Lab2：最小二乘法拟合路径

姓名：崔子寒　　学号:161220026　E-mail：161220026@smail.nju.edu.cn

目录

一、结果展示

1. 实验目的:

在本次试验中，需要从给出的 TDOA(Time-Difference-of-Arrival)数据中，计算出相应点的坐标，利用**最小二乘法**对点的坐标进行拟合，以得出室内定位系统的轨迹。为了降低作业的难度，点的坐标已经提前给出，但是其中有不少误差较大的点，需要想办法进行排除。大致的真实轨迹也已经给出。我们需要根据点的分部情况，选取适当的直线或曲线函数，利用最小二乘逼近对轨迹进行拟合。

2. 实验环境

本次实验依靠在 Window 环境下，使用 Python 语言进行实现。

其中使用 numpy 库进行科学运算，使用 matplotlib 库进行绘制散点图以及函数图像，使用 mat4py 库对 path.mat 数据文件进行解析。

Python 版本为 3.6.5。

3. 实验结果:

为了降低实验难度，已经给出了真实的轨迹，如下图所示:

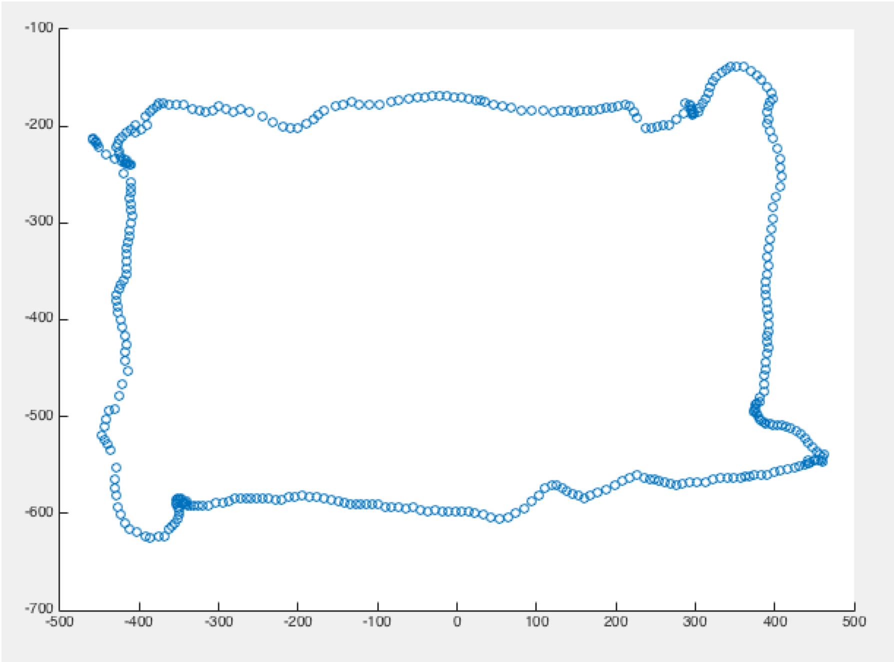


图 1： 真实轨迹

在经过异常点筛除和拟合后，我的实验结果中对于轨迹的模拟如下:

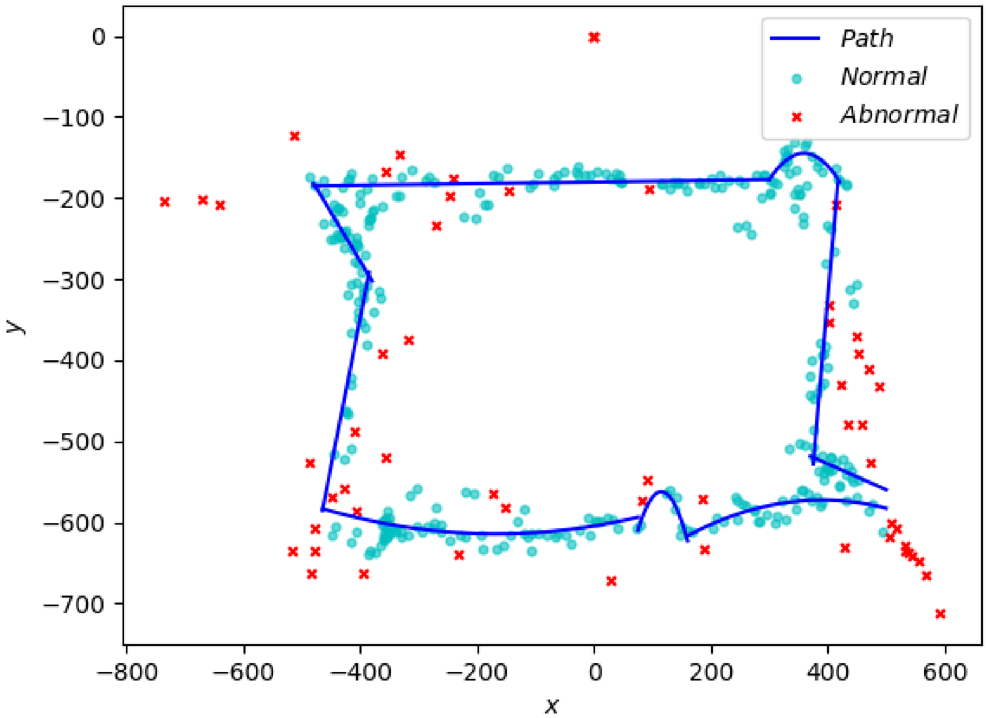


图 2：模拟结果

可以看出，右半部分拟合的结果相对左半部分更好。

拟合用到的相关函数(function.txt)如下：

1. $f(x) = 0.0003487x^2 + 0.11642997x - 604.54815518 \qquad x \in [-465, 75]$
2. $f(x) = -0.0297956x^2 + 6.846243x - 955.654685 \qquad x \in [75, 160]$
3. $f(x) = -0.00083267x^2 + 6.846243x - 699.137377 \qquad x \in [160, 500]$
4. $f(x) = -0.3197076x - 400.10933 \qquad x \in [370, 500]$
5. $f(x) = 8.2533384x - 3623.239595 \qquad x \in [-465, 75]$
6. $f(x) = -0.009371x^2 + 6.7263307x - 1351.795202 \qquad x \in [300, 420]$
7. $f(x) = 0.00979377x - 180.2293874 \qquad x \in [-480, 300]$
8. $f(x) = -1.203957x - 759.938562 \qquad x \in [-480, -380]$
9. $f(x) = 3.6717803x + 1121.82324387 \qquad x \in [-465, -385]$

二、实验过程

1. 实现思路：

1) 异常点处理

在这次实验所给出的数据点中，很多数据点是异常数据点，为了达到更好的拟合效果，需要对这些异常数据点进行判断处理。相对于正常的数据点来说，异常数据点给人的直观感受就是它是偏离轨迹的。轨迹上或者说靠近轨迹的点出现的概率较大，因此正常的数据点相对密集。异常数据点一般来说不会密集的出现，因此一种判断异常数据点的思路可以是：对于这个点，在所有点中，寻找距离它最近的点，如果这个最小距离也是大于某个阈值的，就可以判断出这个点是相对“孤立”的，因此可以依次为依据，选取恰当的阈值，将找出的“孤立”点从图中去掉。

这种思路对于大部分的异常点是有效的，但是问题在于对于一些密集出现的异常点（比如轨迹右下角），是无法给出准确的判断的，因此对于这些特殊的异

常点，需要我们手动去处理。这里我手动剔除了右下角一些明显偏出轨迹的点。

比较无异常点去除策略和有异常点去除策略的两种结果可以发现，在加入了异常点去除策略后，轨迹的拟合效果更好，尤其是矩形的四个顶角部分。

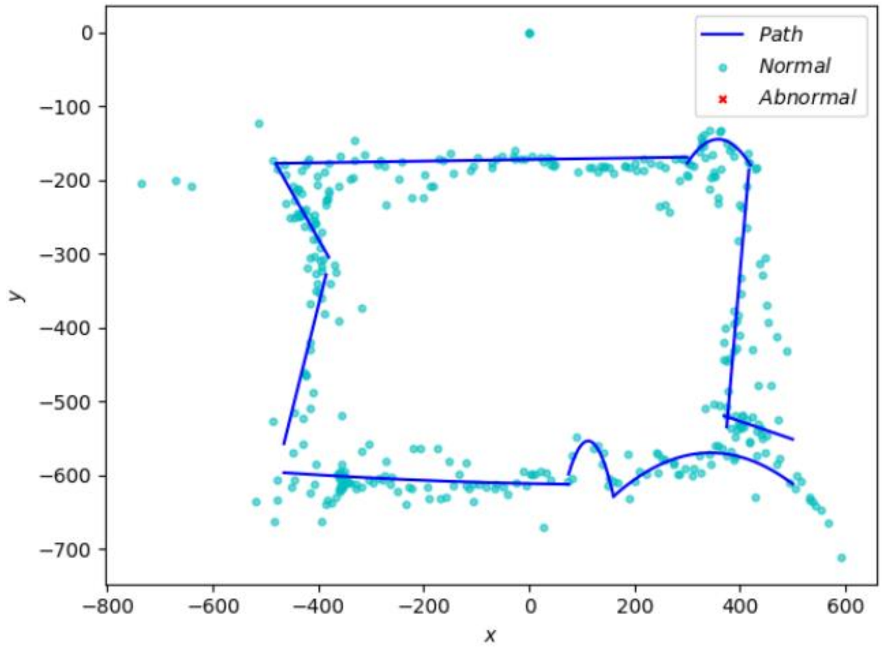


图 3：无异常点去除策略

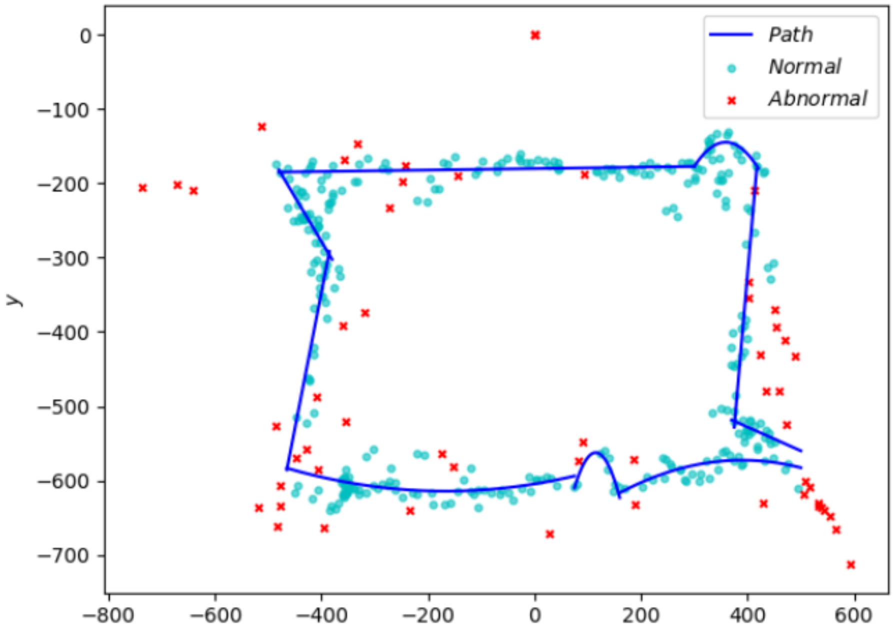


图 4：有异常点去除策略

2）函数拟合

函数的拟合部分采用最小二乘法进行拟合。

最小二乘法要求选取适当的函数，并寻找使得给定的数据点平方误差最小的函数。因此我们需要解决两个问题：(1) 选择什么类型的函数作为拟合函数。(2) 如何确定合适的参数使得平方误差最小。

对于第一个问题：选择什么类型的函数。在我们的实验背景中，路径并没有太多的“弯曲”，而且近似矩形，只有顶角处和一些特殊的地方有略微弯曲，因此我们主要选取一次函数和二次函数对数据点进行分区域拟合。首先根据不同区域中点的特点对区域做出如下的划分。

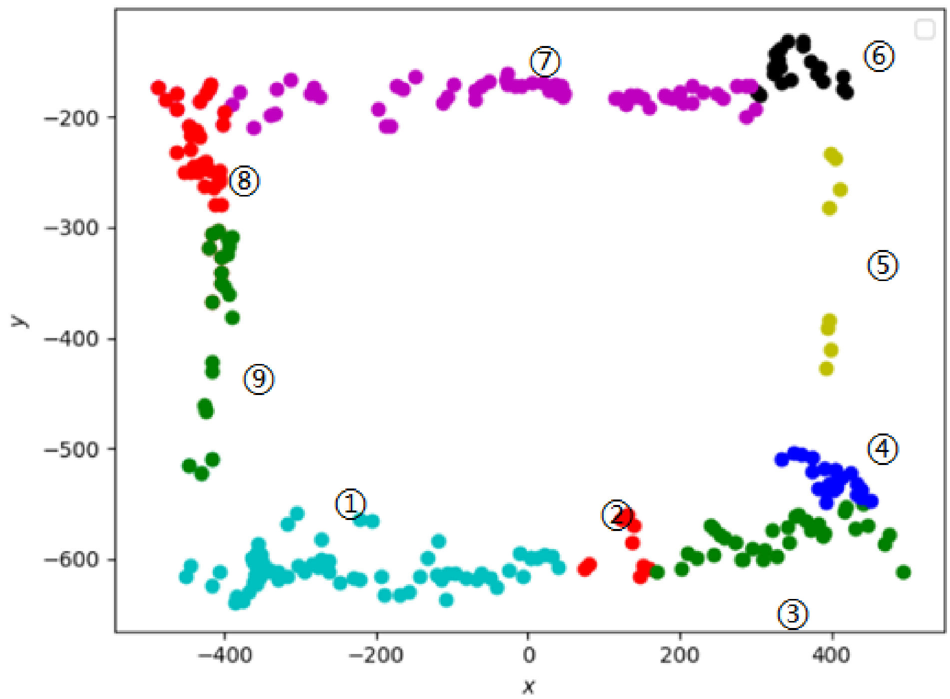


图 5： 区域划分

像区域⑤，区域⑦，区域⑨这些接近直线的轨迹，我们使用一次函数进行拟合，而其他一些有明显抛物线特性的区域，我们采用二次函数进行拟合。

对区域中的点进行划分之后，剩下的工作就是确定函数中的参数。最小二乘法要求平方误差最小，通过计算可以得到平方误差的表达式：

$$F(a_0, a_1, a_2 \dots a_n) = \sum_{k=1}^{n} [f(x_k) - y_k]^2$$

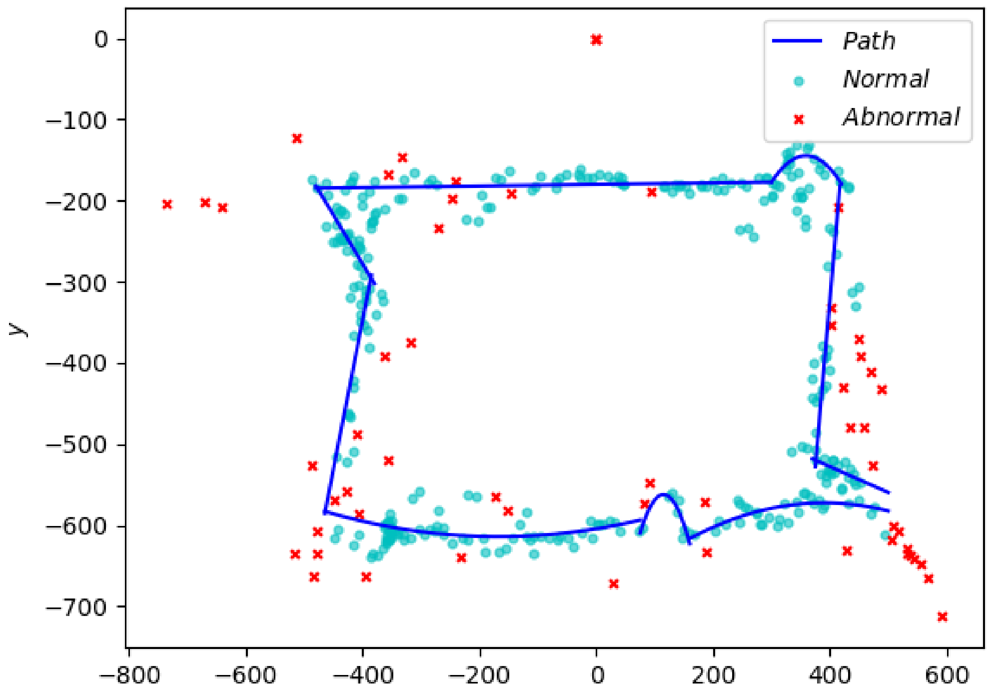
通过多元函数求极值的方法，我们分别求关于a_k的偏导数，并令它为 0，得到了法方程。

$$\sum_{j=0}^{n}\left[\sum_{i=0}^{m}\omega(x_i)\varphi_j(x_i)\varphi_k(x_i)\right]a_j = \sum_{i=0}^{m}\omega(x_i)y_i\varphi_k(x_i)$$

解之即可得参数，具体的运算过程由 numpy 中的库函数来完成，对每个区域都进行拟合后得到的函数分别为：

1. $f(x) = 0.0003487x^2 + 0.11642997x - 604.54815518 \qquad x \in [-465, 75]$
2. $f(x) = -0.0297956x^2 + 6.846243x - 955.654685 \qquad x \in [75, 160]$
3. $f(x) = -0.00083267x^2 + 6.846243x - 699.137377 \qquad x \in [160, 500]$
4. $f(x) = -0.3197076x - 400.10933 \qquad x \in [370, 500]$
5. $f(x) = 8.2533384x - 3623.239595 \qquad x \in [-465, 75]$
6. $f(x) = -0.009371x^2 + 6.7263307x - 1351.795202 \qquad x \in [300, 420]$
7. $f(x) = 0.00979377x - 180.2293874 \qquad x \in [-480, 300]$
8. $f(x) = -1.203957x - 759.938562 \qquad x \in [-480, -380]$
9. $f(x) = 3.6717803x + 1121.82324387 \qquad x \in [-465, -385]$

拟合的效果为：



2. 结果分析:

仅从视觉效果上看，拟合的结果可能还不错，下面我们对拟合的结果进行数值上的分析。计算每个拟合函数的均方误差(deviation.txt):

$$||\delta||_2 = \sqrt{\sum_{i=1}^{n}(\delta_i)^2}$$

结果如下:

区域	均方误差
①	202.1739726969598
②	29.809083415789893
③	76.59462874147809
④	44.71347214446715
⑤	172.4808684939291
⑥	46.92195358224631
⑦	92.13647614347924
⑧	271.2832434543916
⑨	237.54327015371274

从数值上看拟合效果不是很好，可能是由于点比较多，而且比较分散导致的。

3. 代码展示:

这一部分列举相关部分的代码:

（1）解析数据:

```python
import mat4py as mpy

def get_vertixs():
    data=mpy.loadmat('path.mat')
    out=open("vertixs.txt",'w+')
    # out.write(str(data))
    vertixs=list(data['path_chan'])
    for vertix in vertixs:
        out.write(str(vertix)+'\n')
```

```
    out.close()
    return vertixs
```

对于数据的解析需要用到 mat4py 库，通过调用函数 loadmat 得到字典型的数据，取键值‘path_chan’即得到所需要的点坐标。

（2）异常点处理：

```
def judge(vertix,vertixs):
    Threhold = 22*22
    length = cmath.inf
    for i in vertixs:
        if abs(i[0]-vertix[0])<100 and abs(i[1]-vertix[1])<100:
            temp = abs(i[0]-vertix[0])**2+abs(i[1]-vertix[1])**2
            if temp < length and temp !=0:
                length=temp
    if length < Threhold and -600 < vertix[0] < 500  :
        return 1
    else:
        return 0
```

异常点处理主要是找到相对偏离整体特征较大的点，即较为“孤立”的点，并将其剔除。在本次实验的数据中，经过多次尝试，认为如果一个点距离它最近的点的距离大于 22（本次实验的尺度较大），就认为这个点是异常点，并将其剔除。算法较为简单，对于给定的点，只要遍历其他点并加以判断即可。

（3）最小二乘拟合

```
import numpy as np
import matplotlib.pyplot as plt

# fit area 1
temp_x = []
temp_y = []
edge = - cmath.inf
for i in range(0,len(x)):
    if edge < y[i] < -500 and x[i] > -200:
        edge = y[i]
for i in range(0,len(x)):
```

```python
    if y[i] <= edge and x[i] < 60:
        temp_x.append(x[i])
        temp_y.append(y[i])
res = np.polyfit(temp_x, temp_y, 2)
f1_x = np.linspace(-465, 75, 10000)
f1_y = [res[0]*i**2+res[1]*i+res[2] for i in f1_x]
#plt.scatter(temp_x,temp_y,color='c',marker='o')
plt.plot(f1_x, f1_y, color='b', label=r'$Path$',linestyle='-')
output(res,[-465,75])
```

最小二乘拟合的科学计算依靠 numpy 提供的库函数 numpy.ployfit 进行多项式拟合。这里以 area1 的拟合为例。

首先需要选定 area1 的具体范围，将其中的点添加到 temp_x 和 temp_y 中，然后调用 polyfit 进行拟合，这里因为 area1 的估计有二次函数的特征，因此采用二次函数进行拟合。最后在 area1 上将拟合的结果用 plot 函数绘制出。对于其他区域的操作也是如此，这里不再重复说明。

（4）误差计算

```python
def deviation(a,x,y):
    dev=0
    if len(a)==3:
        for i in range(0,len(x)):
            dev+=(a[0]*x[i]**2+a[1]*x[i]+a[2]-y[i])**2
        dev_out.write(str(dev**0.5)+'\n')
    else:
        for i in range(0,len(x)):
            dev+=(a[0]*x[i]+a[1]-y[i])**2
        dev_out.write(str(dev**0.5)+'\n')
```

误差计算按照公式计算即可，结果保存在 deviation.txt 中。