**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](mailto:161220026@smail.nju.edu.cn)

目录

[一、结果展示 2](#_Toc515007407)

[1. 实验目的： 2](#_Toc515007408)

[2. 实验环境 2](#_Toc515007409)

[3. 实验结果： 2](#_Toc515007410)

[二、实验过程 4](#_Toc515007411)

[1. 实现思路： 4](#_Toc515007412)

[2. 结果分析： 8](#_Toc515007413)

[3. 代码展示： 8](#_Toc515007414)

# 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：

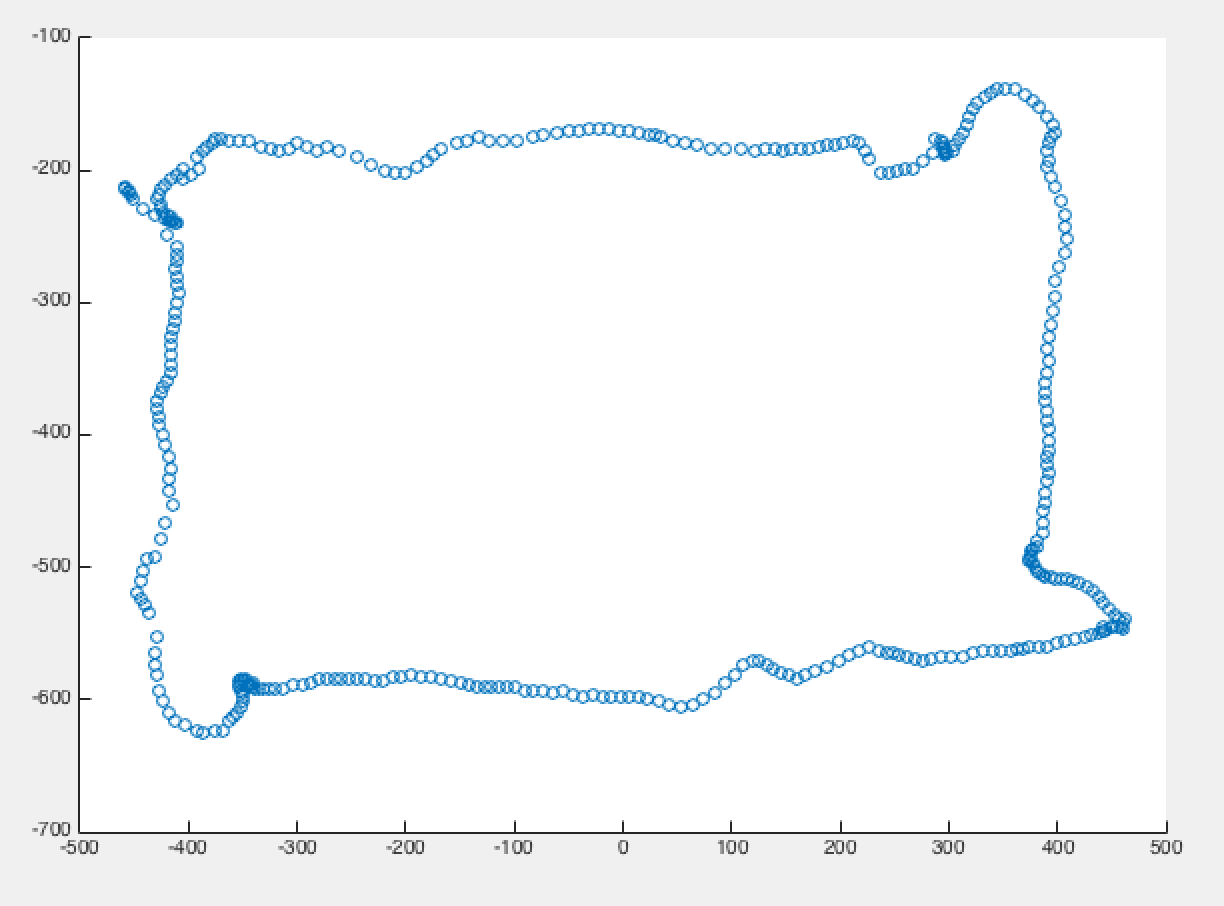


图1： 真实轨迹

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：

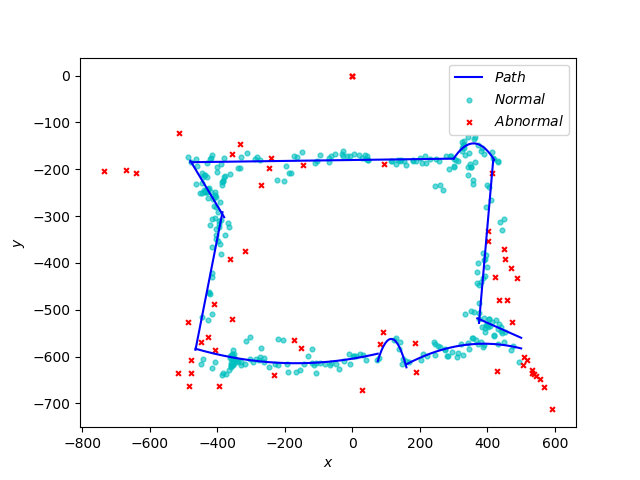


图2：模拟结果

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](实验结果/function.txt))如下：

# 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。

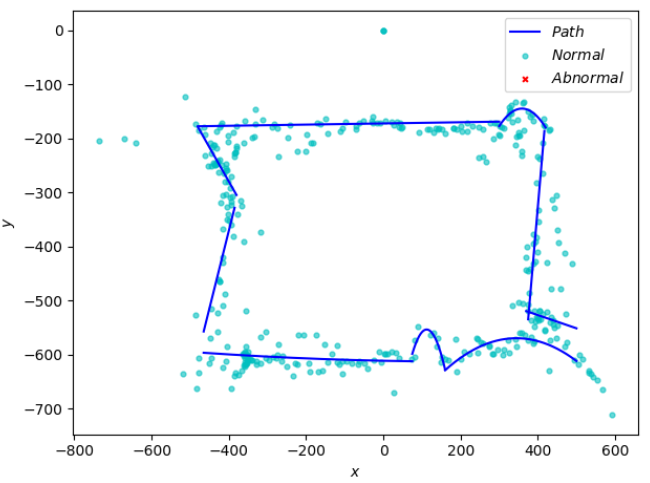


图 3：无异常点去除策略

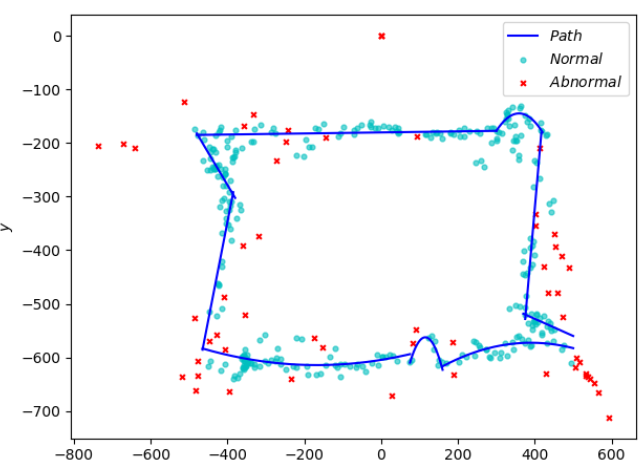


图 4：有异常点去除策略

**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。

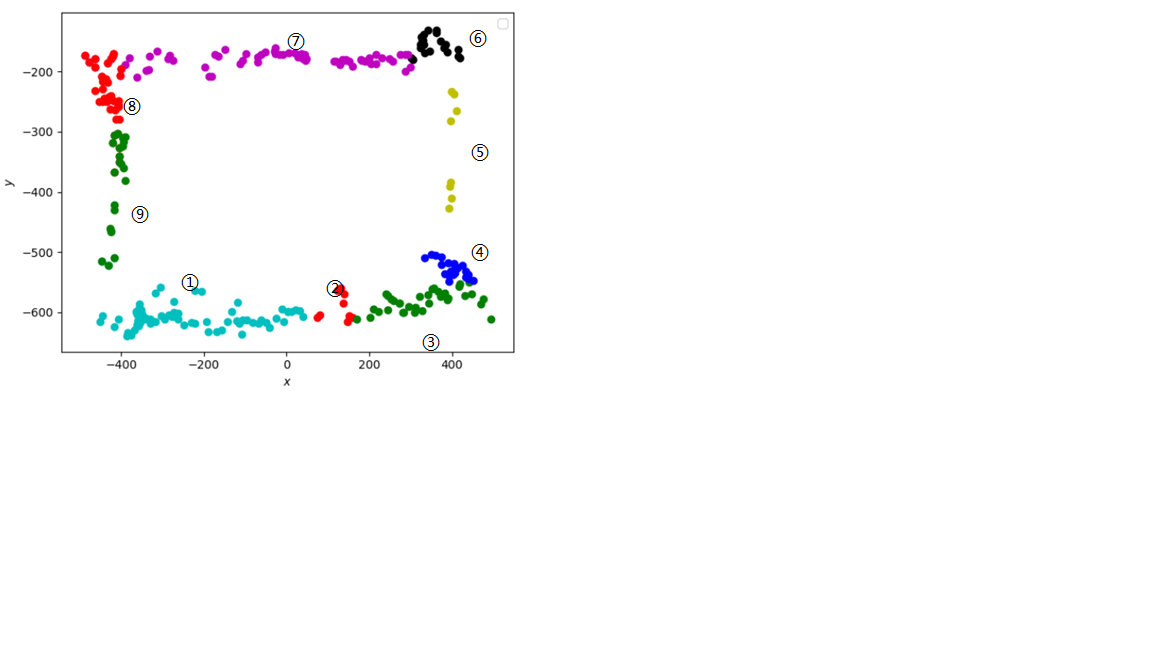
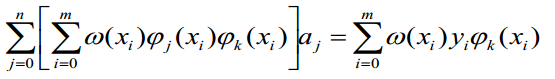


图5： 区域划分

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。

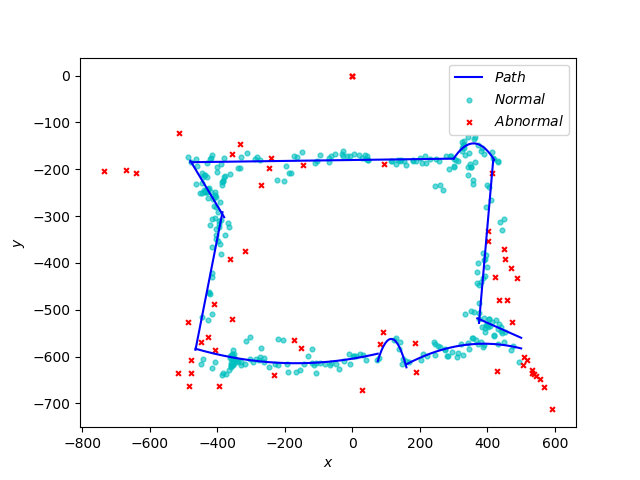
对区域中的点进行划分之后，剩下的工作就是确定函数中的参数。最小二乘法要求平方误差最小，通过计算可以得到平方误差的表达式：

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



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：

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](实验结果/deviation.txt)):

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：

1. **import** mat4py as mpy
3. **def** get\_vertixs():
4. data=mpy.loadmat('path.mat')
5. out=open("vertixs.txt",'w+')
6. # out.write(str(data))
7. vertixs=list(data['path\_chan'])
8. **for** vertix **in** vertixs:
9. out.write(str(vertix)+'\n')
10. out.close()
11. **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：

1. **def** judge(vertix,vertixs):
2. Threhold = 22\*22
3. length = cmath.inf
4. **for** i **in** vertixs:
5. **if** abs(i[0]-vertix[0])<100 **and** abs(i[1]-vertix[1])<100:
6. temp = abs(i[0]-vertix[0])\*\*2+abs(i[1]-vertix[1])\*\*2
7. **if** temp < length **and** temp !=0:
8. length=temp
9. **if** length < Threhold **and** -600 < vertix[0] < 500  :
10. **return** 1
11. **else**:
12. **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

1. **import** numpy as np
2. **import** matplotlib.pyplot as plt
4. # fit area 1
5. temp\_x = []
6. temp\_y = []
7. edge = - cmath.inf
8. **for** i **in** range(0,len(x)):
9. **if** edge < y[i] < -500 **and** x[i] > -200:
10. edge = y[i]
11. **for** i **in** range(0,len(x)):
12. **if** y[i] <= edge **and** x[i] < 60:
13. temp\_x.append(x[i])
14. temp\_y.append(y[i])
15. res = np.polyfit(temp\_x, temp\_y, 2)
16. f1\_x = np.linspace(-465, 75, 10000)
17. f1\_y = [res[0]\*i\*\*2+res[1]\*i+res[2] **for** i **in** f1\_x]
18. #plt.scatter(temp\_x,temp\_y,color='c',marker='o')
19. plt.plot(f1\_x, f1\_y, color='b', label=r'$Path$',linestyle='-')
20. 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

1. **def** deviation(a,x,y):
2. dev=0
3. **if** len(a)==3:
4. **for** i **in** range(0,len(x)):
5. dev+=(a[0]\*x[i]\*\*2+a[1]\*x[i]+a[2]-y[i])\*\*2
6. dev\_out.write(str(dev\*\*0.5)+'\n')
7. **else**:
8. **for** i **in** range(0,len(x)):
9. dev+=(a[0]\*x[i]+a[1]-y[i])\*\*2
10. 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](实验结果/deviation.txt)中。