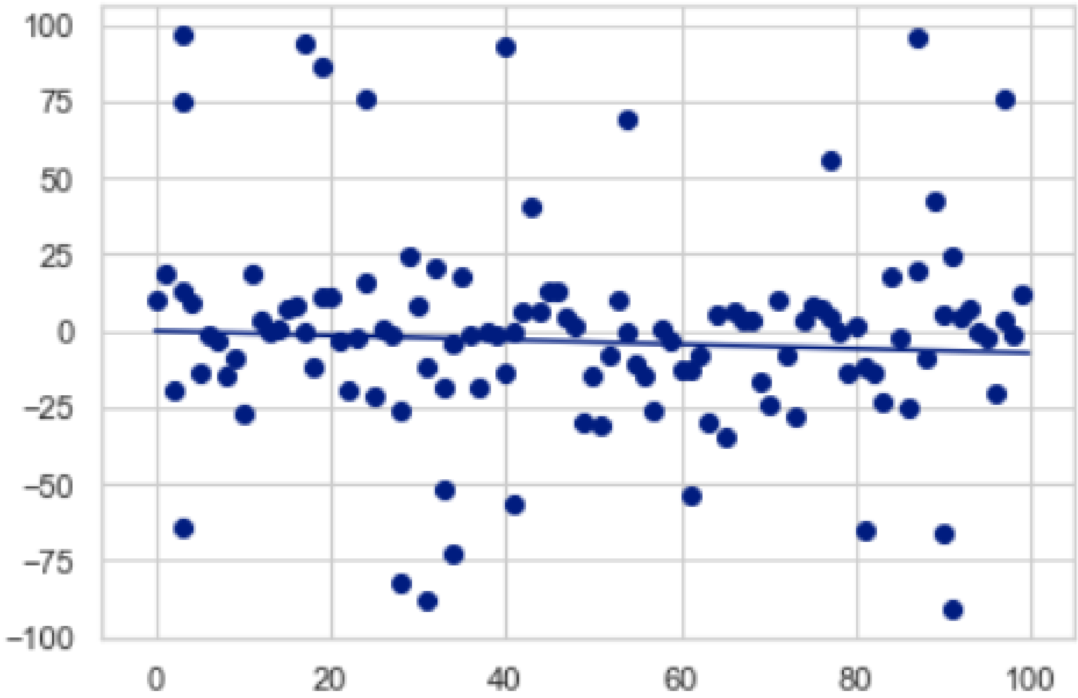

图 2: 最小二乘法拟合

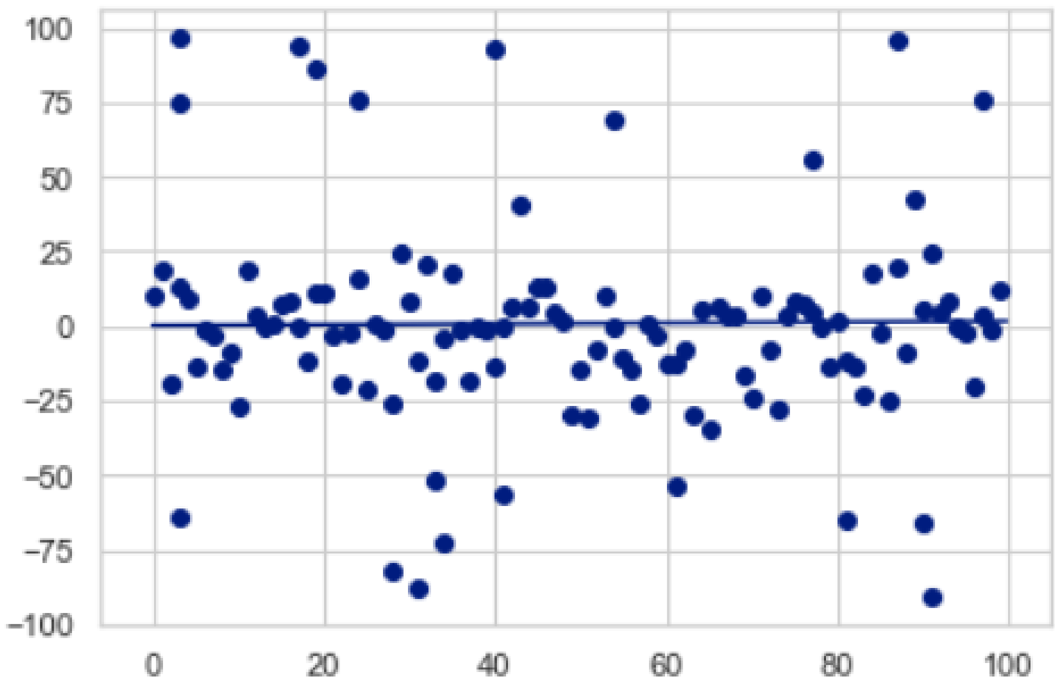

图 3: RANSAC 法拟合

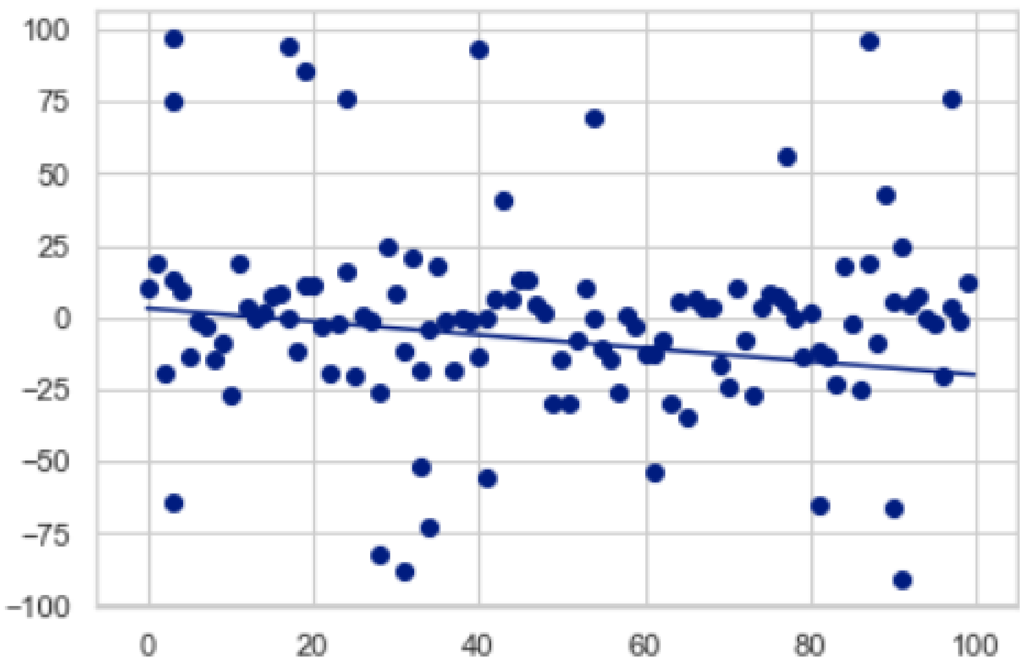

图 4: 霍夫变换法拟合

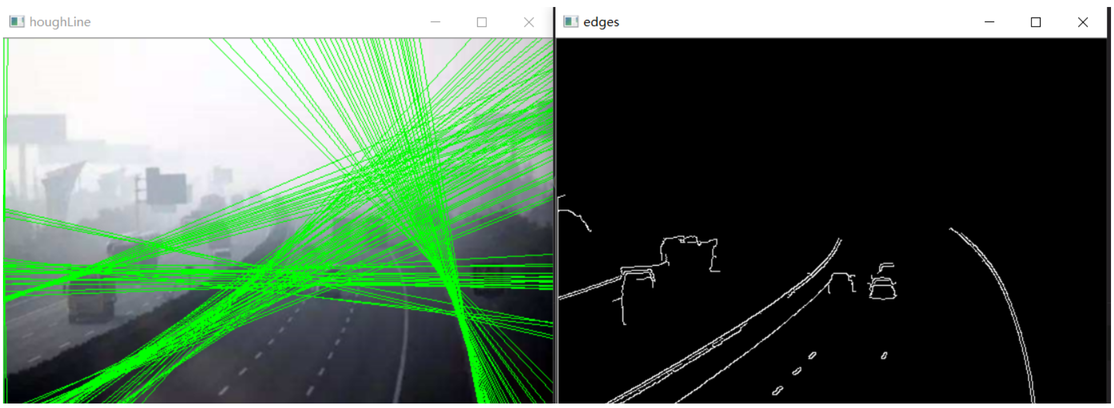

图 5: canny-hough 运行结果