第三次作业报告

姓名 赵鸿博 学号 3017218180

实验目的

- 对LoG的数学形式进行数学推导
- 实现最小二乘法、RANSAC法、霍夫变换法
  - 对直线方程 y = 2x 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，使用三种方法拟合直线
  - 找到一幅简单图像，使用一阶导数或二阶导数找出边缘点，使用三种方法，找到其中的直线

实验过程

实验一

对高斯模型求二阶导数

高斯卷积函数定义为：

$$G_\sigma(x,y)=\frac{1}{\sqrt{2\pi\sigma^2}}exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

而原始图像f(x,y) 与高斯卷积定义为：

$$\triangle[G_\sigma(x,y)*f(x,y)]=[\triangle G_\sigma(x,y)]*f(x,y)=LoG*f(x,y)$$

因为：

$$\frac{d}{dt}[h(t)*f(t)]=\frac{d}{dt}\int f(\tau)h(t-\tau)d\tau=\int f(\tau)\frac{d}{dt}h(t-\tau)d\tau=f(t)*\frac{d}{dt}h(t)$$

所以LoG可以通过先对高斯函数进行偏导操作，然后进行卷积求解。公式表示为：

$$\frac{\partial}{\partial x}G_\sigma(x,y)=\frac{\partial}{\partial x}e^{-(x^2+y^2)/2\sigma^2}=-\frac{x}{\sigma^2}e^{-(x^2+y^2)/2\sigma^2}$$

和

$$\frac{\partial^2}{\partial^2 x}G_\sigma(x,y)=\frac{x^2}{\sigma^4}e^{-(x^2+y^2)/2\sigma^2}-\frac{1}{\sigma^2}e^{-(x^2+y^2)/2\sigma^2}=\frac{x^2-\sigma^2}{\sigma^4}e^{-(x^2+y^2)/2\sigma^2}$$

因此，我们可以推导出LoG数学形式：

$$LoG \triangleq \triangle G_\sigma(x,y) = \frac{\partial^2}{\partial x^2}G_\sigma(x,y) + \frac{\partial^2}{\partial y^2}G_\sigma(x,y) = \frac{x^2+y^2-2\sigma^2}{\sigma^4}e^{-(x^2+y^2)/2\sigma^2}$$

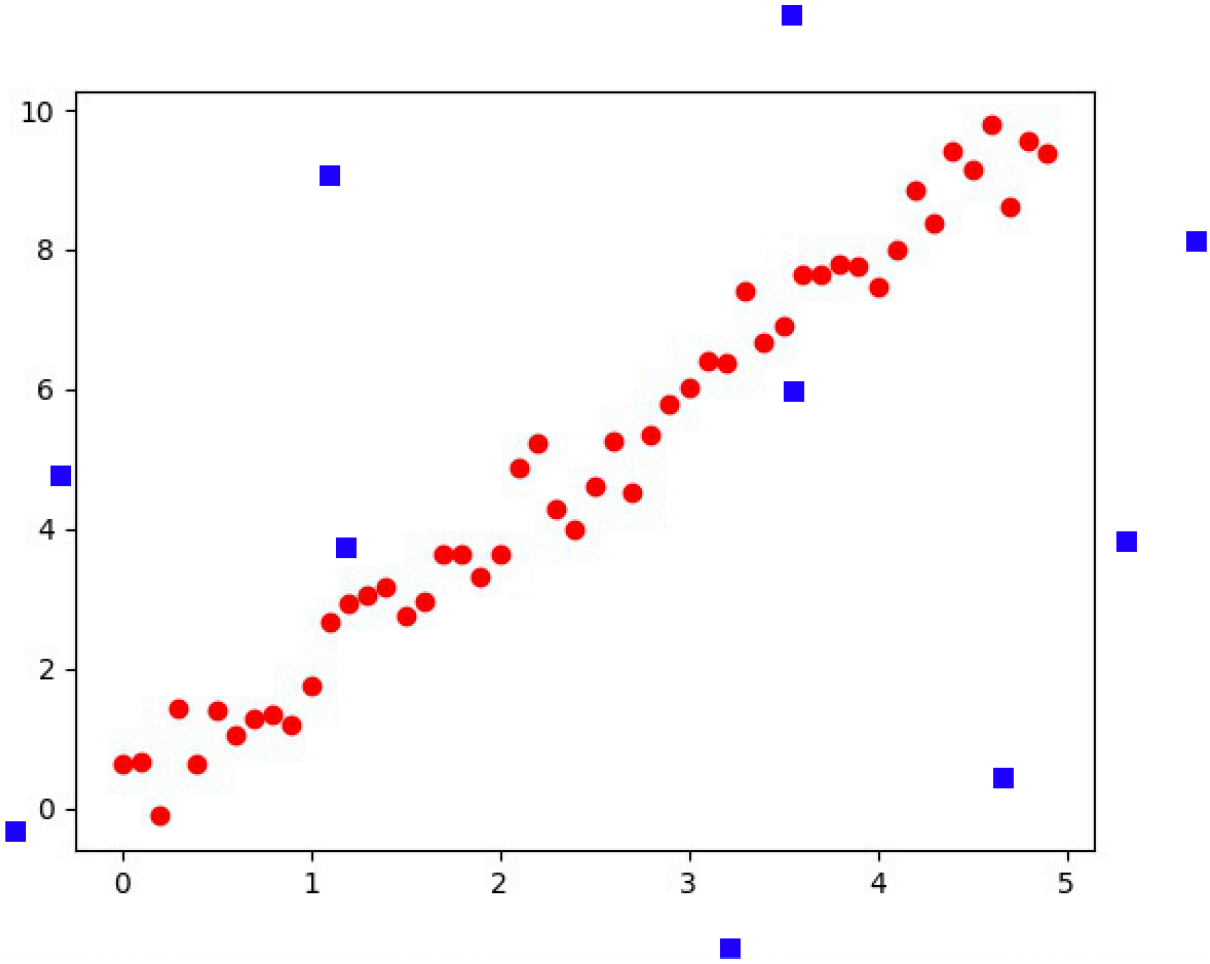
实验二

- 首先生成直线y = 2x中纵坐标符合高斯分布的点并人工添加随机的outlier，绘制出直线的图像

```
X = np.arange(0, 5, 0.1)
Z = [2 * x for x in X]
Y = [np.random.normal(z, 0.5) for z in Z]

plt.plot(X, Y, 'ro')
plt.savefig('./2x.jpg')
image = Image.open("./2x.jpg","r")
line = (0, 0, 255, 1)

for n in range(10):
    i = random.randint(0,image.size[0]-10)
    y = random.randint(0,image.size[1]-10)
    for j in range(10):
        for k in range(10):
            image.putpixel((i + j, y + k), line)
image.show()
image.save("./2x.jpg")
```



最小二乘法

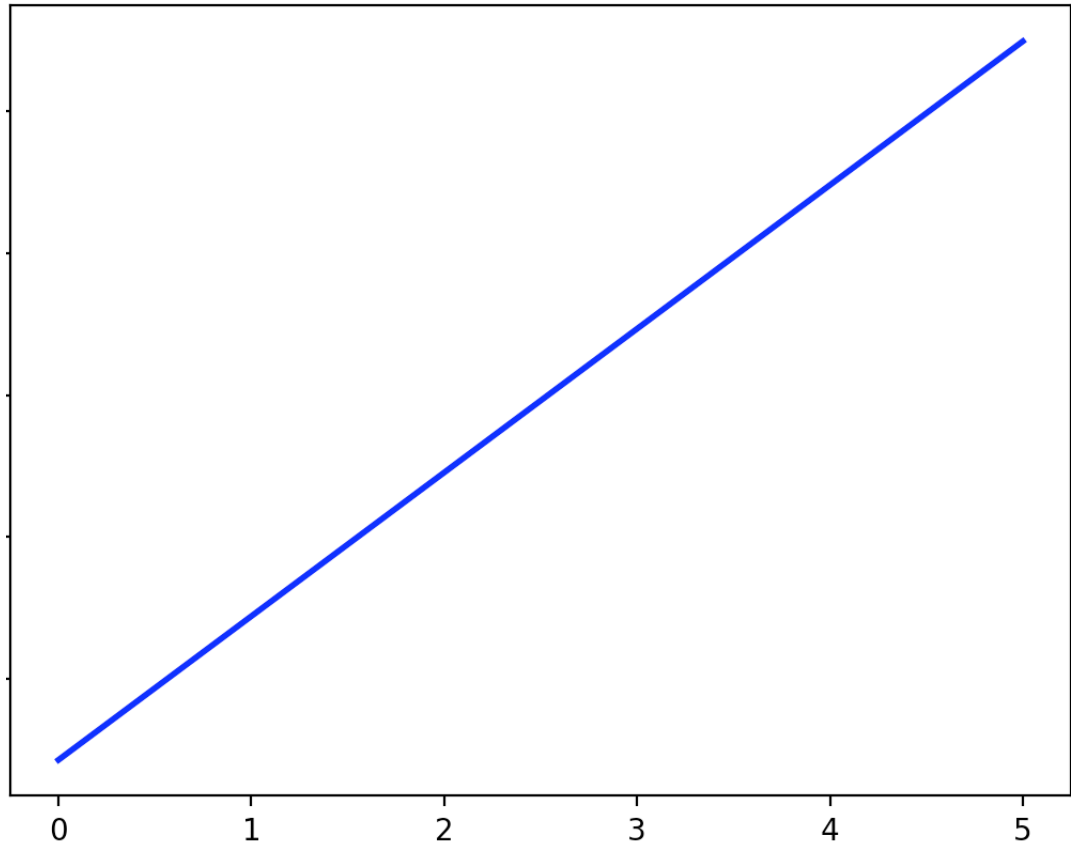
- 设 y = a0 + a1*x，我们利用最小二乘法的正则方程组来求解未知系数 a0 与 a1。
- 由于第一问处理的结果也生成了图片，因此直接进行图片处理，再使用三种方法进行拟合

```
#Part 处理图像
def img_handle(img):
    gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    gray_img = cv2.GaussianBlur(gray_img, (3,3), 0)
    _, img_binary = cv2.threshold(gray_img, 0, 255, cv2.THRESH_OTSU)
    edge = cv2.Canny(img_binary, 50, 150, apertureSize=3)
    img_binary = cv2.erode(img_binary, None, iterations=2)
    img_binary = cv2.dilate(img_binary, np.ones((5, 5), np.uint8),
iterations=2)
    img_sobel = cv2.Sobel(img_binary,cv2.CV_64F,0,1,ksize=5)
    img_sobel = abs(img_sobel)
    _,numpy = cv2.threshold(img_sobel, 500, 255, cv2.THRESH_BINARY)
    return numpy,edge

#Part 2.1 最小二值化
def least_square_method(numpy):
    row, col = numpy.shape
    numpy = numpy[int(row / 5):int(row * 4 / 5), int(col / 3):int(2 * col
/ 3)]
    row, col = numpy.shape
```

```
    cv2.imshow('o', numpy)
    x = np.linspace(0, col, col)
    y = np.array(x)
    for i in range(col):
        numpy_y = row - np.argmax(numpy[:, i])
        y[i] = numpy_y
    N = len(x)
    sumx = sum(x)
    sumy = sum(y)
    sumx2 = sum(x ** 2)
    sumxy = sum(x * y)
    A = np.mat([[N, sumx], [sumx, sumx2]])
    b = np.array([sumy, sumxy])
    a0,a1 = np.linalg.solve(A,b)
    _X = [0, 5]
    _Y = [a0 + a1 * x for x in _X]
    plt.plot(_X, _Y, 'b', linewidth=2)
    plt.title("y = {} + {}x".format(a0, a1))
    plt.show()
```

拟合直线如下



RANSAC法

- RANSAC是通过反复选择数据集去估计出模型，一直迭代到估计出认为比较好的模型。具体的实现步骤可以分为以下几步：
    1. 选择出可以估计出模型的最小数据集；(对于直线拟合来说就是两个点，对于计算Homography矩阵就是4个点)

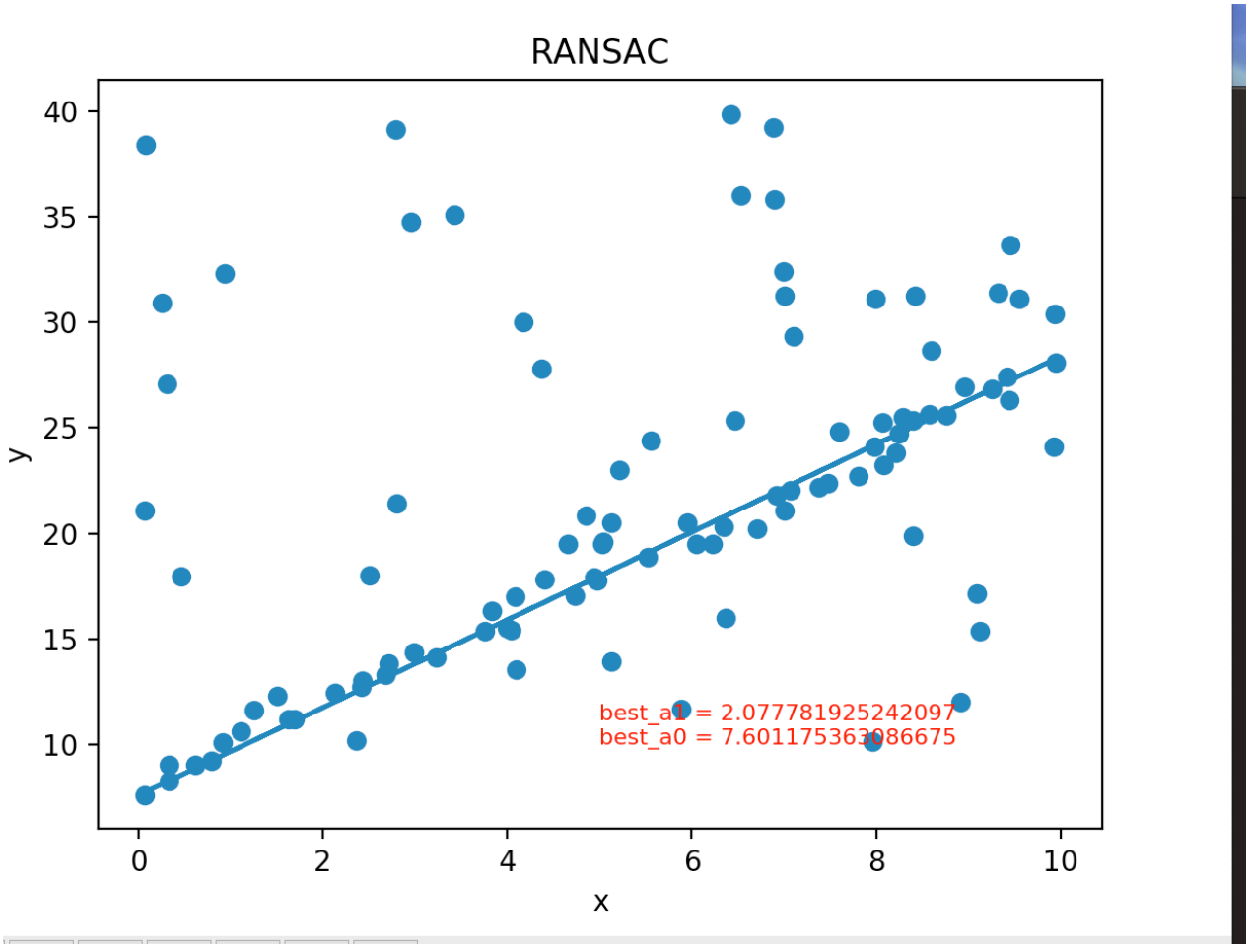
2. 使用这个数据集来计算出数据模型；
3. 将所有数据带入这个模型，计算出“内点”的数目；(累加在一定误差范围内的适合当前迭代推出模型的数据)
4. 比较当前模型和之前推出的最好的模型的“内点“的数量，记录最大“内点”数的模型参数和“内点”数；
5. 重复1-4步，直到迭代结束或者当前模型已经足够好了(“内点数目大于一定数量”)。

- 由于处理图片较为复杂，使用新设置的数据进行拟合直线。函数参数分别为：数据数量，y=ax+b 的系数a,b 迭代次数ite，数据和模型间可接受的差值sigma，希望得到的正确模型的概率p
- 先画散点图，然后迭代，随机在数据中选出两个点去求解模型，进行模型评估，求的最佳解，画图

```
#Part 2.2 RANSAC法
def RANSAC(SIZE,a,b,ite,sigma,p):
    X = np.linspace(0, 10, SIZE)
    Y = a * X + b
    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)
    ax1.set_title("RANSAC")
    # 让散点图的数据更加随机并且添加一些噪声。
    random_x = []
    random_y = []
    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))
    for i in range(SIZE):
        random_x.append(random.uniform(0, 10))
        random_y.append(random.uniform(10, 40))
    RANDOM_X = np.array(random_x)
    RANDOM_Y = np.array(random_y)
    # 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")
    best_a1 = 0
    best_a0 = 0
    pretotal = 0
    for i in range(ite):
        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]]
        a1 = (y_2 - y_1) / (x_2 - x_1)
        a0 = y_1 - a1 * x_1
        total_inlier = 0
        for index in range(SIZE * 2):
```

```
            y_estimate = a1 * RANDOM_X[index] + a0
            if abs(y_estimate - RANDOM_Y[index]) < sigma:
                total_inlier = total_inlier + 1
        if total_inlier > pretotal:
            ite = math.log(1 - p) / math.log(1 - pow(total_inlier / (SIZE * 2), 2))
            pretotal = total_inlier
            best_a1 = a1
            best_a0 = a0
        if total_inlier > SIZE:
            break
    y = best_a1 * RANDOM_X + best_a0
    ax1.plot(RANDOM_X, y)
    text = "best_a1 = " + str(best_a1) + "\nbest_a0 = " + str(best_a0)
    plt.text(5, 10, text,fontdict={'size': 8, 'color': 'r'})
    plt.show()
```

拟合直线如下



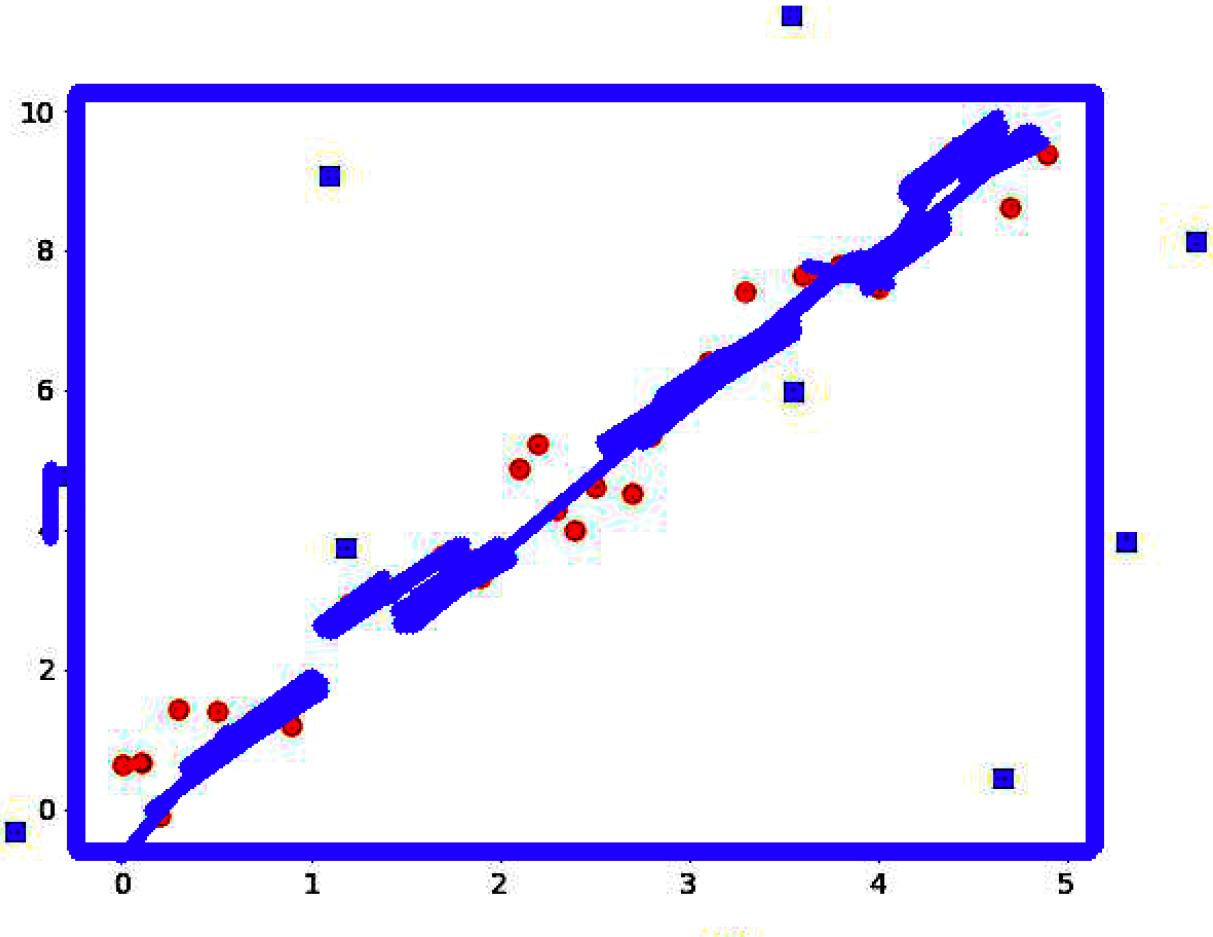
霍夫变换法

- Hesse normal form(Hesse法线式): $r = x\cos\theta + y\sin\theta$

- 其中r是原点到直线上最近点的距离(其他人可能把这记录为ρ，下面也可以把r看成参数ρ)，θ是x轴与连接原点和最近点直线之间的夹角。因此，可以将图像的每一条直线与一对参数(r,θ)相关联。这个参数(r,θ)平面有时被称为霍夫空间，用于二维直线的集合。
- 先进行图像的预处理，得到图像的边界，使用HoughLinesP检测可能的线段
  - 第一个参数是需要处理的原图像，该图像必须为cannay边缘检测后的图像
  - 第二和第三参数：步长为1的半径和步长为π/180的角来搜索所有可能的直线
  - 第四个参数是经过某一点曲线的数量的阈值，超过这个阈值，就表示这个交点所代表的参数对(rho, theta)在原图像中为一条直线
  - 第五个参数：minLineLength-线的最短长度，比这个线短的都会被忽略。
  - 第六个参数：maxLineGap-两条线之间的最大间隔，如果小于此值，这两条线就会被看成一条线

```
#Part 2.3霍夫变换法
def hough_transform(img):
    img = ImageEnhance.Contrast(img).enhance(3)
    img = np.array(img)
    _,edges = img_handle(img)
    lines = cv2.HoughLinesP(edges, 1, np.pi / 180, 10, minLineLength=30, maxLineGap=18)
    for line in lines:
        for x1, y1, x2, y2 in line:
            cv2.line(img, (x1, y1), (x2, y2), (0, 0, 255), 5)
        pass
    img = Image.fromarray(img, 'RGB')
    img.show()
```

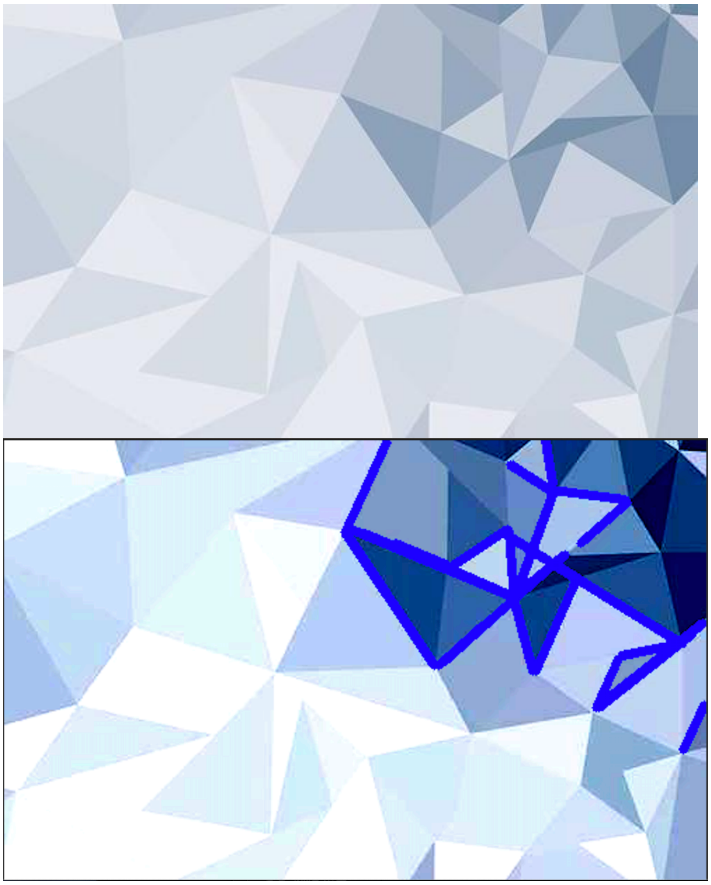
拟合直线如下



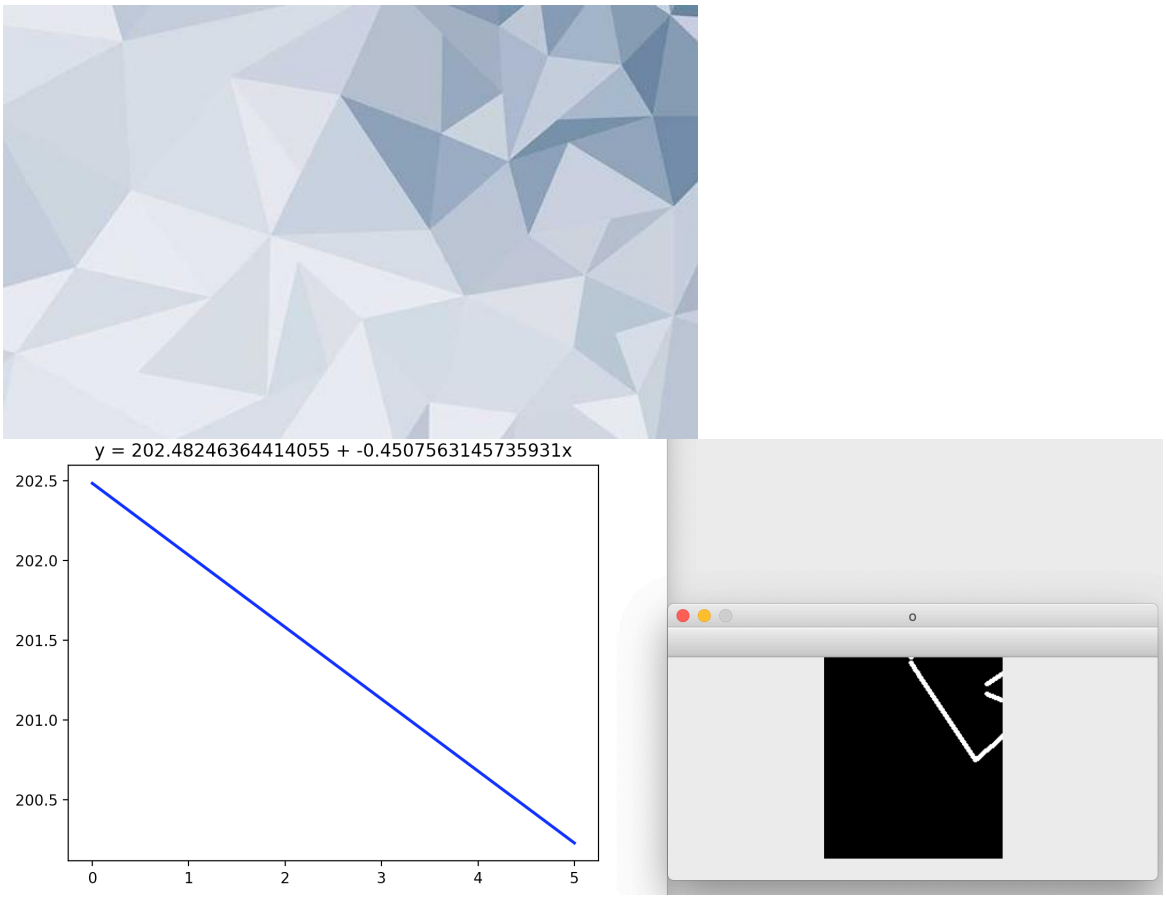
实现简单图像寻找直线

- 因为之前实验都是先进行图像预处理再进行拟合的过程，因此直接替换图片资源即可

霍夫变换法



最小二乘法


y = 202.48246364414055 + -0.4507563145735931x
202.5
202.0
201.5
201.0
200.5
0
1
2
3
4
5