中国研究生创新实践系列大赛
“华为杯”第十八届中国研究生数学建模竞赛

学　　校	西安交通大学
参赛队号	**21106980143**
队员姓名	**1.代明昊**
	2.张德润
	3.陈钦隆

中国研究生创新实践系列大赛
“华为杯”第十八届中国研究生
数学建模竞赛

题 目　信号干扰下的超宽带精确定位的优化建模

摘　　要：

针对信号干扰下的超宽带精确定位问题，本文建立了数据预处理模型、三维定位预测模型、误差校正模型以及干扰识别分类模型，运用遗传算法、三边定位法、BP 神经网络法以及 Kalman 滤波等方法，结合靶点定位误差大小与有无干扰间的对应关系，探究了给定条件下，采用有干扰和无干扰数据的精确定位预测问题，建立了干扰识别的量化分类判据，计算了静态及动态靶点的预测位置坐标，并给出了动态靶点的运动轨迹。

针对问题一，为实现数据的准确预处理，本文建立了包含抓取和清洗两部分的数据预处理模型。首先基于 Pandas 工具库建立数据抓取模型，提取出原始数据中的锚点测距值等关键数据，并以矩阵形式保存；之后创建基于 3σ 准则的数据清洗模型，制定“无用”数据判定标准，结合数据特征针对正常、异常数据分别清洗，得到有效样本数据集；进而依据所建模型完成全部数据文件的预处理，以 24.正常、109.正常、1.异常、100.异常四个数据文件为例，经预处理后，分别保留了 147、273、374、300 个有效数据样本；为提高后续定位模型精度，本文对所得数据集进行了深化处理，通过均值滤波方法提取数据关键特征，处理后每组正常和异常数据个数分别 1 和 2；并采用回归拟合方法校正测量系统误差。

针对问题二，为得到靶点位置的精确定位，本文分别建立了基于正常和异常数据的定位预测模型。首先基于遗传算法建立正常数据定位模型，算法中加入锚点坐标以体现场景信息；之后考虑到异常数据中锚点测距值存在误差，建立基于遗传算法-三边定位耦合算法的定位模型，融合了两种算法的寻优和定位误差计算功能；再后基于 BP 神经网络建立误差修正模型，提升定位预测坐标的精度；为验证模型有效性，本文采用 *MAE* 和 *RMSE* 两个指标分别衡量模型预测结果的 3 维、2 维和 1 维精度，以 3 维计算结果的最大误差为例，正常和异常数据定位模型的 *RMSE* 分别为 138.4702mm 和 315.5685mm，保证了模型的厘米级预测精度；最后利用定位模型对测试数据集进行定位预测，得到前 5 组无干扰数据点空间预测坐标分别为(1139.82, 6008.49, 754.99)、(3175.25, 1686.72, 814.05)、(2737.95, 1128.53, 858.83)、(2453.46, 962.2, 2255.99)、(1451.49, 2538.42, 1895.17)；后 5 组有干扰数据点空间预测坐标分别为(2029.63, 718.43, 512.82)、(4113.92, 1789.25, 1971.96)、(1623.85, 1099.21, 1152.48)、(3540.53, 1965.29, 2316.68)、(4698.75, 2171.56, 1434.10)。

针对问题三，为实现定位模型在不同实际场景下的应用，本文通过改变定位模型中的锚点坐标实现应用场景迁移，并分析“正常数据”、“异常数据”以及误差校正模型所受场景的影响，说明模型不同场景下的适应性；最后利用定位模型对场景 2 下采集得到的数据进行定位计算，说明所建模型的泛化能力良好，计算出前 5 组无干扰数据点的预测坐标分别为(3664.66, 2204.42, 1207.08)、(4233.57, 1723.86, 1062.79)、(3195.48, 1742.96, 1141.67)、(2554.11, 1917.99, 2204.9)和(500.29, 61.29, 523.24)；后 5 组有干扰数据点的预测坐标分别为(4606.71, 2204.40, 1300.66)、(2364.66, 1716.89, 730.41)、(1846.15, 1600.9, 964.7)、(2097.84,

669.32, 837.22)和(1099.24, 674.04, 852.35)。

针对问题四，为实现数据有无干扰的精准识别，本文基于三边定位方法建立干扰识别分类模型。首先采用三边定位模型计算附件 1 中有、无干扰存在时的数据误差，获取大量数据特征，得到干扰与定位预测误差之间的相关关系；之后以干扰识别错误数最少为目标函数，建立单目标优化模型，给出判断正确率最高时的误差临界值，由此制定干扰识别量化判据，认为定位误差高于临界值认为数据受到干扰，反之不受干扰；再对该分类模型的有效性进行验证，本文以同场景下附件 2 数据为测试集进行干扰识别，模型识别干扰成功的概率高达 90%；最后利用模型对附件 4 数据进行干扰识别，识别出 5 个数据点受到干扰，受干扰数据点编号分别为：2、4、5、6、7。

针对问题五，为实现动态靶点的运动轨迹定位，本文综合运用前文所建定位预测和干扰识别分类模型，识别出靶点运动所受到的随机干扰，计算其各时刻预测位置，初步绘制出运动轨迹图；之后采用 Kalman 滤波方法消除运动过程中非视距误差对定位结果的影响，实现对原始预测位置数据的降噪处理；最后采用光滑连续的曲线连接处理后的靶点运动轨迹位置坐标，绘制出靶点的动态运动轨迹。

关键词：UWB 定位 遗传算法 三边定位 神经网络 Kalman 滤波

目 录

一、问题背景与重述分析

1.1 问题背景

随着互联网及物联网技术的发展，室外及室内定位的需求与日俱增，而 GPS 等卫星导航系统难以有效应用到室内环境之中，当前已有许多用以实现室内定位的技术得到发展，包括蓝牙定位、WiFi 定位、射频识别定位、红外定位、超声波定位以及超宽带（UWB）定位技术等[1,2]。其中，超宽带技术是一种无线通讯技术，又称脉冲无线电技术，无需任何载波，可以通过发送纳秒级窄脉冲来完成数据传输实现定位，数据传输速率可达几百 Mbit/s，被认为是具有革命性意义的无线电技术进展，于 2002 年被美国联邦通信委员会批准可以用于民用[3]。超宽带技术具备实时室内外精确跟踪能力，具有穿透力强、时钟分辨率高、功耗较低、抗多径干扰能力强等优势[4]，对定位服务起到了很好的补充作用，在军事及民用等领域都有着广泛的应用，应用场景包括：国家电力工程、司法监狱、铁路交通、工厂工地、大型商超、酒店及会展中心等[5]。

UWB 定位技术可以应用多种方法，包括基于接收信号强度的定位方式（RSSI）和基于时间或角度参数的定位方式，后者根据所选参数，可以分为：信号到达角度方式（AOA）、信号到达时间方式（TOA）、信号到达时间差方式（TDOA）以及信号双向传输时间方式（TOF）[6]。UWB 定位方法之中，基于飞行时间的测距原理（TOF）是最常见的方法之一，该方法属于双向测距技术，通过计算信号在锚点和靶点间的飞行时间乘以光速求得两者之间的距离，该方法下的定位精度可以达到厘米级。然而，复杂多变的室内环境有时会对 UWB 信号产生较强干扰，导致数据传输发生异常波动，以至于产生较大定位误差，甚至造成严重事故。因此，在信号干扰下的条件下，实现超宽带（UWB）精确定位的问题亟待解决。

1.2 问题提出

问题一：数据预处理问题。实际场景测试中，在信号无干扰/有干扰情况下分别采集得到“正常数据”和“异常数据”，首先建立数据抓取模型，从原始数据集中提取有效数据，并把数据转换成矩阵形式，其中一行代表一个样本数据；其次建立数据预处理模型，确立“无用”数据的判定标准；最后输出所有已处理的“正常数据”及“异常数据”数据文件。

问题二：定位模型建立问题。首先结合问题一筛选得到的数据文件，分别针对“正常”与“异常数据”建立体现实验场景信息的靶点定位模型；其次选择合适的误差计算方法，确定所建靶点定位模型的 3 维、2 维及 1 维精度，以此说明模型有效性；最后使用定位模型对题中所给无干扰和有干扰的数据进行精准定位，给出靶点 3 维坐标。

问题三：定位模型不同场景应用问题。首先在问题二已建模型的基础上，研究定位模型在不同实验场景适用性情况，针对无干扰定位模型和有干扰定位模型分别进行分析，以此说明模型的泛化能力；之后利用定位模型对实验场景 2 采集得到的 10 组数据进行计算，得出相应靶点的坐标值，实现模型在不同场景中的应用。

问题四：干扰识别分类问题。首先建立干扰识别模型，展示出有无干扰对 UWB 采集数据特征的影响；之后创建干扰识别判据，给出量化指标判别是否存在干扰；再结合题中所给测试数据分析所建干扰识别模型的有效性；最终使用所建干扰识别模型对附件 4 中 10 组数据进行分类，给出信号有无干扰的判断结果。

问题五：动态靶点运动轨迹定位问题。首先根据静态点的定位模型，结合靶点自身运动规律对靶点的坐标进行定位，初步绘制运动轨迹图；之后考虑动态数据采集信号之时，会随机受到扰动的影响，建立降噪模型，减小随机误差影响；最后绘制靶点动态轨迹图。

1.3 问题分析

对于问题一，为实现对场景实测数据的科学预处理，本文拟分别建立数据的抓取模型和数据清洗模型。首先建立基于 Python 编码调用 Pandas 工具库的数据抓取模型，将原始数据中的有效数据提取出来，转换成矩阵形式并以 Excel 格式保存；为了建立精准的数据清洗模型，先创建“无用”数据的判定准则，拟以数据异常、缺失、相同或相似为“无用”，再确立数据筛选的方法，拟采用拉依达（3σ）准则进行异常数据的剔除；再使用所建判定准则结合数据筛选方法对数据进行清洗，得到仅包含有效样本的数据文件予以保存；最后采用均值滤波、回归拟合等方法对数据进行深化处理，为后续模型的建立奠定基础。

对于问题二，为给出靶点位置的精准定位，本文考虑针对“正常数据”与“异常数据”分别建立定位预测模型。遗传算法具备应用范围广、自适应能力强、鲁棒性高等特点，首先建立基于遗传算法的“正常数据”定位模型，并在模型中加入锚点坐标以体现实际场景信息；考虑到“异常数据”存在误差，拟采用遗传-三边定位耦合算法对模型进行改进，实现先由三边定位算法预测位置并计算误差、后由遗传算法进行寻优，实现对“异常数据”的精准定位；之后拟采用 BP 神经网络对模型定位坐标的误差进行修正，以 MAE 和 RMSE 作为评价指标完成模型有效性验证；最后利用所建模型实现测试数据集的准确定位。

对于问题三，为实现所建定位模型在不同实际场景下的应用，本文拟通过改变定位模型中的锚点坐标实现模型应用场景的迁移；考虑到实际场景的变换存在影响模型准确性的可能，拟通过分别分析“正常数据”、“异常数据”以及误差校正模型所受场景的影响，说明模型在不同场景下的适用性；最后利用定位模型对场景 2 下采集得到的数据进行定位计算，得到相应靶点的预测坐标值，体现模型的泛化能力。

对于问题四，为实现数据有无干扰的精准识别，本文拟基于三边定位算法建立干扰识别分类模型，充分考虑距离测量精度对三边定位算法预测误差的影响，提取是否存在干扰与定位预测误差之间的特征关系；之后以误差临界值两边判断识别错误数最少为目标函数，建立单目标优化模型，确定出判断正确率最高时的误差临界值，由此给出干扰识别判据，建立量化分析指标；再后采用同场景下附件 2 数据作为测试集，拟对干扰识别分类模型的有效性进行验证；最后使用模型对附件 4 数据集进行干扰识别，给出判断结果。

对于问题五，为实现动态靶点的运动轨迹定位，本文拟运用前文已建立的定位预测和干扰识别分类模型，计算出各静态点的预测位置，初步绘制出运动轨迹图；之后拟采用 Kalman 滤波方法消除运动过程中非视距误差对定位结果的影响，实现对原始预测位置数据的降噪处理；最后由处理后的位置坐标数据绘制出靶点的动态运动轨迹，拟采用光滑连续的曲线将靶点运动轨迹上各点连接，表征三维空间内的运动情况。

二、模型假设

1. 假设信号存在干扰时，同一时间仅有一个锚点被干扰，即另外三个锚点数据正常；
2. 假设本问题中的信号接收不存在多路径效应，即不考虑各种反射和折射信号的影响；
3. 假设不考虑信号发射与接收所带来的误差，仅考虑信号异常波动所引起的误差；
4. 假设锚点位置的变化不影响实际距离与测试距离间标定函数的准确性。

三、主要符号说明

本文所涉及的主要变量符号及其含义如表 3-1 所示。

表 3-1 符号说明

符号	含义	单位
d	测量距离	mm
h	估计距离与测量距离误差和函数	mm
L	估计距离与测量距离误差	mm
P	定位标签坐标值	mm
x	坐标轴	mm
y	坐标轴	mm
z	坐标轴	mm
μ	样本均值	mm
σ	样本方差	mm^2
MAE	平均绝对误差	mm
RMSE	均方根误差	mm

四、问题一模型的建立与求解

为实现数据的准确预处理，本文建立了包含数据抓取和数据清洗两部分的预处理模型，使用 Pandas 工具库进行数据批量抓取，创建“无用”数据判定标准结合 3σ 准则针对“正常、异常数据”分别清洗，得到有效数据集；此外，通过均值滤波、回归拟合的方法，对筛选所得数据集进行了深化处理，为后续定位模型及干扰识别模型的建立奠定基础。

4.1 数据预处理模型

获取真实有效的数据是后续模型建立的基础，题目中提供的 UWB 数据集是原始数据，其中包含有无效和错误的数据信息，且原始格式不利于后续模型的调用计算，需要对其进行规整。本文所建数据预处理模型包含两个部分，分别是数据抓取模型和数据清洗模型，前者提取有效数据并保存为矩阵形式，后者去除矩阵数据集中的“无用”信息。

4.1.1 数据抓取模型的建立

题目中所给数据的格式是统一的，数据文件内首行无实际意义，从第 2 行开始以 4 行数据为一组形成一个样品，数据的具体格式如下：

$$\begin{aligned}
&T:144235622:RR:0:0:950:950:118:1910\\
&T:144235622:RR:0:1:2630:2630:118:1910\\
&T:144235622:RR:0:2:5120:5120:118:1910\\
&T:144235622:RR:0:3:5770:5770:118:1910
\end{aligned} \tag{4-1}$$

从式（4-1）中所展示的数据格式可以看出，一行中共有九列数据，数据间采用冒号进行分隔。第 1 列至第 9 列的数据分别表示：Tag 标识、时间戳、缩写（Range Report）、Tag ID、锚点 ID、该锚点测距值、测距值校验值、数据序列号以及数据编号。

本文采用 Python 语言编程实现数据抓取模型的建立，实现对大量数据的处理分析。模型主要采用 Pandas 工具库进行数据分析与处理，主要是对字符串的正则表达式进行匹配，筛选和提取的。模型实现循环读入所有文档中的原始数据并忽略第一行无用数据，采用冒号对其余数据进行分割并将各列数据进行编号，随后以 4 行为一组，将各组中的有用列对应的数据提取出，包括时间戳、锚点 ID、测距值和测距值校验值。最后将每组数据放一行代表一个样品数据，输出数据文件为 Excel 格式。具体代码见附录 1。

4.1.2 数据清洗模型的建立

1）“无用”数据判定准则

对 4.1.1 所抓取的数据进行清洗，筛选掉无用信息，首先需建立数据判定准则来明确何为“无用”数据。题目中给出的判据是数据异常、缺失、相同或相似为“无用”。本文即据此来建立“无用”数据判定准则，为下步工作奠定基础。

数据异常判定：本文认为数据集中趋势、偏差明显异于其它的值为异常值，例如产生突变的值，偏差巨大的值等。该种异常数据可以认定为粗大误差，对模型影响巨大，应予以剔除。处理该种“无用”数据可以采用拉依达准则进行判断。

数据缺失判定：在原始数据中每行应有 9 列数据，若任何一列数据有缺失，则认定该组数据属于数据缺失，予以舍弃。特别要注意锚点与靶点的测距值缺失的情形，该种情形会直接影响数据的准确性。

数据相同判定：本文认定原始数据中的两组数据，如果其九列数据相应完全相同，则该两组数据为相同数据，此种情形属于数据统计重复，应予以舍弃一组。

数据相似判定：对原始实验数据的处理而言，异常值应予以舍弃，正常值予以保留。认为异常值的相似数据也为异常，予以删除，正常数据的相似数据为正常，予以保留。

2）拉依达（3σ）准则去除异常值

拉依达准则，又称为 3σ 准则，是一种常见的可疑数据处理方法，但仅适用于处理正态分布或近似于正态分布的样本分布，且仅适用于数据量较大的样本数据。对于测量次数 $n\geq10$，数据服从正态分布，并且$\{x_i\}$（$i = 1, 2, \cdots, n$）的算术平均值 μ 和标准差 σ 可以计算的情况下，可以针对$\{x_i\}$构建控制限为 3σ 的区间（μ-3σ, μ+3σ）进行异常值识别[7]。定义在该区间范围内的数据为正常数据，其余为异常数据，此时的置信概率为 99.73%，能够认为保留下来的数据满足统计特性，能够有效判别大偏移数据在数据集中的的异常值存在情况[8]。对于数据集$\{x_i\}$（$i = 1, 2, \cdots, n$），已知其算术平均值为 μ，可以求得其剩余误差为 $v_i = x_i - \mu$（$i = 1, 2, \cdots, n$），则可根据贝塞尔公式计算出数据的标准差 σ，公式如下：

$$\sigma=\left[\frac{1}{n-1}\sum_{i=1}^{n}v_i^2\right]^{\frac{1}{2}}=\left\{\left[\sum_{i=1}^{n}x_i^2-\left(\sum_{i=1}^{n}x_i\right)^2/n\right]/(n-1)\right\}^{\frac{1}{2}} \tag{4-2}$$

3）数据清洗流程

已知存在信号正常和受干扰两种工作状况。可以认为在信号不受干扰时，测距误差是由 TOF 测距方法中的时钟漂移和测距延迟导致的[9]，其具备正态分布的特征，可以直接依据 3σ 原则对其中的异常数据进行剔除。而对于信号受障碍物影响的情形，由于锚点发出的信号被遮挡，导致传输时间延后，因而靶点处接收信号处理得到的两点距离会相较真实值有一定程度的增大。考虑到问题中的每一个异常工作点，其靶点精确的位置坐标以及正常工作情况下测得的到每个锚点的距离值是已知的。依此，我们可以根据真实的距离大小及信号正常工作条件下的测量值，对信号受干扰时产生的异常距离信息进行分辨、剔除。而对剩余的数据，同样可以根据 3σ 原则，清理其中超出偶然误差范围的部分。

结合上文对“无用”数据的判定准则以及 3σ 准则去除异常值的方法，即可对数据进行清洗。考虑到数据存在无干扰和有干扰情况下测量的区别，本文针对有/无干扰的数据分别建立了清洗流程进行处理，汇总成图 4-1 所示的数据清洗流程

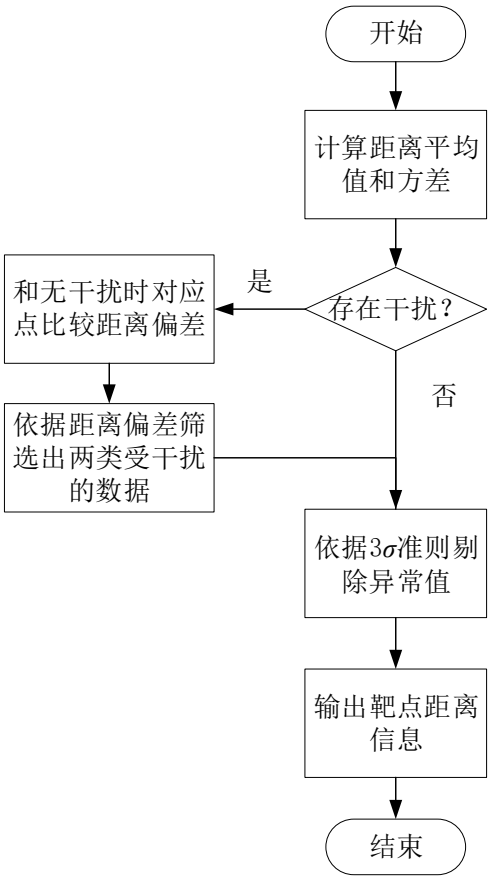


图 4-1 数据清洗流程

如图 4-1 所示，数据清洗之前，首先判断数据的测量条件是否存在干扰，对于无干扰的正常数据，一个样本数据包含四个锚点的测距值，根据锚点的不同将全部样本测距值分为 A0、A1、A2、A3 四组，分别求取四组数据的算术平均值 μ 和标准差 σ，即可依据 3σ 准则剔除异常点，样本数据中任何一个测距值异常，认为该样本异常，予以剔除。对于存在干扰的异常数据，首先对比四组测距值的方差大小，以此区分出干扰锚点与非干扰锚点，然后进行三轮筛选：（1）统计发现绝大部分样本中仅有一个锚点测距值异于真实值，因此假设存在信号干扰时，同一时间仅有一个锚点被干扰，将存在两个及以上锚点数据异常的样本数据剔除；（2）其次，认为异常数据中有且仅有一个锚点测距值异于真实值，需将四个锚点测距值全部正常的样本数据予以剔除，采用正常数据的控制限区间（μ-3σ, μ+3σ）进行筛选，四个锚点数据均落入该区间范围的点予以剔除；（3）最后将剩余样本数据根据受影响锚点的不同分成不同组别，再分别对这些数据计算算术平均值 μ_1 和标准差 σ_1，依据 3σ 准则进行筛选，删除落在控制限区间（μ_1-3σ_1, μ_1+3σ_1）之外的样本数据，并将存留下来的样本数据进行保存。

4.2 数据预处理结果

4.2.1 数据抓取结果

靶点数据在每一个位置都分别在正常及异常工况下各自采集了一组数据，总共有 648 个结果文件，依据题目要求，筛选后的数据表示为矩阵形式，矩阵的每一行即为一组数据样品（即包含某一时间戳下某靶点到四个锚点的测距结果）。据此，筛选掉数据文件中无意义的成分，这些成分包含每个数据文件的开头行、靶点标识符号、锚点 ID、靶点 ID、数据序列号以及数据编号。同时，经过数据比较，发现测距值和测距校正值完全没有偏差，说明该 TOF 测距设备的调零、标定情况良好，所以可略去每一组数据中测距校验值的信息。

由以上分析，依据附录 1 的程序，完成数据的抓取，展示抓取后、尚未筛除异常值的第 109 个点的正常组数据抓取矩阵展示如下：

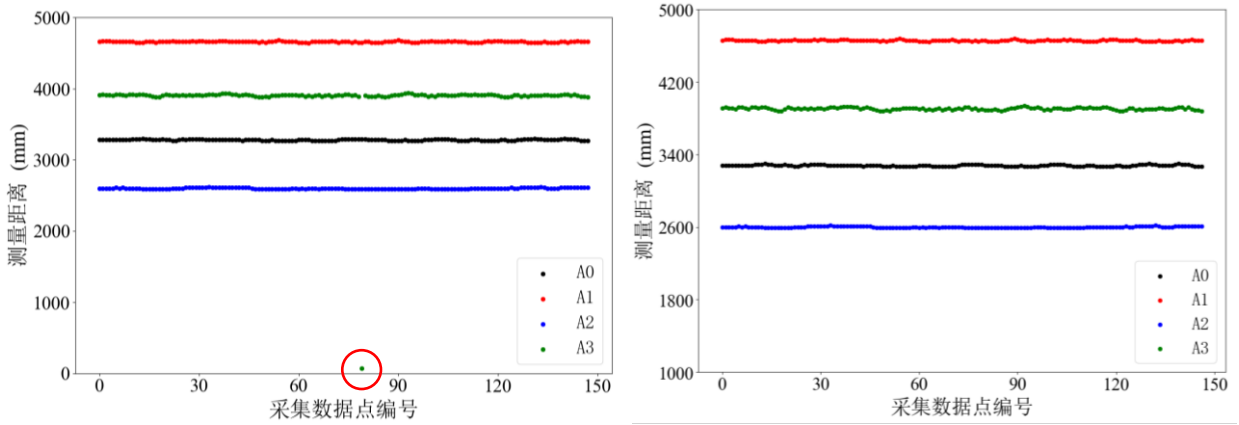
$$\begin{Bmatrix} 4910 & 5310 & 2030 & 2870 \\ 4920 & 5310 & 2020 & 2870 \\ 4920 & 5310 & 2020 & 2870 \\ .. & .. & .. & .. \\ .. & .. & .. & .. \\ 4910 & 5340 & 2040 & 2880 \\ 4910 & 5330 & 2040 & 2880 \end{Bmatrix}$$

其中，从左至右的各列数据分别表示靶点到锚点 A0、A1、A2、A3 的测距值，单位为 mm。

4.2.2 数据筛选结果

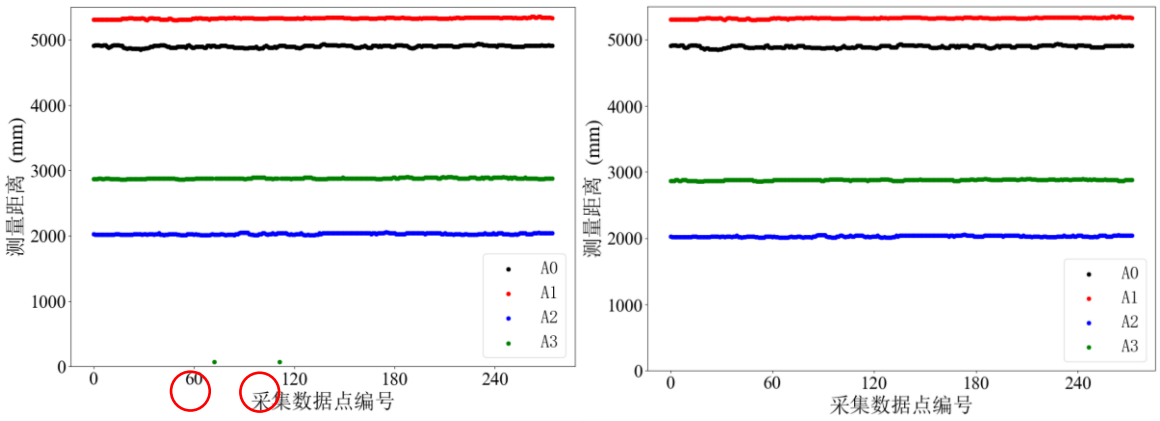
1）正常数据筛选

采用拉依达准则对“正常”数据文件中的样本数据进行筛选，为直观展示出数据筛选前后的对比，以文件 24.正常和文件 109.正常的数据筛选为例，绘制数据前后变化对比图，分别如图 4-2 和图 4-3 所示。通过两幅图的对比，发现被删除的数据均为明显偏离常值的采样数据，为清晰地表示被删除点所在位置，在图中采用红色圆圈予以标记。



（a）筛选前　　（b）筛选后

图 4-2 文件 24.正常数据筛选前后对比

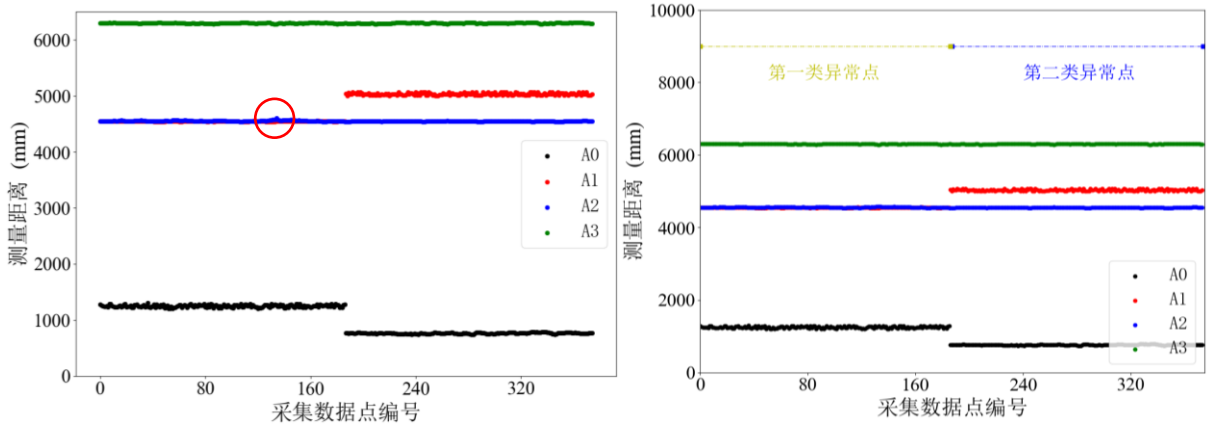


（a）筛选前　　（b）筛选后

图 4-3 文件 109.正常数据筛选前后对比

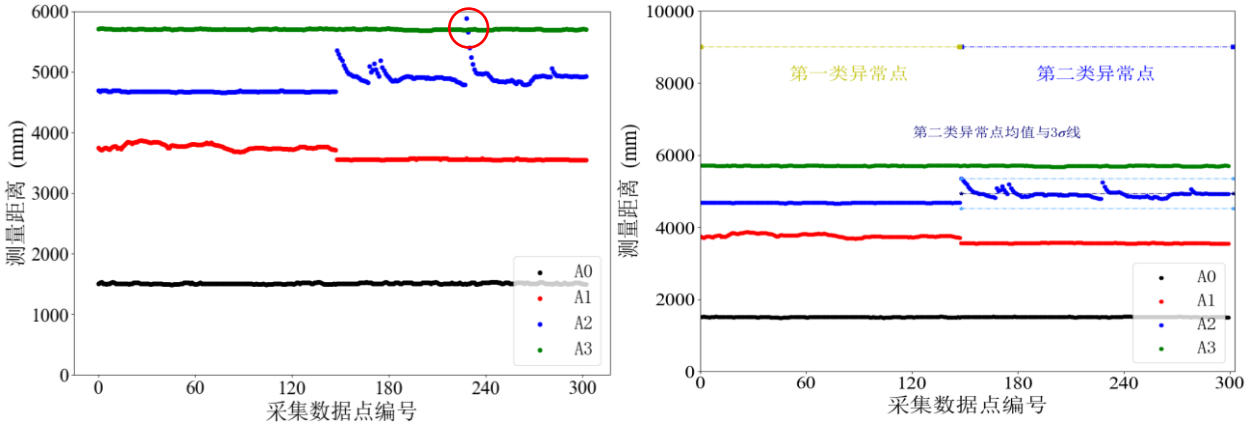
2）异常数据筛选

采用 4.1.2 小节所述对“异常”数据的清洗方法，对文件 1.异常和文件 100.异常两个数据文件进行筛选，分别展示其筛选结果如图 4-4 和图 4-5 所示：



（a）筛选前　　（b）筛选后

图 4-4 文件 1.异常数据筛选前后对比



（a）筛选前　　（b）筛选后

图 4-5 文件 100.异常数据筛选前后对比

当采集的数据是处于“异常”工况时，其测距值会相较“正常”工作时，对应点的测量值有一定程度正向的偏差，即测距值较正常值偏大。经过数据可视化处理，该偏差值展现出以下几个特征：

（1）该偏差的具体表现有两种形式，一种为如图 4-4 中 A0 和 A1 锚点的稳定断层式偏差，即从图中可以明显观察到某一段测距结果的平均值明显高于另一段；另一种为如图 4-5 中的 A2 锚点的波浪式偏差，相较第一种偏差，该类偏差没有明显的数值突跃，而是表现为在高于合理常值的一个范围内的波动，导致这两类测距偏差分别的原因可能是外界对锚点施加障碍物干扰的形式的差异——前者可能是将一屏障长时间放置于锚点和靶点的传输路径上，后者则可能是通过人的随机走动、在锚点前摆放一非静止干扰物等方法实现的干扰。

（2）但无论干扰以何种形式出现，在对多组数据的观察分析中，可以得到，在一个测距样本内，至多只有一个锚点的测距出现偏差，即同一时间只有一个锚点的信号传输受到了障碍物的阻挠，据此，可以将每组“异常”数据集中的采样点都分为如图 4-4、4-5 所示的两段，在每一段中，都分别只会有一个锚点到靶点的测距值因为障碍物的存在产生了

正向偏差。

（3）同时，观察每组数据集合中从始至终未受干扰的测距锚点（如图 4-4 中的 A2、A3 和图 4-5 中的 A0、A3）可以发现即便其它锚点受到障碍物的干扰，这些锚点测距结果并不随之产生严重的波动或变化，这说明，一种障碍物对锚点的干扰是相互独立的，障碍物的存在只会影响到其最近影响的锚点，而并不会对远端的其它三个锚点产生任何形式的影响。

根据“异常”工作点的筛选可视化结果，总结得到其误差分布特征，并在此基础上，同样根据拉伊达准则筛去其中的粗大误差点（如红圈中所示）。

3）数据筛选结果展示

采用上面数据清洗过程，针对所有抓取后的数据文件进行异常值剔除处理，得到筛选后的数据文件。重点列出题中要求展示的 4 个数据文件所保留的数据，分别为第 24 个正常工作点、第 109 个正常工作点、第 1 个信号受干扰点和第 100 个信号受干扰点的数据筛选结果，由于数据较多不便完全列出，仅挑选数据矩阵的前三行和后两行数据进行展示，具体展示如下：

$$\begin{Bmatrix} 3280 & 4660 & 2600 & 3910 \\ 3280 & 4670 & 2600 & 3920 \\ 3280 & 4670 & 2600 & 3910 \\ .. & .. & .. & .. \\ .. & .. & .. & .. \\ 3270 & 4660 & 2610 & 3890 \\ 3270 & 4660 & 2610 & 3880 \end{Bmatrix} \quad \begin{Bmatrix} 4910 & 5310 & 2030 & 2870 \\ 4920 & 5310 & 2020 & 2870 \\ 4920 & 5310 & 2020 & 2870 \\ .. & .. & .. & .. \\ .. & .. & .. & .. \\ 4910 & 5340 & 2040 & 2880 \\ 4910 & 5330 & 2040 & 2880 \end{Bmatrix}$$

24.正常　　109.正常

$$\begin{Bmatrix} 760 & 4550 & 4550 & 6300 \\ 760 & 4550 & 4550 & 6300 \\ 770 & 4550 & 4550 & 6300 \\ .. & .. & .. & .. \\ .. & .. & .. & .. \\ 770 & 5010 & 4550 & 6290 \\ 770 & 5030 & 4550 & 6290 \end{Bmatrix} \quad \begin{Bmatrix} 1510 & 3750 & 4690 & 5710 \\ 1510 & 3720 & 4680 & 5720 \\ 1510 & 3710 & 4690 & 5720 \\ .. & .. & .. & .. \\ .. & .. & .. & .. \\ 1500 & 3550 & 4920 & 5710 \\ 1500 & 3550 & 4930 & 5700 \end{Bmatrix}$$

1.异常　　100.异常

上述数据矩阵中从左至右的各列数据分别表示靶点到锚点 A0、A1、A2、A3 的测距值，单位为 mm。本文所建数据清洗模型对异常数据敏感，相较于保留下的数据数量，因粗大误差被删除的样本点数量较少，接下来展示上述 4 个数据文件中去除的样本数据，如表 4-1 所示。以第 24 个正常工作点的采样数据为例，删去了 1 组明显不正常的数据样本，而对其余的样本予以保留，对 109.正常、1.异常、100.异常三个数据文件分别删除了 2、1、3 个样本。从结果来看，上述四个数据文件分别保留了 147、273、374、300 个数据样本。此外，对于其它数据文件中数据的筛选结果可以参见上传附件。

表 4-1 样例点数据筛选情况表

样例点信息	去除数据样本编号	到 A0 距离 /mm	到 A1 距离 /mm	到 A2 距离 /mm	到 A3 距离 /mm
正常 24	80	3290	4650	2590	70
正常 109	73	4910	5320	2020	70
	112	4910	5330	2040	70
异常 1	135	1270	4550	4600	6280
异常 100	229	1500	3570	5880	5690
	230	1510	3560	5660	5700
	231	1510	3560	5400	5700

4.3 数据二次处理

把筛选好的数据进一步应用于建立定位模型前，首先还需要将初筛的数据再进一步处理，使其可作为初始数据，应用在之后确定的定位算法上。该处理其主要包含对各个测点距离进行均值滤波和对测距值系统误差校正两个部分。

4.3.1 数据关键特征提取

在问题一中，已经通过基本的筛选略去了各组数据中的异常样本，以图 4-3（b）中某一“正常”工作点的数据样本集为例，发现初筛后的数据集已经没有明显偏离的情况，而大部分采样点在平均值附近波动。据此，可以说明筛选后“正常”情况下的工作点，已剔除了其中的粗大误差，而仅剩下测量数据时的偶然误差，导致数据的上下浮动。为了在建立模型时尽可能地降低计算量，并消除偶然误差的影响，将某一工况下，确定靶点到各个锚点的距离进行均值滤波后提取，作为该靶点到每个锚点距离的最终测量结果。

再以图 4-5（b）中某一“异常”工作点的数据样本集为例，发现初筛后的“异常点”数据集除了采样点在平均值附近，同样由于偶然误差导致的波动外，还由于锚点处障碍物的阻隔，延长了锚点和靶点的数据传输时间，使某次测量内，测点距离会发生部分群体性地偏移。因此，相较于“正常”的数据集，信号传输受干扰的数据集需要额外对这部分发生偏移的数据样本进行处理、遴选，以便之后建立干扰识别模型的依据。其筛选标准即根据其和对应正常点平均值的偏离程度高低。筛选后，同样把每部分的靶点到各个锚点的距离分别作均值滤波，作为最终测量结果。由前述，每组“异常”点的采样数据集中，都分别有两个锚点在不同时刻受干扰产生误差，因此，如此处理后，每组“异常”数值将会被分为两组，每组得到了不同的测距值均值序列。

图 4-6 和图 4-7 展示了对“正常数据”与“异常数据的”均值滤波结果，图中各锚点距离呈有规律的周期性变化，与实际靶点运动轨迹相符，图 4-7 中标注了异常数据中存在干扰锚点的滤波均值，其结果普遍略高于相比于无干扰数据。

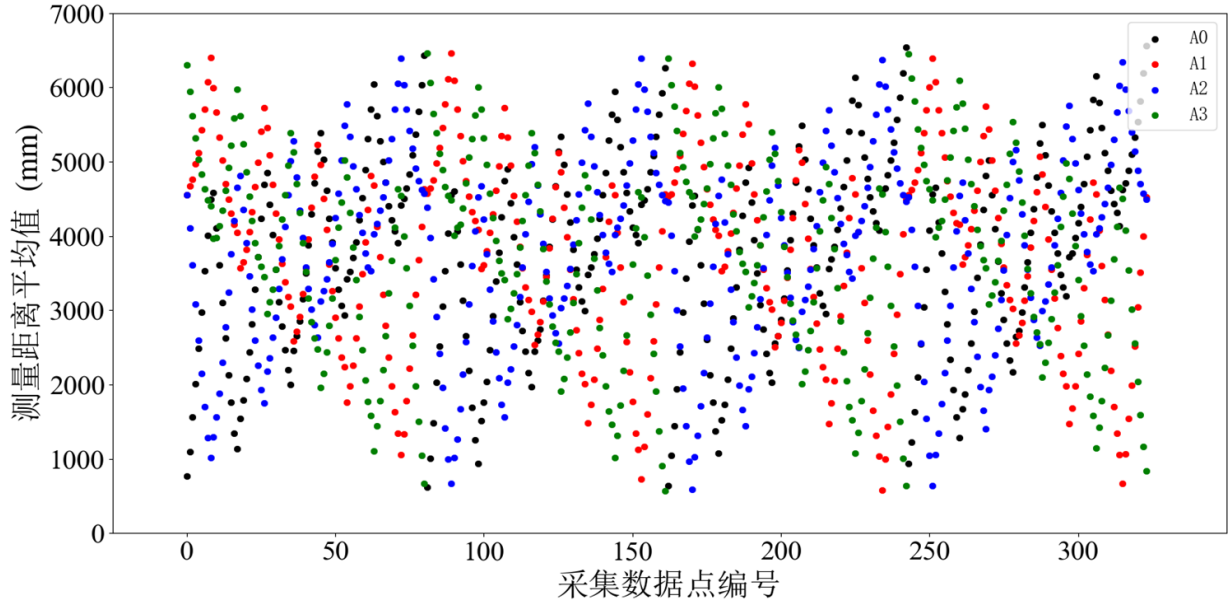


图 4-6 “正常”数据均值滤波结果

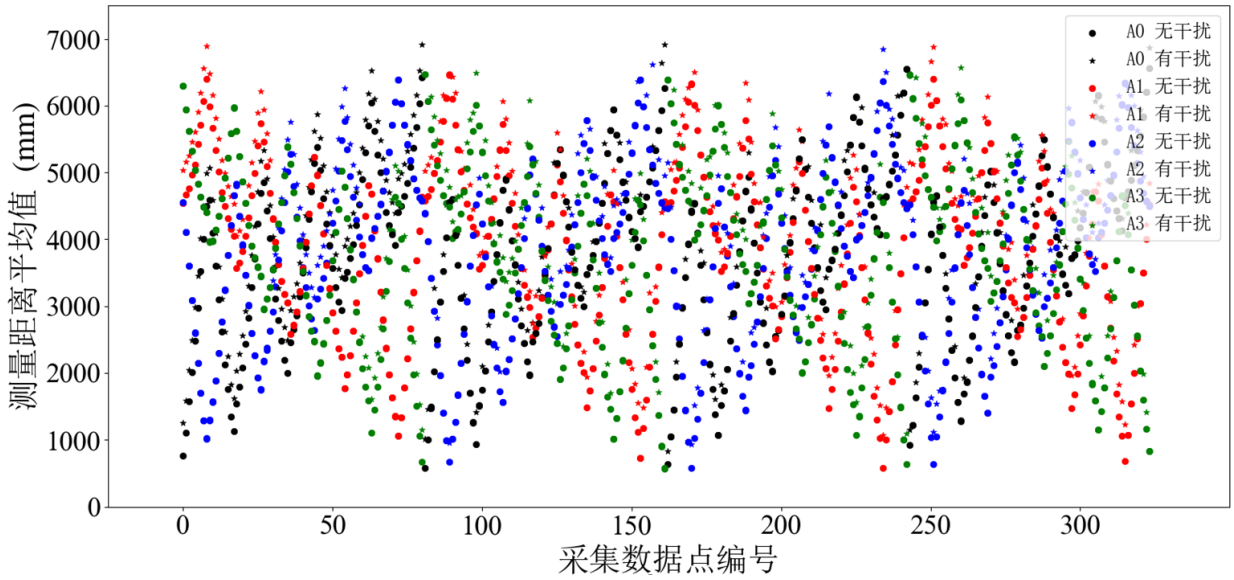
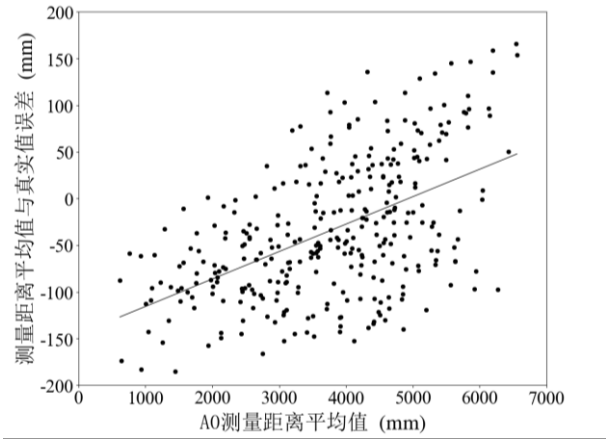


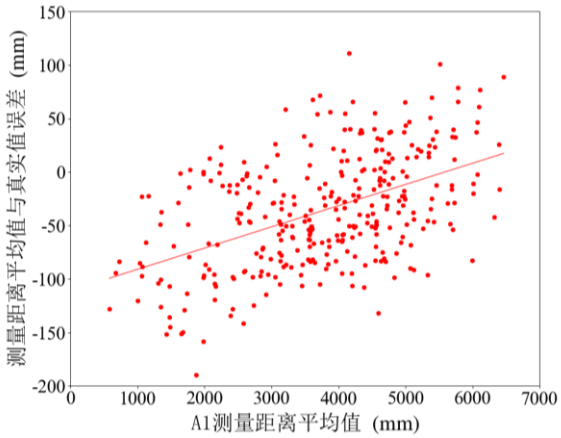
图 4-7 “异常”数据均值滤波分类结果

4.3.2 测量系统误差校正

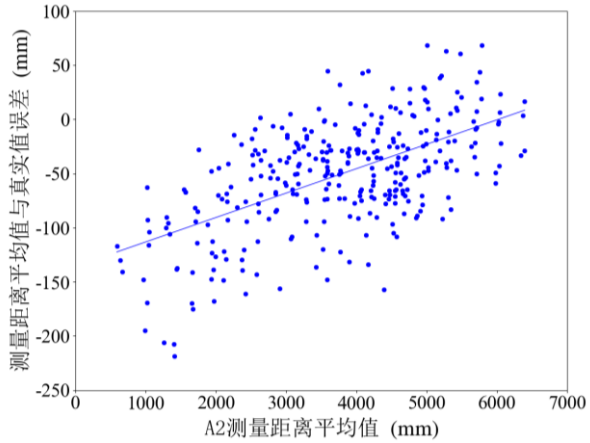
另外，注意到取平均值后，每组数据样本内，靶点到各个靶点的距离测量结果和真实值也并不相同，这说明即便各个锚点及靶点在使用前已经过良好的标零和校正，它们在发射、接收信号时仍然存在一定的系统误差，且每个锚点由于其自身工作状况不同，其系统误差种类和大小也存在差异[10]。为了消除这部分误差，从“正常数据”的 324 个文件中总结系统误差特性，以测量值为自变量，真实距离与测量值两者的差值为因变量，分别对四个锚点的测量值进行线性回归拟合，得到四条回归曲线，回归结果如下图所示：



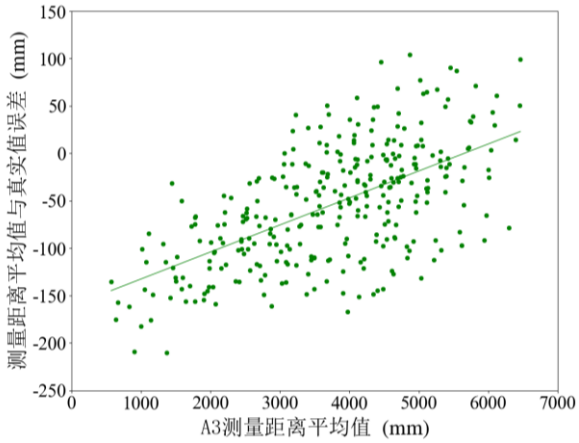
（a）A0 锚点



（b）A1 锚点



（a）A2 锚点



（b）A3 锚点

图 4-8 各锚点测距误差回归拟合结果

A0-A3 四个锚点的误差回归方程，从上往下，其拟合结果为：

$$\begin{aligned}\Delta d_0 &= 0.0294 d_0 - 0.0145 \\ \Delta d_1 &= 0.0198 d_1 - 0.0111 \\ \Delta d_2 &= 0.0226 d_2 - 0.0136 \\ \Delta d_3 &= 0.0285 d_3 - 0.0161\end{aligned} \tag{4-3}$$

用最小二乘法的线性拟合关系，得到其回归曲线，曲线两侧均匀分布各个测点数值，说明拟合效果良好，各点系统误差得到有效消除。另外，每条回归方程其初始截距恒负，说明由于系统误差导致的传输迟延效应，测得的观测距离总是会比实际值偏大的。

至此，完成了第一问的数据抓取和筛选内容，并通过均值滤波、回归拟合的方法，对筛选的数据集进行了深化处理，为后文精确定位模型以及干扰识别模型的建立奠定基础。

五、问题二模型的建立与求解

为实现对不同类型数据样本的空间定位，本文先是提出了遗传算法和遗传—三边定位方法耦合的两种定位算法，采用这两种定位算法，实现了对锚点工作不受干扰和锚点工作受到障碍物干扰两种工况下，不同数据样本的精确定位；基于神经网络算法，建立定位模型的误差修正模型；之后，引入平均绝对误差和均方根误差两个评判标准，对模型的精确

度进行评价，评价结果表明定位结果和真值偏差较小，定位结果十分精准。最后，利用建立的定位模型，代入该实验场景下的另一组样例数据点，对其进行准确空间定位。

5.1 基于改进遗传算法的定位预测模型

目前，在利用传感器网络进行室内定位时，常见的定位方法有质心定位算法、DV-Hop算法[11]、凸规划算法[12]等，以上这些算法之前多应用于二维平面的物体位置确定问题中，若应用在三维复杂环境下的定位问题，其计算函数复杂度大大提高、对求解空间提出了很高的要求，结果准确性也受影响。

而遗传算法作为一种基于模拟生物进化过程的算法，以决策变量的编码作为运算处理的对象，求解寻优过程中并不关联复杂目标函数的求微求导，仅使用适应度函数值来度量个体的优良程度，对除适应度函数之外，其它辅助信息的要求很低。其全局搜索性、灵活性及与多种算法配合的可扩展性，拓宽了其应用范围。鉴于题目中给定信息的有限程度和不大的数据量，非常符合遗传算法的适用条件[13]。因此，本文采用遗传算法作为定位模型的基本算法。

如上述，对于采集的数据集中，未受信号干扰的部分，运用遗传算法可较为准确地对其空间坐标进行定位，而由于该阶段的定位模型尚未设计对“异常”点的识别，而如上一章所述，经过预处理后的数据集表明，锚点工作受到外界障碍物干扰时，同一时间仅有一个锚点被干扰，且干扰信号在测距结果上具体表现为测距结果的不正常偏移。可以预见到，该偏移会导致在对“异常”数据点进行位置求解时，用于遗传算法的每组四个的测距数据其可靠性下降，若继续沿用对“正常”数据点的位置求解方法，其精度必然难以达到要求。因此，需要对原有的遗传算法模型作适当地修正和改进，使其应用于“异常”数据集合定位时也具备相当的精度。

考虑前述的传统空间定位方法，其主要缺陷为较为复杂的算法在对大量数据求解时，其精度和时间成本的不理想。而根据遗传算法求解得到的“异常”数据定位结果，其精度虽然不足，但已经完成了相当部分的位置计算，在其基础上，若引入上述的各类传统空间定位方法对其结果进行精确化细致化迭代，最后的求解结果可以得到有效改善。在这个思路的指引下，本文提出一种遗传算法和三边定位方法耦合的改进算法，由此算法实现对已知“异常”工况点的空间定位。

进一步地，利用对“正常”工作点和“异常”工作点分别创建的定位模型，可以得到各个靶点的预估位置，该预估位置势必和靶点的真实位置存在一定偏差，将该偏差深入地进行修正，能够得到更优的位置预测结果；而考虑到神经网络算法在建立庞大数据间的回归关系时，具备无可比拟的优越性，本文将预测位置与真实值比较，得到两者误差分布情况，再基于神经网络方法，构建预测误差和预测位置之间的回归关系，后期建立位置误差修正模型。实现对其误差进行进一步修正，得到最优定位结果。

最后，需要依据题目要求对模型的预测结果进行准确性和有效性评判，目前，在空间定位问题中，均方根误差（RMSE）以及平均绝对误差（MAE）是两种最为常见的判别定位精确度的标准，所以，本文根据预估结果在空间不同维度的均方根误差及平均绝对误差大小，判别模型的精准及有效程度。

综上，本问的思路框图如下流程图所示：

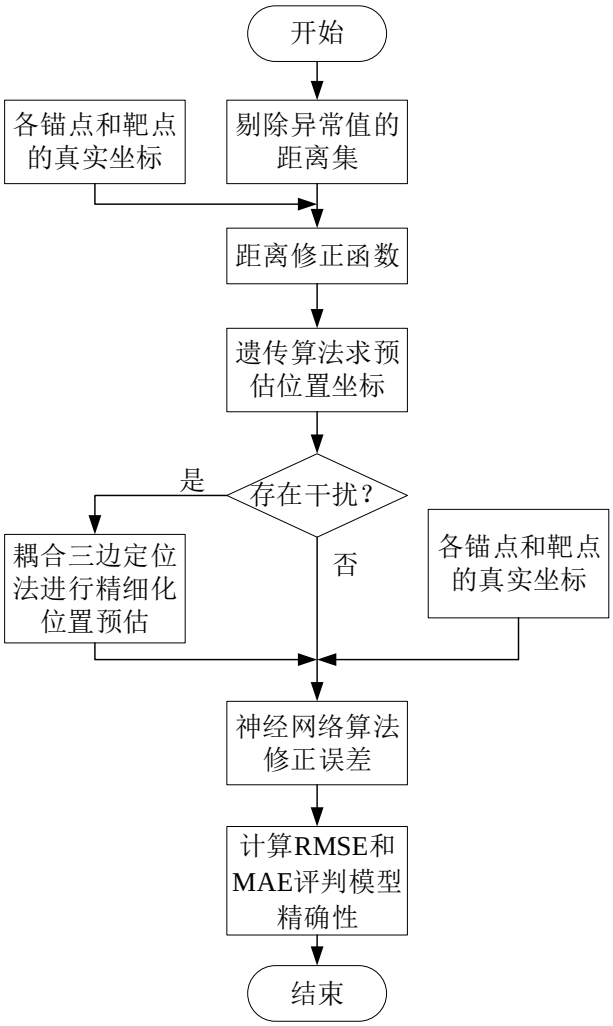


图 5-1 第二问求解思路流程图

5.1.1 遗传算法原理

上章已完成对建模数据的二次处理，该节主要利用处理完成的数据和题目给出的信息，结合遗传算法，实现在锚点未受干扰情况下采集到的“正常”数据的定位模型建立[14]。

遗传算法是一种用计算机方法模拟自然界的动物物种由低级种群到高级种群的进化过程的仿真方法。 是从某一初始群体（在计算机中即为原始数据）开始，采用适者生存，不适者淘汰的自然法则选择数据个体，通过引入交叉、变异算子以不断生成后代数据群体，逐代升级进化，直到其产生满足适应度条件的数据个体为止。综上，遗传算法的演化过程和算法可用以下各符号描述：

$$\mathrm{GA}=(P(0),N,L,s,g,p,f,t) \tag{5-1}$$

其中， $P(0)$ 表示开始时的数据群体， $P(0)=(p_1(0),p_2(0),p_3(0),\cdots\cdots p_n(0))\in I^N$ ； I 表示长度为 L 的二进制串空间，将数据转化为 0-1 二进制是为了模拟生物种群中的基因在染色体上的编码情况， $I=B^N=\{0,1\}^L$ ； N 表示初始种群中的数据个数； s 表示遗传算法进行过程中的种群选择（淘汰）策略， $s:I^N\longrightarrow I^N$ ； g 表示遗传算子，常见情况下，有复制算子、交叉算子以及变异算子三种算子，即：

$$g:\begin{cases} Q_r(\text{复制算子}):I\longrightarrow I \\ Q_c(\text{交叉算子}):I\times I\longrightarrow I\times I \\ Q_m(\text{变异算子}):I\longrightarrow I \end{cases} \tag{5-2}$$

P 表示遗传算子的不同操作概率，常见情况下有繁殖概率、交叉概率以及变异概率三种，即：

$$P:\begin{cases} P_r(\text{繁殖概率}) \\ P_c(\text{交叉概率}) \\ P_m(\text{变异概率}) \end{cases} \tag{5-3}$$

f 表示种群的适应度函数（寻优目标函数），$f:I\longrightarrow R^+$；

t 为种群的进化终止准则，$t:I^N\longrightarrow\{0,1\}$。

以上述的适应度函数为依据，遗传算法对初始数据群体中的每个个体进行遗传操作，对数据群体中的每个数据进行类似于自然选择对生物个体的逐世代迭代处理问题，其基本的算法框图如流程图 5-2 所示[15]：

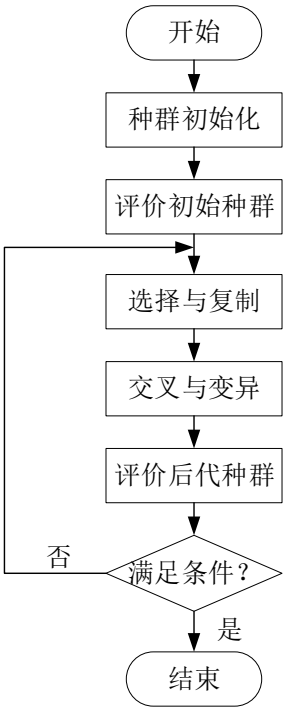


图 5-2 遗传算法标准流程图

5.1.2 基于“正常”数据的建模

对于本问，鉴于题目中已经给定了各个工作靶点的真实坐标和距离测量结果，可依此设定适应度函数（寻优目标函数），选用各个“正常”工况点经过均值滤波和拟合函数校正的四组测距值作为数据种群初始值，筛选种群进化（迭代过的数据）中最满足算法设定寻优目标的个体，作为种群进化的终止条件。

对于初始数据种群 $P(0)$ 中的任意第 i 个元素，可以在题设中找到其真实坐标 (X_i,Y_i,Z_i),，种群进化后，其位置预估值会发生变化，设其预估坐标 $(\tilde{x}_i,\tilde{y}_i,\tilde{z}_i)$；该元素到第 j 个锚点的标定距离 d_{ij} 已在前节中求得，设第 j 个锚点的真实坐标为 (x_j,y_j,z_j)，可度量其预估坐标到任意靶点的距离和实际距离的差值如下：

$$L_{ij}=\left|\sqrt{(\tilde{x}_i-x_j)^2+(\tilde{y}_i-y_j)^2+(\tilde{z}_i-z_j)^2}-d_{ij}\right| \tag{5-4}$$

可以预见，在理想的预测情况下，使得上述值为 0 的靶点预估坐标值$(\tilde{x}_i,\tilde{y}_i,\tilde{z}_i)$是定位坐标的最适解，但实际预测时，只能期望该值尽可能小而非总等于 0，所以，设定寻优目标函数为：

$$\begin{aligned} h(X)&=\sum_{j=1}^{4}L_j \\ f&=\min[h(X)] \end{aligned} \tag{5-5}$$

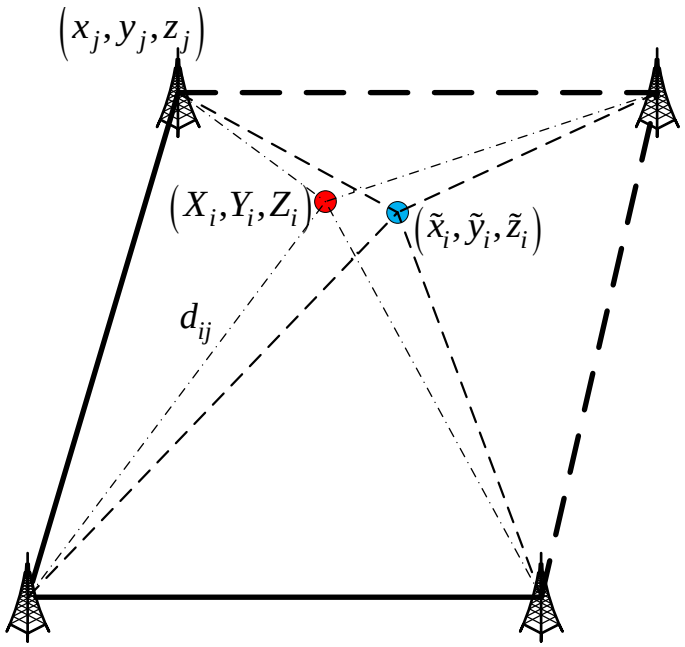


图 5-3 各坐标及物理量实际意义示意图

对于种群的基本约束条件即为种群的个体在题目要求的靶点运动范围内。

对于初始种群，使用二进制编码不能直接反映群体结构，占据一定的内存空间，每个个体代表空间的一个位置。

由于是求取函数最小值，因此，在实际算法运行中，采用选择概率为 0.001.

由于定位模型的原始数据量较大，本文采用赌轮法作为选择算子，交叉算子采用多点交叉法，三维坐标分别对应交叉；对变异算子进行合理的调节，使得群体中选择概率低的个体，其变异能力较强。

当群体迭代到规定代数或者达到规定收敛程度时， 终止遗传算法迭代过程，得到最终优化定位结果。

5.1.3 基于“异常”数据的建模

在基于遗传算法对其位置进行初步预估后，由于“异常”数据中，总有靶点到某锚点的距离值和实际值产生偏离，因此在遗传算法的基础上，引入三边定位算法，耦合于遗传算法对这类数据点的位置进行正确估计。

三边定位方法是一种依据多传感器网络的常见空间定位手段，可根据待求节点到各个传感器节点之间的位置关系，来分辨其详细位置，应用在三维空间时，其需要四个传感器节点到待测节点之间的距离作为已知量，列写一多三元高次方程组，求解未知节点的具体位置，符合本题的背景。因此，本文将三边定位方法作为一个工具，将其和前述遗传算法

进行耦合，实现对“异常”数据位置的精准判别。由于篇幅所限，三边定位方法的基本计算原理可参加第七章，本节主要介绍利用该耦合模型的建模思路。

鉴于带干扰的“异常”点，其定位误差是由某时刻内，靶点到某一靶点的测距值偏大导致的，若能够对该偏大的测距值进行合理调整，便可以实现对信号传输受障碍物干扰的样点位置校正。基于此，根据干扰数据的遗传算法计算结果，提出一种迭代式的三边定位算法：

（1）已知一个数据样本内，靶点到四个对应锚点的距离测量值；

（2）依次利用遗传算法寻优减小四个测量值的大小，使用三边定位法计算修正后的定位坐标；

（3）重新计算修正后的定位坐标到四个锚点的对应距离，判别其与原始距离的偏差，若偏差小于迭代要求值，则输出修正定位坐标，否则返回上一步继续修正测距值和定位坐标；

（4）得到最终的修正坐标结果。

该迭代三边定位算法流程图如下图所示：

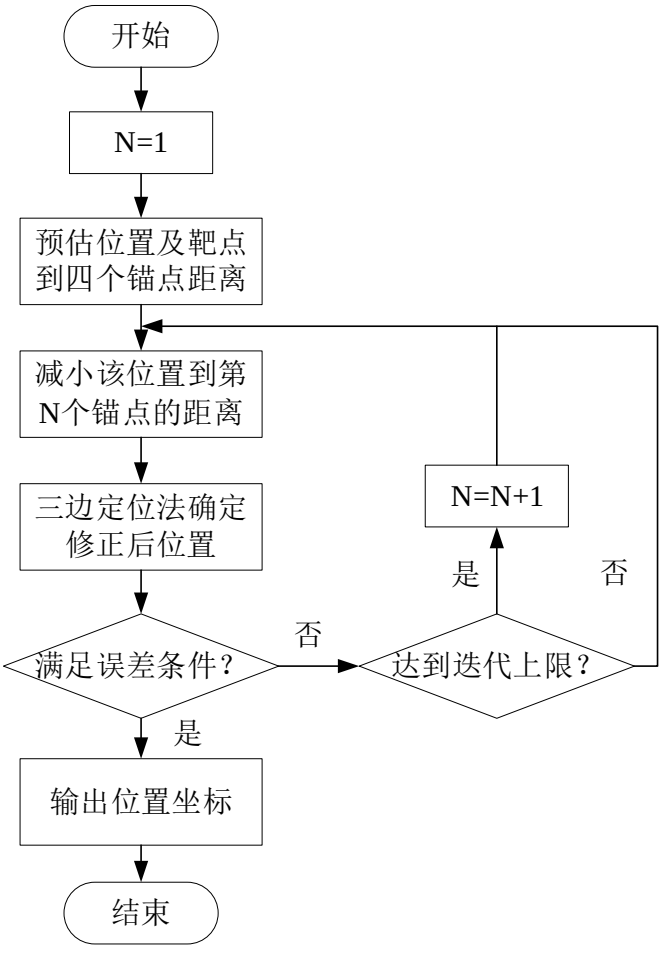


图 5-4 遗传-三边定位耦合算法流程图

本题中，由数据的处理结果发现，所有“异常”数据点，其最大偏差值不会超过 600 毫米，因此取图 5-4 中每个距离测量值的迭代上限为 600 毫米，即约束遗传算法寻优函数的距离减小值满足：

$$\Delta d_{ij}=\left|\hat{d}_{ij}-\tilde{d}_{ij}\right|\le 600\text{mm} \tag{5-6}$$

其中，$\hat{d}_{ij}$为迭代前，第 i 个靶点位置到第 j 个锚点的标定距离，$\tilde{d}_{ij}$则表示迭代后第 i 个靶点位置到第 j 个锚点的标定距离。

而迭代需满足的误差条件即：

$$\begin{aligned}\tilde{L}_{ij} &= \left|\sqrt{(\tilde{x}_i - x_j)^2 + (\tilde{y}_i - y_j)^2 + (\tilde{z}_i - z_j)^2} - \hat{d}_{ij}\right| \\ \tilde{h}(X) &= \sum_{j=1}^{4} \tilde{L}_{ij} \\ \tilde{f} &= \min[\tilde{h}(X)]\end{aligned} \tag{5-7}$$

其中，$(\hat{x}_i, \hat{y}_i, \hat{z}_i)$为迭代前的坐标预估值，$(\tilde{x}_i, \tilde{y}_i, \tilde{z}_i)$为迭代后的坐标预估值。

至此，完成对“正常”以及“异常”两类工况点数据的定位模型建立。

5.2 模型修正及有效性验证

这一节中，引入神经网络算法，作为模型的误差修正，并以平均绝对误差（MAE）和均方根误差(RMSE)作为模型判别标准，对模型的定位精度和准度作出评价。

5.2.1 基于 BP 神经网络误差预测的模型修正

BP 网络是一种含有隐含层的多层前馈网络，采用全局逼近方法的学习算法，主要包括初始化、计算实际输出和权值调整三部分，能够逼近任意非线性映射关系，模型泛化能力良好[16]。本文使用 BP 神经网络模型的函数逼近功能，即用输入向量和相应的输出向量训练网络以逼近一个函数，由此实现对模型的误差进行预测。值得一提的是，本模型中训练的输入向量为坐标数据值，输出向量为误差的预测值。基于所得误差预测结果，对 5.1 定位预测模型所得定位结果进行修正，本方法相当于是对预测结果和真实值之间进行了一次标定，能够显著提高模型的精度。

BP 神经网络是由输入层、中间层与输出层所组成的阶层型神经网络，中间层可扩展为多层，神经网络的结构如图 5-1 所示。该模型的原理是相邻层之间各神经元进行全连接，而每层所包含的各神经元之间无连接，网络按有教师示教的方式进行学习，当一对学习模式提供给网络后，各神经元获得网络的输入响应产生连接权值（Weight）。然后按减小希望输出与实际输出误差的方向，从输出层经各中间层逐层修正各连接权，回到输入层。此过程反复交替进行，直至网络的全局误差趋向给定的极小值，最终完成学习的过程[17]。

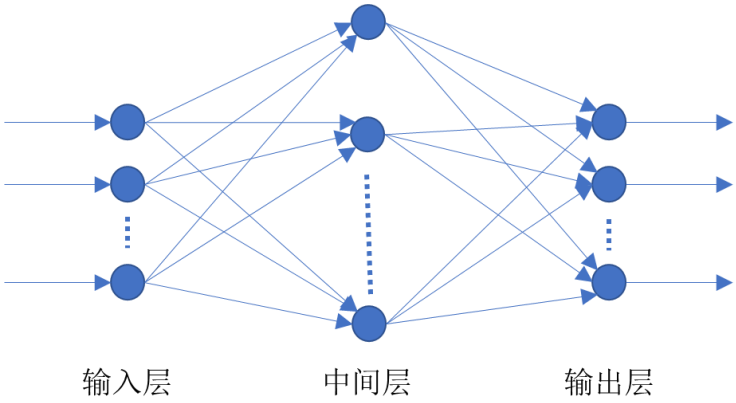


图 5-5 神经网络模型

1）数据归一化处理

应用 BP 神经网络进行预测，需要对输入层数据做标准化处理，归一于[-1, 1]，与此同时输出层的预测值输出范围也为[-1, 1]，在后续对所建模型进行误差分析时，需将所得输

出数据进行反归一化处理再进行比较。本模型对分别定位点 x、y、z 坐标以及定位模型预测结果误差进行归一化处理，采用 Matlab 软件中的 mapminmax 函数实现。mapminmax 原理表述成数学公式为：

$$y = (\mathrm{y}_{\max} - y_{\min}) \times \frac{(x - x_{\min})}{x_{\max} - x_{\min}} + y_{\min} \tag{5-8}$$

公式（5-8）中，如果 $x_{\max} = x_{\min}$，会出现除数为 0 的情况，则此时数据不变。

2）BP 算法构建

为建立定位模型预测结果的误差与靶点位置坐标的关系，输入层神经元个数设置为 3，分别为靶点位置的 x、y、z 三个方向坐标值；输出层神经元个数为 1，设为定位模型预测结果的误差值；隐含层节点个数设置为 8 个，由模型的收敛速度与计算精度试验得出。

输入层有 3 个输入节点 x_1, x_2, x_3，输出层有 1 个节点 y，隐含层含有 q 个神经元（$q = 8$），隐含层的第 i 个神经元（$i = 1, 2, 3$）在样本 p 的作用下输入为：

$$net_i^p = \sum_{j=1}^{6} w_{ij} x_j^p - \theta_i \tag{5-9}$$

其中，x_j^p 为输入节点 j 在样本 p 作用下的输入；w_{ij} 为输入层第 j 个节点与隐含层第 i 个节点之间的权值；θ_i 为隐含层神经元的阈值。

输出层第 k 个神经元（本模型中 $k = 1$）的输入为：

$$net_k^p = \sum_{j=1}^{q} w_{ki} o_i^p - \theta_k, k = 1 \tag{5-10}$$

其中，θ_k 为隐输出层神经元的阈值；w_{ki} 为隐含层第 i 个节点与输出层第 k 个节点的权值；o_i^p 为输入节点 j 在样本 p 作用下的输出，对输入节点来说二者相当。系统网格对 3 个训练样本的总误差函数为：

$$J = \frac{1}{2} \sum_{p=1}^{3} \left(t_k^p - o_k^p \right)^2, k = 1 \tag{5-11}$$

其中，t_k^p 为在样本作用下输出节点的目标值，权值修正采用有动量的梯度下降法，使网络逐渐收敛。

权重系数增量公式为：

$$\begin{aligned} w_{ki}(k+1) &= w_{ki}(k) + \eta \sum_{p=1}^{3} \beta_k^p o_i^p \\ \beta_k^p &= o_k^p \left(1 - o_k^p\right)\left(t_k^p - o_k^p\right) \\ w_{ij}(k+1) &= w_{ij}(k) + \eta \sum_{p=1}^{3} \beta_j^p o_j^p \\ \beta_j^p &= o_j^p \left(1 - o_j^p\right) \beta_j^p \cdot w_{ki}, k = 1 \end{aligned} \tag{5-12}$$

其中，η 为学习速率，本文设为 0.01。

综上所述，本文所构建算法由三部分组成，首先对网格初始化，将所有加权系数设置

为最小随机数；然后根据输入层节点 x_1, x_2, x_3 计算隐含层神经元及输出层神经元的输出，并根据输出值计算与实际值的误差；最后对权值 w_{ki} , w_{ij} 进行修正，直至误差小于给定值或者超出迭代次数为止。

5.2.2 模型有效性验证

首先选定误差评定方法，据此对训练好的模型进行验证。本文分别计算出所建模型的 3 维、2 维及 1 维精度，其中，3 维精度针对（x, y, z）进行误差分析，2 维精度对（x, y）进行误差分析，1 维精度对 x、y、z 三个方向分别进行误差分析。

评定所建定位模型预测误差的大小，可采用平均绝对误差 MAE 以及均方根误差 RMSE 两个指标进行精度分析。二者计算公式分别如下：

$$MAE = \frac{\sum_{i=1}^{n} \left\| P'_{\text{tag},i} - P_{\text{tag},i} \right\|}{n} \tag{5-13}$$

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{\left\| P'_{\text{tag},i} - P_{\text{tag},i} \right\|^2}{n}} \tag{5-14}$$

其中，$P'_{\text{tag},i}$ 表示定位标签 i 坐标的计算值，$P_{\text{tag},i}$ 表示定位标签 i 坐标的真实值。

依此，结合之前建立的数据定位模型、误差修正模型对题设中给定的数据进行定位和误差分析。

1）对“正常”数据集的验证

现将题目中采集到的“正常”数据集合，代入本章建立的遗传算法定位模型，获得 324 个正常数据点的定位坐标，其三维空间分布与各坐标结果如图 5-5 所示。

可见，原有的各个靶点，其位置分布在四个高度层上，与靶点轨迹图中四个平面对应，每个高度上以近似 S 型的路径移动，在坐标图中反映出 x 与 y 坐标的周期性变化。与图 5-6 对比可知，利用已有模型对“正常”数据进行定位时，x、y 轴方向的位置求解较为准确，z 轴方向存在一定偏差，但整体趋势与真实轨迹相一致。

图 5-6 中，在 x、y 轴上，所有的定位点都分布得较为集中，其定位结果的方差较小，定位效果良好；而在 z 轴的高度方向上，每一层的定位点相较其它两个方向更分散，说明这个维度上的定位并不如另两个维度。而观察图 5-6，可以注意到靠近锚点 A3（5000,5000，1300）一侧的靶点，其预测位置更倾向于向这一锚点集中，而靠近 A3 对角位置的锚点 A0（0,0,1300）,一侧的靶点，其预测位置更为发散，这说明定位模型的误差分布是根据其位置坐标有一定分布关系的，这映证了前述的神经网络方法修正定位误差的可行性。基于此

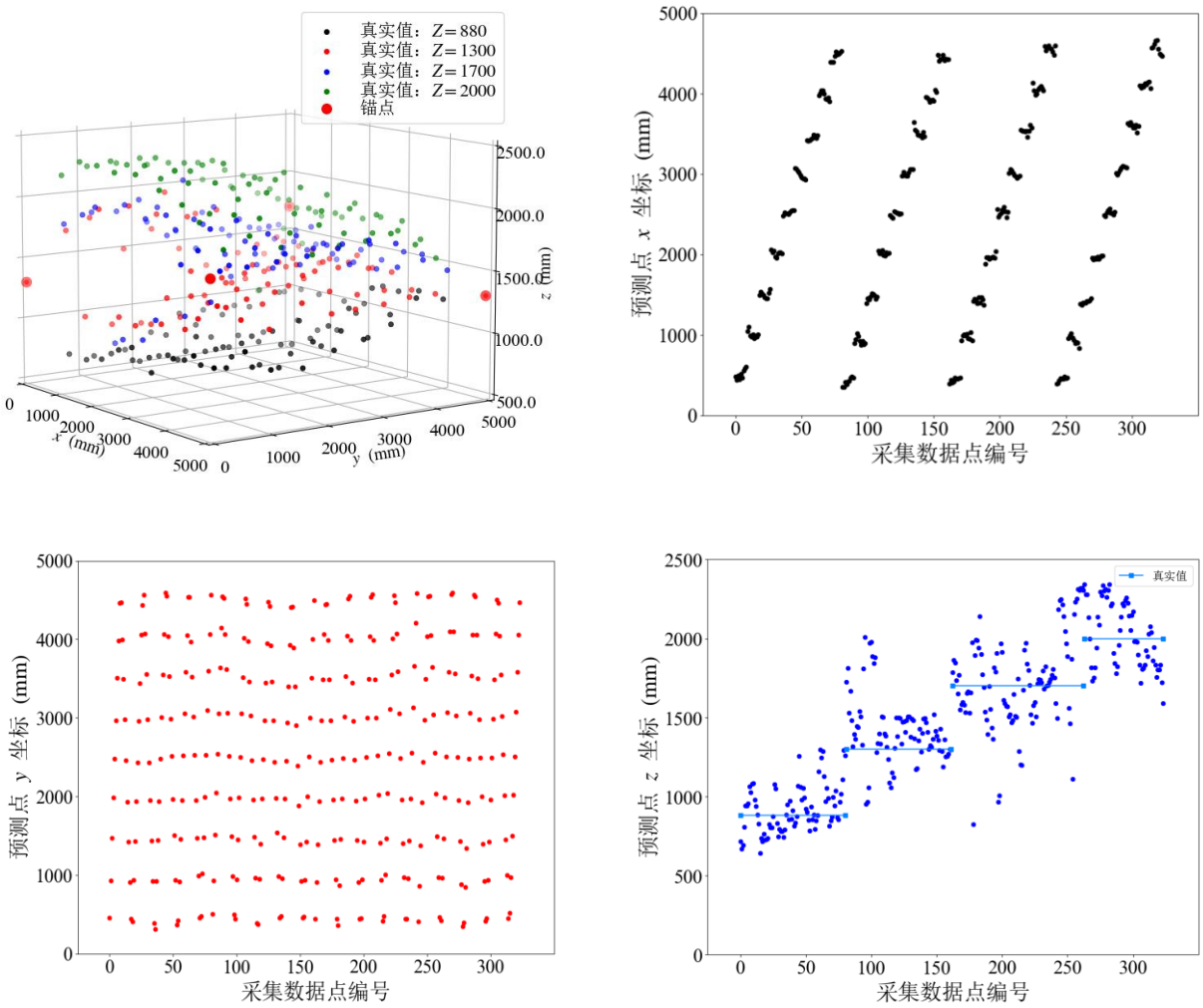


图 5-6 “正常”数据点分布（处理前）与各个坐标轴定位

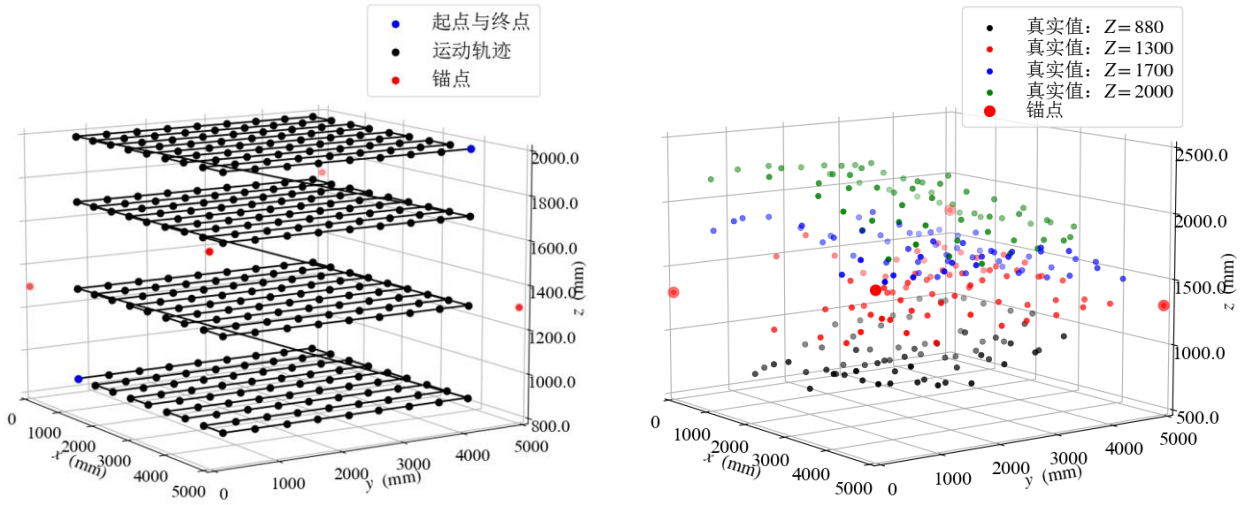


图 5-7 真实结果和“正常”数据集（处理后）的定位结果比较

引入误差修正模型。在修正前，首先对预测结果进行预处理，利用拉伊达准则，剔除误差预测中的严重偏差，处理结果如上图所示。

将处理后的数据代入神经网络算法，取其中 80%的数据为训练集，10%的数据为验证

集、10%数据为测试集，最后得到训练数据、测试数据和验证数据、总体数据的回归结果如下图所示：

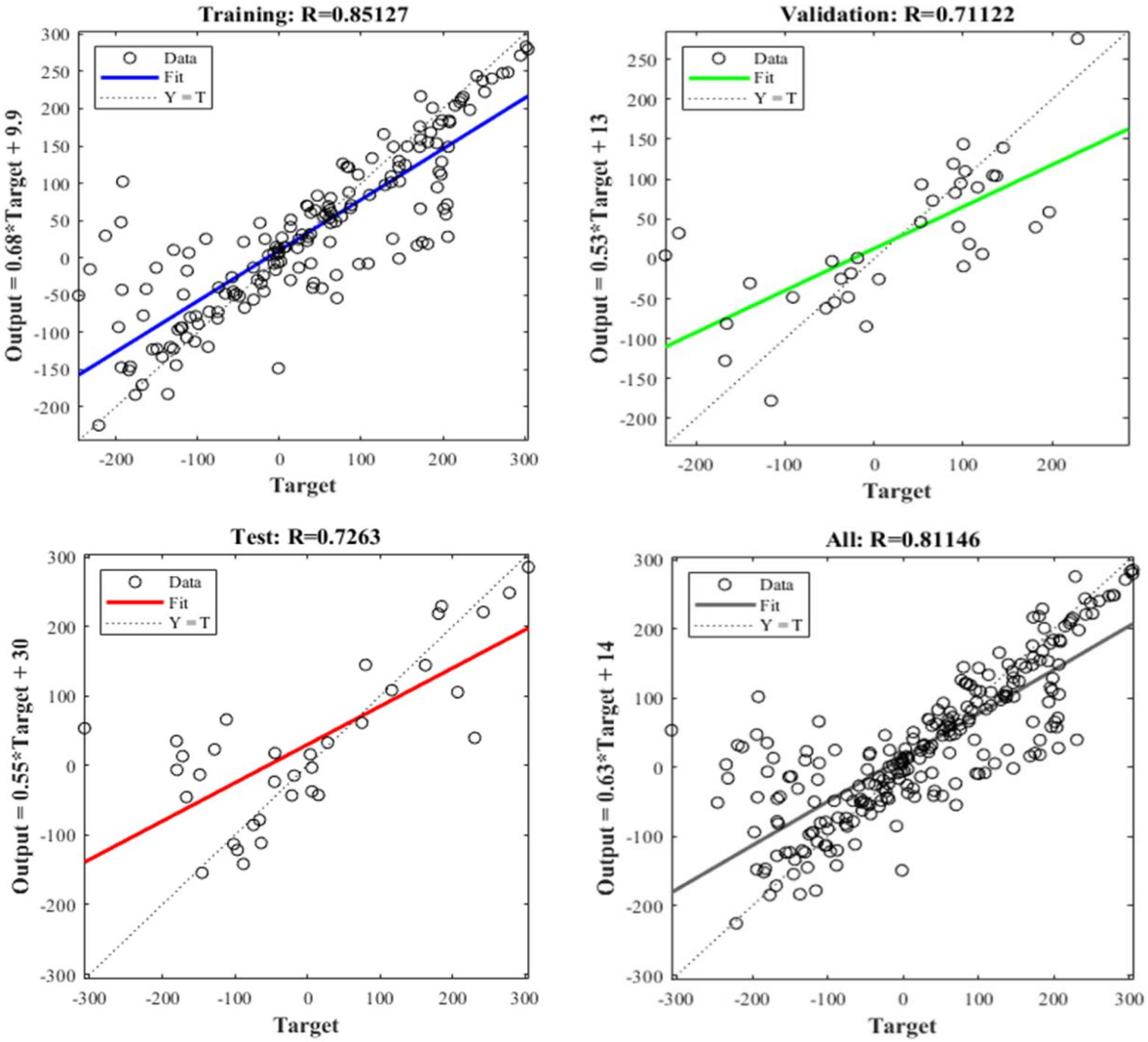


图 5-8 各个数据集合的训练结果

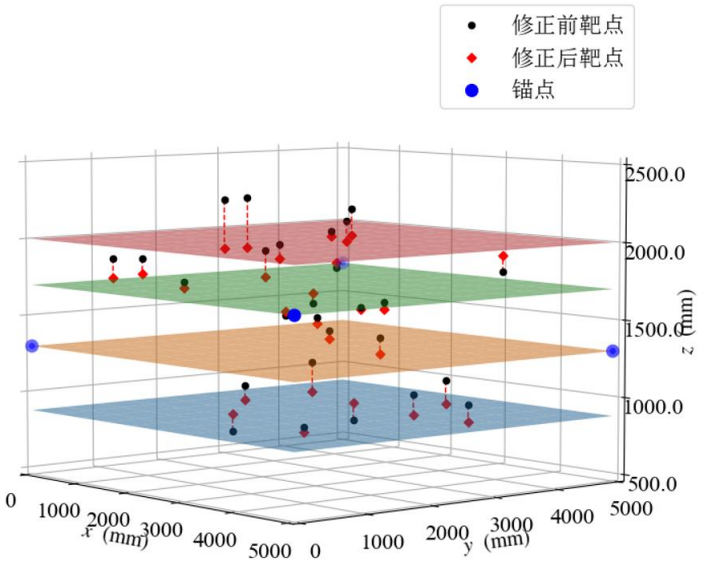


图 5-9 神经网络测试集修正结果

图 5-8 中，各个集合的回归系数都控制在 0.7 以上，回归拟合效果良好，证明模型对于 z 坐标的误差修正效果较为优秀。验证模型对某个测试集的修正效果，将修正后的测试集的坐标点表示在三维空间上，结果如图 5-9 所示。

图中用四个分层平面表征前述的四个“正常”数据集合的分布高度，由图可见，经过误差模型修正后，所有测试集的点都落回其正确的高度上，说明该误差修正模型的运用效果很好，让所有的点的 z 轴方向偏差都得到了合理的补偿。

2）对“异常”数据集的验证

同样地，为了更直观地展示各个位置的位置偏差，作出各个坐标轴上的偏差如图 5-10 所示：

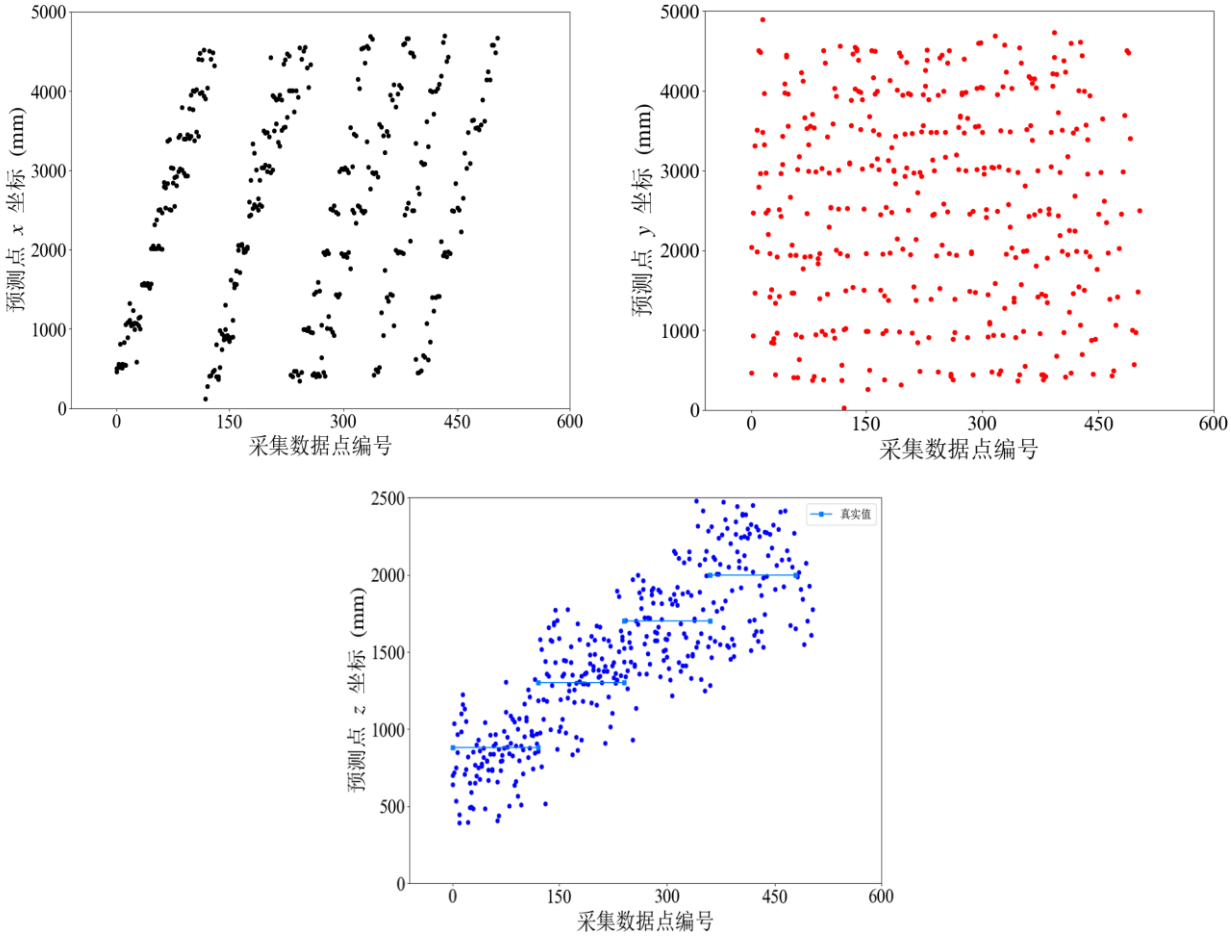


图 5-10 “异常”数据点各个坐标轴定位

和“正常”数据集相似，各个数据样点在 x、y 轴上定位效果要远优于在 z 轴方向上。注意的是，对于采集的 648 个数据样本，只有其中 508 的数据，在 z 轴方向的定位落在较为合理范围内，即对于本文所建立的遗传—三边定位算法，在高度方向上拥有 80%左右的可靠性。

对于该现象，作出如下解释：

（1）传统的三边定位方法，应用在二维定位时，需要保证各个定位圆必须相交，且某一边的测距偏差会对定位结果产生相当的影响，而在三维位置定位时，这种影响显然是会被靶点到锚点的某个距离偏差更大程度地倍化了；

（2）由于这个测试场景，四个锚点分布在一个扁长型的长方体上，该长方体的高度

维度较长度和宽度两个维度要小很多，这意味着同样程度定位圆半径的偏差，在高度方向引起的误差会比另外两个维度大得多，即该维度对测距偏差更为敏感，也由此，可以作出判断：若是把该模型应用于场景二，由于其锚点分布的长方体相较于场景一更为细长，那么对其上测点数据作出的空间位置估计，相较于该场景一，势必更加精确。

由此，完成对定位模型可行性的验证，接下来分析其定位精度。

如前述，使用平均绝对误差（*MAE*）和均方根误差（*RMSE*）分析其标定精度，列出“异常”和“正常”数据集的误差计算结果如下表所示：

表 5-1 数据集误差计算结果

样本类别	误差类别	x/mm	y/mm	z/mm	（x,y）/mm	(x,y,z)/mm
“正常”数据	*MAE*	51.8221	50.5716	66.4267	80.6227	119.2698
	RSME	63.2700	62.1823	100.9334	88.7115	138.4702
“异常”数据	*MAE*	101.8070	93.9945	195.9572	146.6718	269.1134
	RSME	149.3874	143.0337	238.3448	206.8218	315.5685

如上表，相较于受干扰的测点，未受干扰的测点，其误差基本可以控制在厘米级，满足对精确定位的数量级要求。而“异常”工作点，由于测距误差的客观存在，其误差较“正常”点较大，但也都可以控制在合理范围内。以上误差验证结果，证明本模型具备相当的精确性。

5.3 模型精准定位结果

在前一节的基础上，该节对题目另外给出的测点，作出位置预测，其定位结果如表 5-2：

表 5-2 实验场景 1 下 10 组数据定位结果

样例点序号	*x*/mm	*y*/mm	z/mm
1	1139.82	608.49	754.99
2	3175.25	1686.72	814.05
3	2737.95	1128.53	858.83
4	2453.46	962.20	2255.99
5	1451.49	2538.42	1895.17
6	2029.63	718.43	512.82
7	4113.92	1789.25	1971.96
8	1623.85	1099.21	1152.48
9	3540.53	1965.29	2316.68
10	4698.75	2171.56	1434.10

得到的前五个在“正常”工况下测得的数据点空间坐标分别为(1139.82,6008.49,754.99)、(3175.25,1686.72,814.05) 、 (2737.95,1128.53,858.83) 、 (2453.46,962.2,2255.99) 、(1451.49,2538.42,1895.17)；后五个在“异常”情况下测得的数据点，其预估空间坐标分别为 (2029.63,718.43,512.82) 、 (4113.92,1789.25,1971.96) 、 (1623.85,1099.21,1152.48) 、(3540.53,1965.29,2316.68)、(4698.75,2171.56,1434.10).

并将其展示在三维空间上：

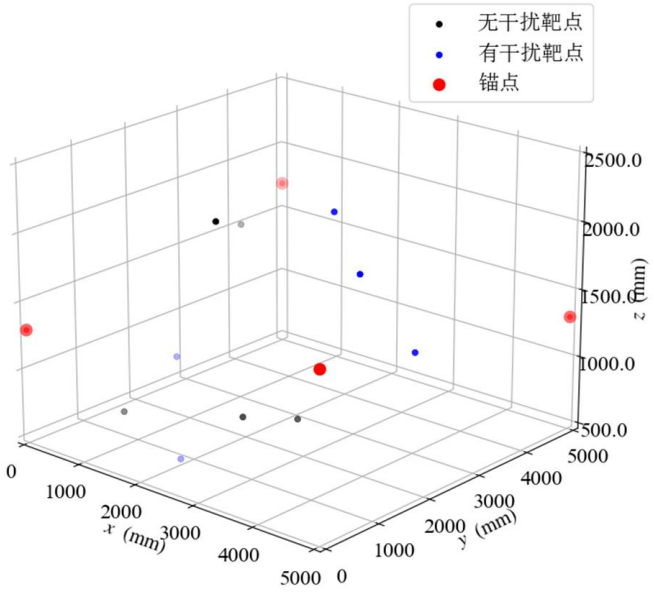


图 5-10 实验场景 1 下 10 组数据定位结果

至此，完成第二问的全部工作内容。

六、问题三模型的建立与求解

为了说明模型的推广能力和普适程度，本文通过另一实验场景采集到的原始数据，对这些数据的空间位置进行标定，根据标定结果，验证了定位模型在多种实验场景下的可靠性，解决了所建立模型的泛化问题。

6.1 不同场景适用性分析

在这一节中，主要从理论的角度，分析已建立的定位模型的泛化能力，由于上一问中，对“正常”和“异常”两类数据是分别建立定位模型的，所以该节中也各自分析其模型的推广可行性。

6.1.1 正常数据所受影响

如 5.1 节所述，在利用遗传算法对“正常”工作的数据点进行建模时，根据预估的位置点和锚点的距离之和与靶点到各锚点距离值和的差值作为寻优函数，最后找到的使该函数值最小的点坐标作为定位结果。可见，由于定位过程中不涉及对待测靶点真实坐标的使用，更换实验场景后，该算法的核心思想没有受到影响，并不会存在由于真实坐标缺失使得对“正常”工作的数据集，无法标定其空间位置的情况。

6.1.2 异常数据所受影响

同理，本文所建立的遗传—三边定位耦合空间定位模型，在用于求解“异常”工作点的空间位置时，没有使用题目提供的各靶点的真实坐标，所以当锚点坐标随实验场景发生变化时，同样不会对这类数据点的定位产生任何影响。

6.1.3 误差校正模型所受的影响

值得注意的是，在上一问建立基于神经网络方法的误差校正模型时，本文是通过靶点的预估位置和题目给定的靶点真实位置这两个已知量求解的，所以，若新的实验场景中，待求坐标的真实值并没有给出，那么，该误差修正模型便无法实现了，不过亦如上一章的

结果讨论所述，即便不存在对误差数据的训练，之前建立的两种定位模型，其精度和准确性也可控制在题目要求的厘米级，颇具可靠性，所以在该问中，不必担心由于误差校正模型的缺失导致求解精度、准度不足的问题。

通过以上分析，说明本文在前一问中所建立的空间定位模型，即便在这一问中更换过锚点的位置，仍具备相当的可操作性。所以，对于这一问的求解思路，和上一章是相近的，只需要在原程序基础上，修改其锚点坐标即可，详细求解思路流程图如下：

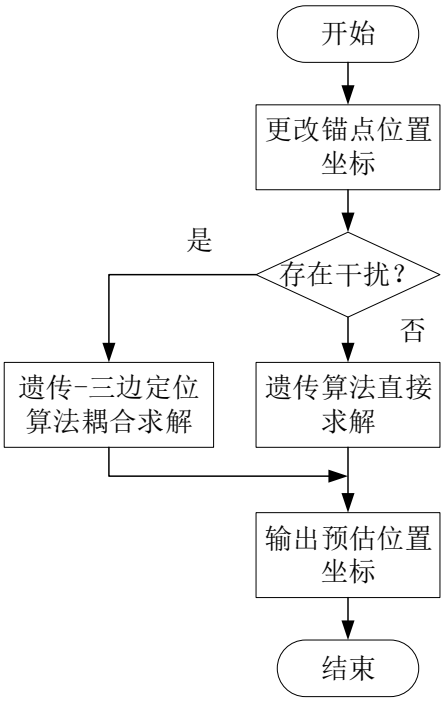


图 6-1 问题三求解思路流程图

6.2 不同场景数据精准定位结果

具体下一节中，代入题目数据，求解得到各个待测点的标定坐标值。其中，已知前 5 组为“正常”数据点，而后 5 组为“异常”数据点。得到预估的 10 个靶点空间位置如下表：

表 6-1 实验场景 2 下 10 组数据定位结果

样例点序号	*x*/mm	*y*/mm	*z*/mm
1	3664.66	2204.42	1207.08
2	4233.57	1723.86	1062.79
3	3195.48	1742.96	1141.67
4	2554.11	1917.99	2204.90
5	500.29	61.29	523.24
6	4606.71	2204.40	1300.66
7	2364.66	1716.89	730.41
8	1846.15	1600.90	964.70
9	2097.84	669.32	837.22
10	1099.24	674.04	852.35

得到的前五个在“正常”工况下测得的数据点空间坐标分别为(3664.66,2204.42,1207.08)、(4233.57,1723.86,1062.79)、(3195.48,1742.96,1141.67)、(2554.11,1917.99,2204.9)、(500.29,61.29,523.24)；后五个在“异常”工况下测得的数据点，其预估空间坐标分别为(4606.71,2204.40,1300.66)、(2364.66,1716.89,730.41)、(1846.15,1600.9,964.7)、(2097.84,669.32,837.22)、(1099.24,674.04,852.35)

在三维空间内，标识出各自的空间位置如下所示：

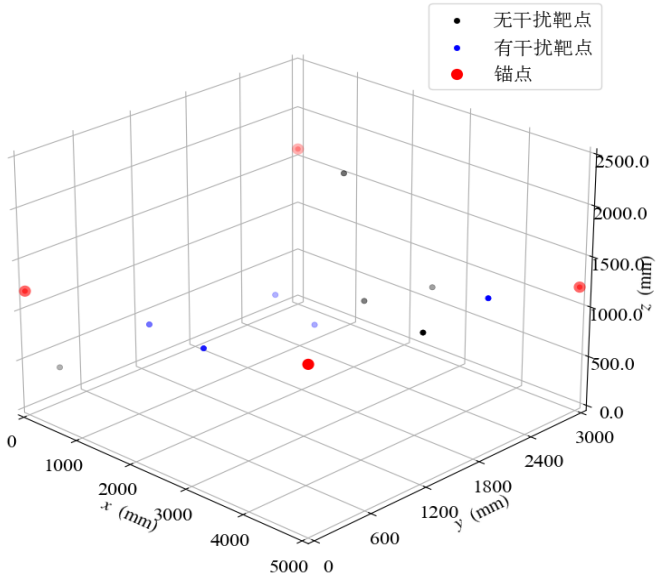


图 6-2 实验场景 2 下 10 组数据定位结果

至此，完成任务三中对已建立精确定位模型的泛化和推广工作，对新数据对应空间坐标的成功求解，说明建立的模型具备相当高的可靠性和普适性。

七、问题四模型的建立与求解

为了实现数据有无干扰的精准识别，考虑到干扰存在与否对三边定位模型预测的精度有重大影响，本文基于三边定位方法建立干扰识别分类模型；首先采用三边定位模型计算附件 1 中有、无干扰存在时的数据误差，获取大量数据特征；之后建立单目标优化模型求解最优误差界限，以此作为干扰识别分类判据；再利用附件 2 中同场景下测试数据对该分类模型的有效性进行验证；最后计算给出附件 4 中数据有无干扰的识别结果。

7.1 基于三边定位的干扰识别分类模型

7.1.1 三边定位算法原理

本文所解决的问题是基于 TOF 测距原理的 UWB 定位问题，具体而言是依据三维空间中分布着的 4 个锚点定位基站，分别测量出其与靶点目标之间的距离，基于定位基站坐标及所测四个距离求出靶点目标的坐标值[18]。锚点与靶点的三维定位示意图如图 7-1 所示。

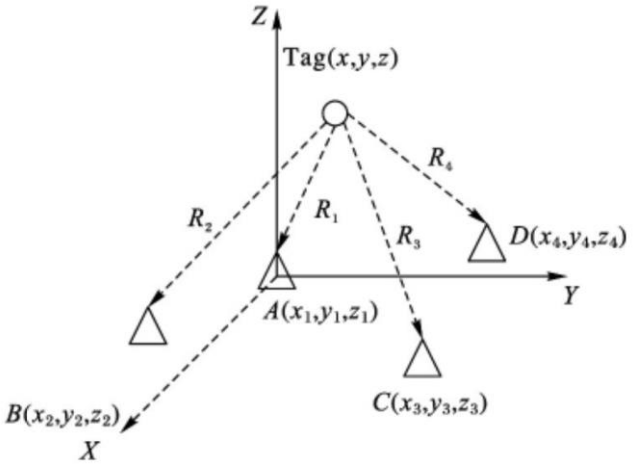


图 7-1 基于 TOF 的三维定位分布图[19]

由图 7-1 所示，已知四个锚点的坐标分别为(x_1, y_1, z_1)、(x_2, y_2, z_2)、(x_3, y_3, z_3)、(x_4, y_4, z_4)，并且知道 Tag 点到四个锚点的距离分别为 R_1、R_2、R_3、R_4，为求 Tag 点坐标，可以列出如下方程组：

$$\begin{cases}(x-x_1)^2+(y-y_1)^2+(z-z_1)^2=R_1^2\\(x-x_2)^2+(y-y_2)^2+(z-z_2)^2=R_2^2\\(x-x_3)^2+(y-y_3)^2+(z-z_3)^2=R_3^2\\(x-x_4)^2+(y-y_4)^2+(z-z_4)^2=R_4^2\end{cases} \tag{7-1}$$

式(7-1)中各方程的问题在于含有非线性项 x^2，y^2 和 z^2，本文通过方程相减来消除这些非线性项，使用前三个方程依次减去第四个方程，由此可以得到 3 个全新的方程，是关于 x, y, z 的线性方程。线性方程易于求解，可采用最小二乘法进行求解，可以将新方程组写成如下形式：

$$B=A\begin{bmatrix}x\\y\\z\end{bmatrix} \tag{7-2}$$

其中：

$$B=\begin{bmatrix}R_1^2-x_1^2-y_1^2-z_1^2-R_4^2+x_4^2+y_4^2+z_4^2\\R_2^2-x_2^2-y_2^2-z_2^2-R_4^2+x_4^2+y_4^2+z_4^2\\R_3^2-x_3^2-y_3^2-z_3^2-R_4^2+x_4^2+y_4^2+z_4^2\end{bmatrix} \tag{7-3}$$

$$A=-2\begin{bmatrix}x_1-x_4 & y_1-y_4 & z_1-z_4\\x_2-x_4 & y_2-y_4 & z_2-z_4\\x_3-x_4 & y_3-y_4 & z_3-z_4\end{bmatrix} \tag{7-4}$$

由于恰好有 4 个锚点得到了 3 个方程求解 3 个未知数，通过求解以下方程即可得到靶点 P 的位置坐标：

$$P=A^{-1}B \tag{7-5}$$

7.1.2 干扰识别分类判据

由 7.1.1 节算法原理可以看出，三边定位算法属于解析算法，能够用于定位预测问题模型的求解，靶点到锚点的测量距离的精确度直接影响到模型求解定位结果的精度。即一般而言，当靶点与锚点之间距离的测量值足够精准时，三边定位算法求解结果足够精确，当距离测量值误差较大时，三边定位算法的求解结果偏差会明显增大；反之可以认为如果三边定位算法求解误差较大，则说明靶点与锚点间的距离测量值大概率存在误差，而干扰的存在是测量值误差的产生原因，故认为三边定位求解的误差与干扰存在显著关系。因此，可以实现依据三边定位算法的求解误差来判别该锚点数据的测量是否存在干扰。

针对附件 1 中各数据样本，使用三边定位算法计算出样本数据的测量误差。测量误差的计算方法可参见式（5-5）。同一个测点位置对应一组正常数据与一组异常数据，正常数据可求得一组误差，异常数据分为两类各求出一组误差，采用两组数据求平均的方法对异常数据的误差进行处理。以同一位置对应正常数据误差为横坐标，以异常数据误差为纵坐标，绘制出同一位置有无干扰情况下的误差分布图，如图 7-2 所示。

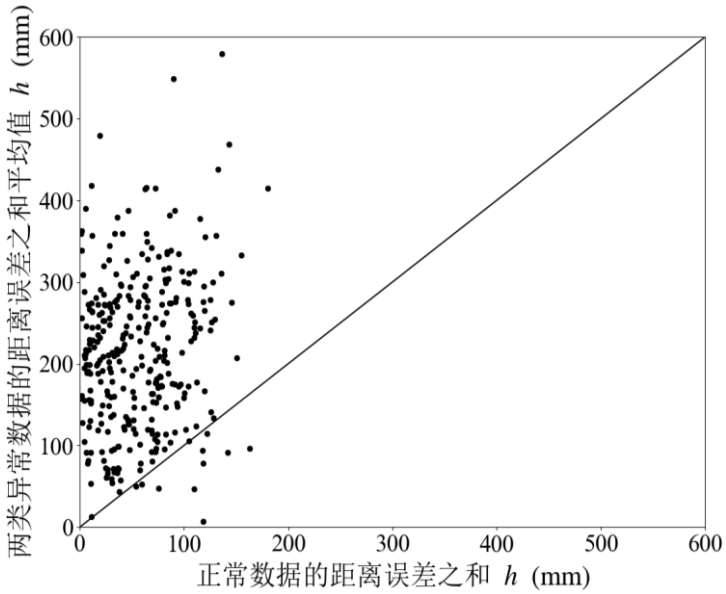


图 7-2 有、无干扰情况下误差对比图

如图 7-2 所示，图中斜率为 1 的直线代表同一位置正常数据与异常数据测量误差相等的情况，该直线之上代表异常数据误差高于正常数据，反之代表正常数据误差高于异常数据。由此看出总共 324 个靶点位置中，仅有 10 处位置正常数据定位误差高于异常数据，充分说明了本文所建误差分类模型在原理上是可靠的，即当三边定位模型计算出的靶点位置与实际位置偏差较大时，大概率认为测量存在干扰。

为建立量化指标直观得出数据存在干扰与否的判定依据，本文以判断错误数最少为目标进行优化，建立关于区分干扰点与非干扰点的误差临界值的单目标优化模型，可以表述如下：

$$\min\ N \tag{7-6}$$

$$s.t.\begin{cases} N = err(\text{正常}) + err(\text{异常}) \\ 0 \le err(\text{正常}) \le 324 \\ 0 \le err(\text{异常}) \le 648 \end{cases} \tag{7-7}$$

其中，N 表示正常点及异常点的判断错误总数。

确定误差临界值后，认为误差高于临界值的正常点以及误差低于临界值的正常点为判

断错误，由此统计出不同误差临界值所对应的判断错误数如图 7-3 所示。

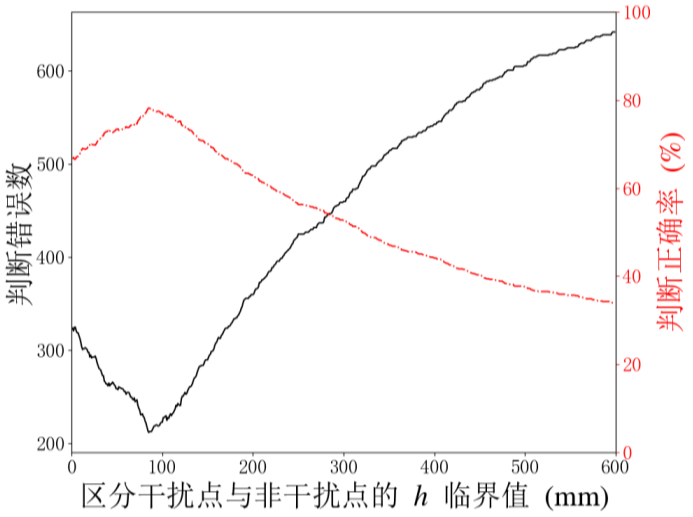


图 7-3 单目标优化结果

由图 7-3 单目标优化结果可以看出，将区分干扰点与非干扰点的误差临界值设为 87.5 mm 时，模型对应的判断错误数最少，即此时模型判断正确率最高，可以达到 78.19%。

7.2 分类模型的有效性验证

在使用 7.1 节所建干扰识别分类模型对附件 4 数据进行判别之前，需对模型进行验证，以说明模型的有效性。本文采用附件 2 中采集于相同场景下（实验场景 1）的测试数据对分类模型进行验证，使用三边定位算法对其误差进行计算，计算结果表 7-1 所示，其中数据编号 1~5 表示正常数据，数据编号 6~10 表示异常数据。

表 7-1 测试数据误差计算结果

数据编号	1	2	3	4	5
误差/mm	8.73	22.10	4.81	31.18	22.06
数据编号	6	7	8	9	10
误差/mm	95.01	269.62	241.34	36.07	416.06

由表 7-1 可以看出，正常数据的预测距离误差均小于 87.5 mm，异常数据中除编号 9 的误差为 36.07 低于临界值之外，其余误差均高于临界值。故使用 7.1 所建基于三边定位的干扰识别分类模型使用测试集数据成功验证了其有效性，针对预测集数据识别干扰成功的概率高达 90%。

7.3 附件 4 数据干扰识别结果

经过测试集数据的验证，充分证明本文所建干扰识别分类模型是可靠的，由此对题中所给附件 4 中数据进行干扰识别分析。采用三边定位法计算出各数据误差如表 7-2 所示。

表 7-2 附件 4 数据误差计算结果

数据编号	1	2	3	4	5
误差/mm	10.23	209.62	33.78	109.66	198.16
数据编号	6	7	8	9	10
误差/mm	115.77	135.19	39.60	34.31	14.41

由表 7-2 可以看出，数据编号为 1、3、8、9、10 的五个点计算误差低于 87.5，认为这五个点为“正常”点，未受到干扰；数据编号为 2、4、5、6、7 的五个点为“异常”点。

八、问题五模型的建立与求解

为了解决一运动靶点在空间上的轨迹定位问题，本文将上述各个模型联合，识别其信号干扰情况，确定其各时刻位置，并结合 Kalman 滤波方法筛去运动时产生的非视距误差，得到了一连续、平滑的空间运动轨迹，反映了测点一段时间内的真实运动情况。

8.1 基于 Kalman 滤波的运动轨迹定位模型

在对室内的运动物体进行定位时，存在距离监测设备本身以及外部条件导致的信号噪声，这些噪声信号若没有经过恰当地处理，会对最终结果的准确性造成极大的影响，目前，动态轨迹的测距信号降噪方法通常涉及平均滤波、高斯滤波、小波方法等[20]，对于该题中受到的干扰，具体表现是对某个锚点的传输路径进行障碍物阻隔，即为非视距传播造成的误差，它会使时间测量值发生正向的偏差。针对这种类型的干扰导致的信号噪声，借助 Kalman 滤波的方法[21, 22]，可以有效地去噪。

Kalmon 滤波的方程包括状态方程和观测方程两个部分，其本质是用一种递推的方法，根据系统本身的状态转移矩阵和观测矩阵，当测量方程已知的条件下，从一系列存在噪声的实验数据中，及时地最优估计系统不同时刻的状态。其状态方程和观测方程的离散化形式如下：

$$\begin{cases} X_k = \Phi_{k/k-1} X_{k-1} + \Gamma_{k/k-1} W_{k-1} \\ Z_k = H_k X_k + V_k \end{cases} \tag{8-1}$$

在这种算法下，可以根据当前状态进一步预测下一刻的状态：

$$\hat{X}_{k/k-1} = \Phi_{k/k-1} \hat{X}_{k-1} \tag{8-2}$$

根据当前状态进一步预测均方根误差：

$$P_{k/k-1} = \Phi_{k/k-1} P_{k-1} \Phi^T{}_{k/k-1} + \Gamma_{k-1} Q_{k-1} \Gamma^T{}_{k-1} \tag{8-3}$$

滤波增益、下一刻状态估计以及状态估计的均方根误差分别为：

$$\begin{cases} K_k = P_{k/k-1} H^T{}_k (H_k P_{k/k-1} H^T{}_k + R_k)^{-1} \\ \hat{X}_k = \hat{X}_{k/k-1} + K_k (Z_k - H_k \hat{X}_{k/k-1}) \\ P_k = (I - K_k H_k) P_{k/k-1} \end{cases} \tag{8-4}$$

引入 Kalman 滤波方法，本题可依据如下求解思路构建动态物体的空间轨迹定位：

（1）将题设中给出的不同时刻测得的数据样点，视作时间上独立的静态数据点，根据干扰识别模型，判别各样点分属于“正常”或是“异常”工作点；

（2）根据其类别，分别代入遗传算法定位模型和遗传—三边定位算法模型求解、标定其三维空间坐标；

（3）将三维坐标按数据采集时间顺序排列，初步得到其未经滤波的空间运动轨迹；

（4）对所得的样点，通过 Kalman 滤波方法，过滤去采样中的非视距信号噪声，得到精准的物体运动轨迹图。

综上，可总结该问的求解思路流程图如下：

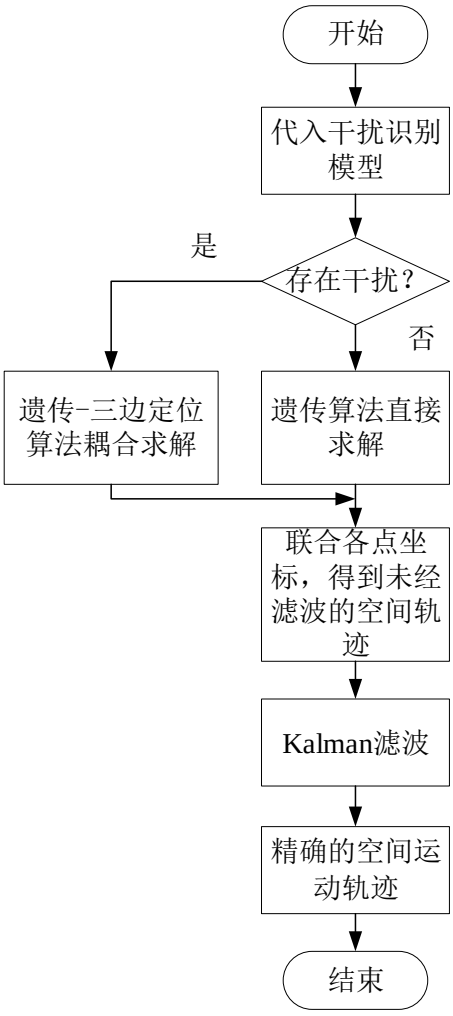


图 8-1 问题五求解思路流程图

8.2 运动轨迹图绘制

基于以上求解思路，代入初始数据，得到未经滤波和经过滤波的两条 z 方向的曲线在图 8-2 中表示如下：

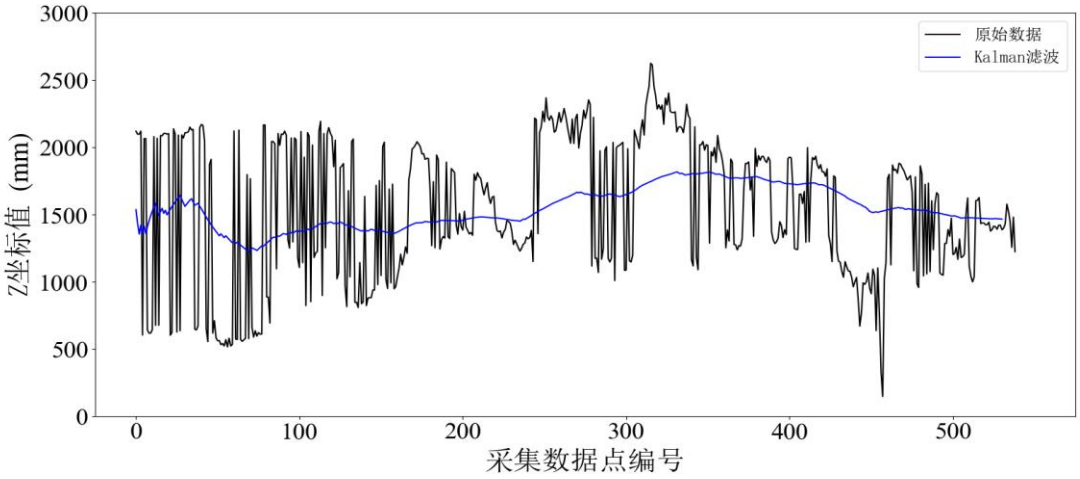


图 8-2 动态情境下 z 方向定位结果

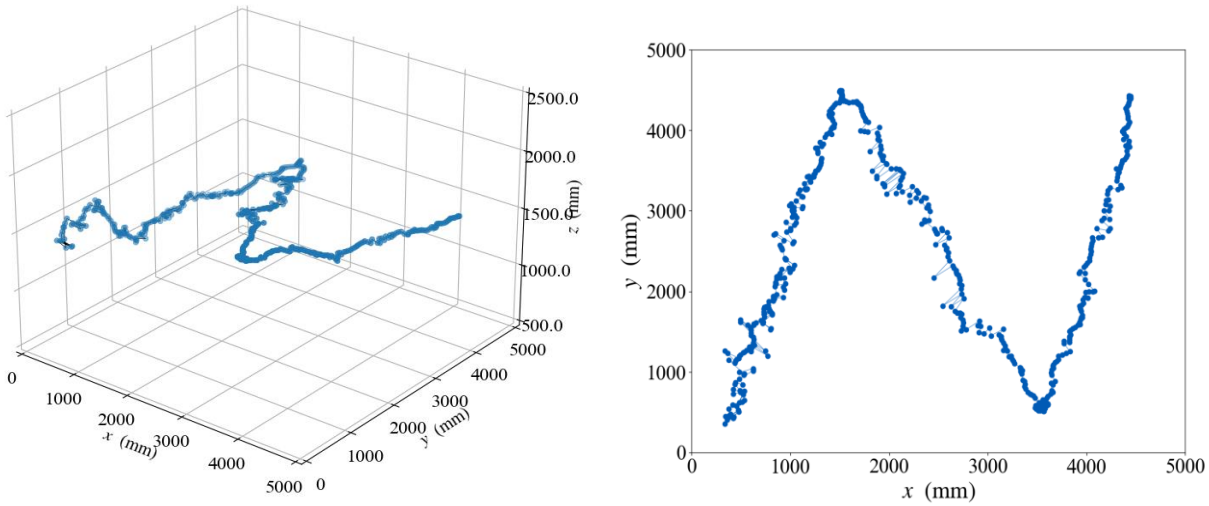


图 8-3 空间上（左）和 XOY 平面上（右）定位轨迹图

如上图所示，在未滤波前，各数据的预测空间点坐标有一定程度的波动，而经过滤波后，动态物体的空间运动轨迹变得连续、平滑，说明 Kalman 滤波方法有效地过滤了由于 NLOS 非视距因素导致的测距误差，结合原有的空间定位模型，有效地实现了对运动物体的空间运动轨迹还原。

至此，完成了动态空间定位模型的建立。

九、模型的评价与推广

9.1 模型评价

针对问题一，本文通过 Python 编程，建立样本数据抓取和筛选模型对题目提供的数据进行有效地遴选，对初选的数据进行进一步滤波和校正，减小了测距过程中的系统误差。该数据抓取即筛选模型，成功滤去各数据集中严重偏离常值的样品，达到了题目的要求，极大提高原始数据的可视性和可用性，对于后续模型的建立具有重大意义。

针对问题二，为实现对不同类型数据样本的空间定位，本文先是提出了遗传算法和遗传-三边定位方法耦合的两种定位算法，采用这两种定位算法，实现了对锚点工作不受干扰和锚点工作受到障碍物干扰两种工况下，不同数据样本的精确定位，通过算法在样例点上的应用，证明了建立算法的可行性；又基于神经网络算法，建立定位模型的误差修正模型，进一步提高了定位结果的精准性；引入平均绝对误差和均方根误差两个评判标准，对模型的精确度进行评价，评价结果表明定位结果和真值偏差较小，定位结果十分精准。最后，利用建立的定位模型，代入该实验场景下的另一组样例数据点，对其进行空间定位，进一步说明模型简明有效、可操作性强。

针对问题三，将本模型推广应用在另一锚点坐标不同的实验场景下，成功对其中测得的一组样例数据点进行分析、准确定位，说明所建立模型的泛化性强、普适性高。

针对问题四，本文充分考虑距离测量精度对三边定位算法预测误差的影响，提取出干扰与定位预测误差之间的特征关系，并以识别错误数最少为目标函数，建立单目标优化模型，确定出判断正确率最高时的误差临界值，给出量化判定指标，成功实现对测试集信号有无干扰的识别。本文所创建的量化判据有利于直观评判，且大大提高了干扰识别效率，为解决 UWB 精确定位的重点和难点问题提供了一种参考。

针对问题五，综合运用前文所建定位预测和干扰识别分类模型，分析一个动态靶点的

运动情况，甄别该靶点运动时信号传输受干扰状况，结合 Kalman 滤波方法，消除运动过程中非视距误差对定位结果的影响，对其一段时间内在空间位置内的移动轨迹进行精确定位，成功绘制出该测点的运动曲线。结果表明建立的干扰识别模型稳定性高、定位模型的性能良好，可以由静态测点定位推广到深层次的动态轨迹定位上。

9.2 模型改进

问题一建立的数据筛选模型，对数据集内的样本进行了有效遴选，剔除其中的异常值，但大部分数据集仍有相当庞大的样本，结合其它使用多种滤波和数据处理算法，可以进一步减小剩余样本量，减小模型计算成本。

问题二建立的定位模型，在预测“异常”数据集的空间位置时，存在一定程度精度不足的问题，结合除三边定位算法外更高精度的定位方法，可以进一步提高模型的精确性。

参考文献

[1] Yassin A, Nasser Y, Awad M, et al. Recent advances in indoor localization: A survey on theoretical approaches and applications[J]. IEEE Communications Surveys & Tutorials, 2016, 19(2): 1327-1346.

[2] Dardari D, Closas P, Djurić P M. Indoor tracking: Theory, methods, and technologies[J]. IEEE Transactions on Vehicular Technology, 2015, 64(4): 1263-1278.

[3]Howitt I, Jore G. Low-Rate Wireless Personal Area Networks (LR-WPANs)[C]//Wireless Communications and Networking. 2011: 1-314.

[4]高健. 基于 UWB 通信的多目标室内精确定位研究[D].合肥工业大学,2020.

[5]云酷科技，全面解析 UWB 及其应用场景，http://www.qianjia.com/zhike/html/2020-02/11_19601.html，2021 年 10 月 14 日。

[6]李伟. 室内超宽带定位导航系统的设计与实现[D].电子科技大学,2017.

[7]吴昊,王深远. 基于拉伊达准则的 GNSS 变形监测异常数据识别算法[J]. 科技创新与生产力,2019(1):30-34. DOI:10.3969/j.issn.1674-9146.2019.01.030.

[8] Siegmund D. Error probabilities and average sample number of the sequential probability ratio test[J]. Journal of the Royal Statistical Society: Series B (Methodological), 1975, 37(3): 394-401.

[9] 严嘉祺. 基于 UWB 的室内定位系统的算法与误差分析[D].哈尔滨工业大学,2020.

[10] 李汉波. 基于 UWB 定位的室内无人机编队系统设计与实现[D].厦门大学,2019.

[11] 王金鑫,赖旭芝,吴敏,Simon X.Yang.基于遗传算法的三维无线传感器网络定位新算法[J].高技术通讯,2008,18(06):579-584.

[12] 温培博,李行健,杨文.多基站下基于信号到达时间的室内三维定位[J].数学的实践与认识,2017,47(14):151-161.

[13]陈秋莲,王成栋.基于 Matlab 遗传算法工具箱的优化计算实现[J].现代电子技术,2007(02):124-126+129.

[14]谢亚琴,张业荣.基于信号到达时间的 UWB 定位算法[J].南京邮电大学学报(自然科学版),2007(03):71-75.

[15] Guo Fei. scikit-opt[CP].Github,2021. https://github.com/guofei9987/scikit-opt

[16] 蔡祯祺. 基于数值天气预报 NWP 修正的 BP 神经网络风电功率短期预测研究[D].浙江大学,2012.

[17] 卞佳兴,朱荣,陈玄.基于改进双向测距-到达时间差定位算法的超宽带定位系统[J].计算

机应用,2017,37(09):2496-2500+2511.
[18] Kun Jin. UWB localization algorithm based on BP neural network compensation extended Kalman filter[J]. Journal of Physics: Conference Series,2021,1885(4):
[19] 马琳, 董赫, 谭学治,等. 一种联合最小二乘和三边定位的超宽带室内定位方法.
[20] 曹子腾. 室内定位技术的研究及其在北斗导航服务中的应用[D].石家庄铁道大学,2020.
[21] 王川阳,王坚,宁一鹏,余航.超宽带定位的降噪方法研究[J].测绘科学,2019,44(04):175-181.
[22] 李静,刘琚.用卡尔曼滤波器消除 TOA 中 NLOS 误差的三种方法[J].通信学报,2005(01):130-135+141.

附录（程序）

1. 任务一源码

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
    from scipy import spatial, integrate, interpolate, optimize
    import sys
sys.path.append("Lib/scikit")
from sko.GA import GA              # 用户自定义库
from sko.SA import SA              # 用户自定义库
from sko.PSO import PSO            # 用户自定义库
import Lib.myFunctions as my   # 用户自定义库

    """ matplotlib 字体设置 """
config = {"font.family": 'serif', "font.size": 20, "mathtext.fontset": 'stix', "font.serif": ['SimSun'], }
plt.rcParams.update(config)
plt.rcParams['axes.unicode_minus'] = False   # 用来正常显示负号

    """ 第一问 """
not2D_num = 0
sigema3_R_min = np.zeros((324, 4))
sigema3_R_max = np.zeros((324, 4))
sigema3_W_e1_min = np.zeros((324, 4))
sigema3_W_e1_max = np.zeros((324, 4))
sigema3_W_e2_min = np.zeros((324, 4))
sigema3_W_e2_max = np.zeros((324, 4))
A1_e1, A2_e1, A3_e1, A4_e1 = [], [], [], []
A1_e2, A2_e2, A3_e2, A4_e2 = [], [], [], []
    """ 数据处理 """
    for i in range(324):
    filename_R = './正常数据/' + str(i+1) + '.正常.txt'
    filename_W = './异常数据/' + str(i+1) + '.异常.txt'
```

```
    df_R = pd.read_table(filename_R, sep=':', skiprows=1, names=['x1', 'time', 'x2', 'tagID', 'type',
'length', 'length2', 'Serial No.', 'No.'])
    df_W = pd.read_table(filename_W, sep=':', skiprows=1, names=['x1', 'time', 'x2', 'tagID', 'type',
'length', 'length2', 'Serial No.', 'No.'])
    df_R = df_R.iloc[:, [1, 3, 4, 5, 7, 8]].astype(float)
    df_W = df_W.iloc[:, [1, 3, 4, 5, 7, 8]].astype(float)
    # 按照点的类型分类
    df_R_p, df_W_p = [0, 0, 0, 0], [0, 0, 0, 0]
    for t in range(4):
        df_R_p[t] = df_R.loc[df_R['type'] == t, :]
        df_R_p[t].index = np.arange(df_R_p[t].shape[0])
        df_W_p[t] = df_W.loc[df_W['type'] == t, :]
        df_W_p[t].index = np.arange(df_W_p[t].shape[0])
    # 求平均值，标准差
    df_R_p_mean, df_W_p_mean, df_R_p_std, df_W_p_std = [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]
    for t in range(4):
        df_R_p_mean[t] = df_R_p[t].loc[:,'length'].mean()
        df_W_p_mean[t] = df_W_p[t].loc[:,'length'].mean()
        df_R_p_std[t] = df_R_p[t].loc[:,'length'].std()
        df_W_p_std[t] = df_W_p[t].loc[:,'length'].std()

    """  ----- 正常数据筛选与处理 -----  """
    # 求正常数据.txt 3sigema 内的保留数据
    cut_array_R = np.array([])
    for t in range(4):
        sigema3_R_min[i,t] = df_R_p_mean[t] - 3 * df_R_p_std[t]
        sigema3_R_max[i,t] = df_R_p_mean[t] + 3 * df_R_p_std[t]
        out3sigema_R_index = df_R_p[t].loc[(df_R_p[t]['length'] < sigema3_R_min[i,t]) |
(df_R_p[t]['length'] > sigema3_R_max[i,t]), 'length'].index.to_numpy()
        cut_array_R = np.append(cut_array_R, out3sigema_R_index)
    cut_array_R = np.unique(cut_array_R)
    left_array_R = np.setdiff1d(df_R_p[0].index.to_numpy(), cut_array_R)
    df_R_left = [0, 0, 0, 0]
    for t in range(4):
        df_R_left[t] = df_R_p[t].loc[left_array_R, :]
    #print('编号', i+1, '正常数据删除数：', -(df_R_left[t].shape[0] - df_R_p[t].shape[0]))
    #print('编号', i + 1, '正常数据中保留个数：', df_R_left[t].shape[0])
    df_R_final = df_R_left[0].loc[:, ['tagID', 'Serial No.', 'No.']]
    df_R_final.loc[:, 'A0'] = df_R_left[0]['length']
    df_R_final.loc[:, 'A1'] = df_R_left[1]['length']
    df_R_final.loc[:, 'A2'] = df_R_left[2]['length']
    df_R_final.loc[:, 'A3'] = df_R_left[3]['length']

    """  ----- 异常数据处理 -----  """
```

```
    # 统计异常维度数目:使用方差排序取较大两个
    err_num, nomal_num = 2, 0
    sort_mean_R = np.argsort(np.array(df_W_p_std))
    err_index_list = sort_mean_R[[-1, -2]]

    # 统计异常维度下，去除正常数据标准 3sigema 内数据后的点个数
    out3sigema_array_W, in3sigema_array_W = [0, 0, 0, 0], [0, 0, 0, 0]
    out3sigema_array_W_numlist, in3sigema_array_W_numlist, err_pointnum_list = [], [], []
    for t in err_index_list:
        out3sigema_W_index = df_W_p[t].loc[(df_W_p[t]['length'] < (df_R_p_mean[t] - 3 *
df_R_p_std[t])) | (df_W_p[t]['length'] > (df_R_p_mean[t] + 3 * df_R_p_std[t])), 'length'].index.to_numpy()
        out3sigema_array_W[t] = out3sigema_W_index   # 3σ 外
        in3sigema_array_W[t] = np.setdiff1d(df_W_p[t].index.to_numpy(), out3sigema_array_W[t]) #3σ 内
        out3sigema_array_W_numlist.append(out3sigema_array_W[t].shape[0])
# 3σ 外计数
        in3sigema_array_W_numlist.append(in3sigema_array_W[t].shape[0])
# 3σ 内计数
        err_pointnum_list.append(df_W_p[t].shape[0])
# 异常维度的点总数计数
    double_point_num = out3sigema_array_W_numlist[0] + out3sigema_array_W_numlist[1] -
err_pointnum_list[0]

    e1, e2 = err_index_list[0], err_index_list[1]
    if out3sigema_array_W_numlist[0] < 40:
        out3sigema_array_W[e1] = in3sigema_array_W[e2]
        out3sigema_array_W_numlist[0] = out3sigema_array_W[e1].shape[0]
        double_point_num = out3sigema_array_W_numlist[0] + out3sigema_array_W_numlist[1] -
err_pointnum_list[0]
    if out3sigema_array_W_numlist[1] < 40:
        out3sigema_array_W[e2] = in3sigema_array_W[e1]
        out3sigema_array_W_numlist[1] = out3sigema_array_W[e2].shape[0]
        double_point_num = out3sigema_array_W_numlist[0] + out3sigema_array_W_numlist[1] -
err_pointnum_list[0]

    # 异常点中去除同时异常点，然后计算平均值与标准差，3σ 筛选
    e1, e2 = err_index_list[0], err_index_list[1]
    jiaoji = np.intersect1d(out3sigema_array_W[e1], out3sigema_array_W[e2])
    out3sigema_array_W[e1] = np.setdiff1d(out3sigema_array_W[e1], jiaoji)
    out3sigema_array_W[e2] = np.setdiff1d(out3sigema_array_W[e2], jiaoji)
    df_W_p_e1, df_W_p_e2 = [0, 0, 0, 0], [0, 0, 0, 0]
    for t in range(4):
        df_W_p_e1[t] = df_W_p[t].loc[out3sigema_array_W[e1], :]
        df_W_p_e2[t] = df_W_p[t].loc[out3sigema_array_W[e2], :]
```

```
    df_W_p_e1_mean, df_W_p_e2_mean, df_W_p_e1_std, df_W_p_e2_std = [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]
    for t in range(4):
        df_W_p_e1_mean[t] = df_W_p_e1[t].loc[:,'length'].mean()
        df_W_p_e2_mean[t] = df_W_p_e2[t].loc[:, 'length'].mean()
        df_W_p_e1_std[t] = df_W_p_e1[t].loc[:,'length'].std()
        df_W_p_e2_std[t] = df_W_p_e2[t].loc[:, 'length'].std()

    cut_array_W_e1, cut_array_W_e2 = np.array([]), np.array([])
    for t in range(4):
        sigema3_W_e1_min[i, t] = df_W_p_e1_mean[t] - 3 * df_W_p_e1_std[t]
        sigema3_W_e1_max[i, t] = df_W_p_e1_mean[t] + 3 * df_W_p_e1_std[t]
        sigema3_W_e2_min[i, t] = df_W_p_e2_mean[t] - 3 * df_W_p_e2_std[t]
        sigema3_W_e2_max[i, t] = df_W_p_e2_mean[t] + 3 * df_W_p_e2_std[t]
        out3sigema_W_e1_index = df_W_p_e1[t].loc[(df_W_p_e1[t]['length'] < sigema3_W_e1_min[i, t]) | (df_W_p_e1[t]['length'] > sigema3_W_e1_max[i, t]), 'length'].index.to_numpy()
        out3sigema_W_e2_index = df_W_p_e2[t].loc[(df_W_p_e2[t]['length'] < sigema3_W_e2_min[i, t]) | (df_W_p_e2[t]['length'] > sigema3_W_e2_max[i, t]), 'length'].index.to_numpy()
        cut_array_W_e1 = np.append(cut_array_W_e1, out3sigema_W_e1_index)
        cut_array_W_e2 = np.append(cut_array_W_e2, out3sigema_W_e2_index)
    cut_array_W_e1 = np.unique(cut_array_W_e1)
    cut_array_W_e2 = np.unique(cut_array_W_e2)

    cut_array_W = np.union1d(cut_array_W_e1, cut_array_W_e2)
    left_array_W_e1 = np.setdiff1d(out3sigema_array_W[e1], cut_array_W)
    left_array_W_e2 = np.setdiff1d(out3sigema_array_W[e2], cut_array_W)
    left_array_W = np.union1d(left_array_W_e1, left_array_W_e2)

    #print(cut_array_W_e1, cut_array_W_e2)
    df_W_left = [0, 0, 0, 0]
    for t in range(4):
        df_W_left[t] = df_W_p[t].loc[left_array_W, :]
    df_W_left_e1, df_W_left_e2 = [0, 0, 0, 0], [0, 0, 0, 0]
    for t in range(4):
        df_W_left_e1[t] = df_W_p[t].loc[left_array_W_e1, :]
        df_W_left_e2[t] = df_W_p[t].loc[left_array_W_e2, :]
    #print('编号', i+1, '异常数据删除数：', -(df_W_left[t].shape[0] - df_W_p[t].shape[0]))
    #print('编号', i+1, '异常数据中保留个数：', df_W_left[t].shape[0])
    df_W_final = df_W_left[0].loc[:, ['tagID', 'Serial No.', 'No.']]
    df_W_final.loc[:, 'A0'] = df_W_left[0]['length']
    df_W_final.loc[:, 'A1'] = df_W_left[1]['length']
    df_W_final.loc[:, 'A2'] = df_W_left[2]['length']
    df_W_final.loc[:, 'A3'] = df_W_left[3]['length']
    df_W_final_e1 = df_W_left_e1[0].loc[:, ['tagID', 'Serial No.', 'No.']]
```

```
        df_W_final_e1.loc[:, 'A0'] = df_W_left_e1[0]['length']
        df_W_final_e1.loc[:, 'A1'] = df_W_left_e1[1]['length']
        df_W_final_e1.loc[:, 'A2'] = df_W_left_e1[2]['length']
        df_W_final_e1.loc[:, 'A3'] = df_W_left_e1[3]['length']
        df_W_final_e2 = df_W_left_e2[0].loc[:, ['tagID', 'Serial No.', 'No.']]
        df_W_final_e2.loc[:, 'A0'] = df_W_left_e2[0]['length']
        df_W_final_e2.loc[:, 'A1'] = df_W_left_e2[1]['length']
        df_W_final_e2.loc[:, 'A2'] = df_W_left_e2[2]['length']
        df_W_final_e2.loc[:, 'A3'] = df_W_left_e2[3]['length']
""" 输出"""
df_R_final.to_excel('./输出/第一问 excel/' + str(i+1) + '.正常.xlsx', index = True)
df_W_final.to_excel('./输出/第一问 excel/' + str(i+1) + '.异常.xlsx', index = True)
df_W_final_e1.to_excel('./输出/第一问 excel-异常分两类/' + str(i + 1) + '.异常-1.xlsx', index=True)
df_W_final_e2.to_excel('./输出/第一问 excel-异常分两类/' + str(i + 1) + '.异常-2.xlsx', index=True)
```

2. 任务二源码

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
    from scipy import spatial, integrate, interpolate, optimize
    import sympy
    import sys
sys.path.append("Lib/scikit")
from sko.GA import GA                # 用户自定义库
from sko.SA import SA                # 用户自定义库
from sko.PSO import PSO              # 用户自定义库
import Lib.myFunctions as my   # 用户自定义库

    """ matplotlib 字体设置 """
config = {"font.family": 'serif', "font.size": 20, "mathtext.fontset": 'stix', "font.serif": ['SimSun'], }
plt.rcParams.update(config)
plt.rcParams['axes.unicode_minus'] = False   # 用来正常显示负号

    """ 第二问 """
    A0 = np.array([0, 0, 1300])
A1 = np.array([5000, 0, 1700])
A2 = np.array([0, 5000, 1700])
A3 = np.array([5000, 5000, 1300])
    df_tag = pd.read_excel('./Data/Tag 坐标信息.xlsx')
df_tag_xyz = np.array([df_tag['x'], df_tag['y'], df_tag['z']])
A_dis = [0, 0, 0, 0]
```

```
i1, i2, i3 = 0, 0, 0
real_dis_list, A1234_list = [], []

    # 计算定位坐标
def triposition(A0, da, A1, db, A2, dc, A4, dd):
                    """ 三边定位 """
    xa, ya, za = A0[0], A0[1], A0[2]
    xb, yb, zb = A1[0], A1[1], A1[2]
    xc, yc, zc = A2[0], A2[1], A2[2]
    xd, yd, zd = A3[0], A3[1], A3[2]

    x, y, z = sympy.symbols("x, y, z")
    f1 = 2 * x * (xa - xd) + np.square(xd) - np.square(xa) + 2 * y * (ya - yd) + np.square(yd) - np.square(ya) +
2 * z * (za - zd) + np.square(zd) - np.square(za) - (np.square(dd) - np.square(da))
    f2 = 2 * x * (xb - xd) + np.square(xd) - np.square(xb) + 2 * y * (yb - yd) + np.square(yd) - np.square(yb)
+ 2 * z * (zb - zd) + np.square(zd) - np.square(zb) - (np.square(dd) - np.square(db))
    f3 = 2 * x * (xc - xd) + np.square(xd) - np.square(xc) + 2 * y * (yc - yd) + np.square(yd) - np.square(yc) +
2 * z * (zc - zd) + np.square(zd) - np.square(zc) - (np.square(dd) - np.square(dc))
    result = sympy.solve([f1, f2, f3], [x, y, z])

    locx, locy, locz = result[x], result[y], result[z]
    return [locx, locy, locz]

def disErr_scene1(tag_loc, distance4):
                    """ 误差计算 """
    tag_loc = np.array(tag_loc).astype(float)
    A0_ = np.array(A0)
    A1_ = np.array(A1)
    A2_ = np.array(A2)
    A3_ = np.array(A3)
    dis0 = np.linalg.norm(tag_loc - A0_)
    dis1 = np.linalg.norm(tag_loc - A1_)
    dis2 = np.linalg.norm(tag_loc - A2_)
    dis3 = np.linalg.norm(tag_loc - A3_)

    err = np.fabs(dis0 - distance4[0]) + np.fabs(dis1 - distance4[1]) + np.fabs(dis2 - distance4[2]) +
np.fabs(dis3 - distance4[3])
    return err

def sanbian_gene_opFunc(x):
        """ 遗传算法目标函数 """
    A_m = A_dis.copy()
    A_m[minus_object] = A_m[minus_object] - x
    res1 = triposition(A0, A_m[0], A1, A_m[1], A2, A_m[2], A3, A_m[3])
```

```
    err1 = disErr_scene1(res1, A_m)
    return err1

    def distance4(tag_loc):
        """ 距离计算函数 """
    tag_loc = np.array(tag_loc)
    A0 = np.array([0, 0, 1300])
    A1 = np.array([5000, 0, 1700])
    A2 = np.array([0, 5000, 1700])
    A3 = np.array([5000, 5000, 1300])
    return [np.linalg.norm(tag_loc - A0), np.linalg.norm(tag_loc - A1), np.linalg.norm(tag_loc - A2),
np.linalg.norm(tag_loc - A3)]

    df_tag = pd.read_excel('./Data/Tag 坐标信息.xlsx')
df_tag_xyz = np.array([df_tag['x'], df_tag['y'], df_tag['z']])

""" 距离系统误差校正 """
err_A1_list, err_A2_list, err_A3_list, err_A4_list = [],[],[],[]
mean_A1_list, mean_A2_list, mean_A3_list, mean_A4_list = [],[],[],[]
for i in range(324):
    filename_R = './输出/第一问 excel/' + str(i + 1) + '.正常.xlsx'
    filename_W = './输出/第一问 excel/' + str(i + 1) + '.异常.xlsx'
    df_R = pd.read_excel(filename_R)
    df_W = pd.read_excel(filename_W)
    #平均值
    df_R_mean, df_W_mean, df_R_std, df_W_std = [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]
    df_R_mean[0] = df_R.loc[:, 'A0'].mean()
    df_R_mean[1] = df_R.loc[:, 'A1'].mean()
    df_R_mean[2] = df_R.loc[:, 'A2'].mean()
    df_R_mean[3] = df_R.loc[:, 'A3'].mean()
    mean_A1_list.append(df_R_mean[0])
    mean_A2_list.append(df_R_mean[1])
    mean_A3_list.append(df_R_mean[2])
    mean_A4_list.append(df_R_mean[3])
    A1234 = np.array([df_R_mean[0], df_R_mean[1], df_R_mean[2], df_R_mean[3]])
    tag_xyz = [df_tag.loc[i,'x'], df_tag.loc[i,'y'], df_tag.loc[i,'z']]
    real_dis = np.array(distance4(tag_xyz))
    A1234_list.append(A1234)
    real_dis_list.append(real_dis)
    err_A1, err_A2, err_A3, err_A4 = A1234 - real_dis
    err_A1_list.append(err_A1)
    err_A2_list.append(err_A2)
    err_A3_list.append(err_A3)
    err_A4_list.append(err_A4)
```

```
    #print(err_A1, err_A2, err_A3, err_A4)

# 误差变化散点图、拟合
err_fit1 = np.polyfit(mean_A1_list, err_A1_list, 1)
err_fit2 = np.polyfit(mean_A2_list, err_A2_list, 1)
err_fit3 = np.polyfit(mean_A3_list, err_A3_list, 1)
err_fit4 = np.polyfit(mean_A4_list, err_A4_list, 1)
err_fit_curve1 = np.polyval(err_fit1, np.linspace(np.min(mean_A1_list), np.max(mean_A1_list),10))
err_fit_curve2 = np.polyval(err_fit2, np.linspace(np.min(mean_A2_list), np.max(mean_A2_list),10))
err_fit_curve3 = np.polyval(err_fit3, np.linspace(np.min(mean_A3_list), np.max(mean_A3_list),10))
err_fit_curve4 = np.polyval(err_fit4, np.linspace(np.min(mean_A4_list), np.max(mean_A4_list),10))
print(err_fit1, err_fit2, err_fit3, err_fit4)

fig = plt.figure()
plt.xlabel(r'A0 测量距离平均值  $\rm(mm)$', fontsize=30)
plt.ylabel(r'测量距离平均值与真实值误差  $\rm(mm)$', fontsize=30)
plt.xlim(0, 7000)
plt.ylim(-200, 200)
plt.xticks(np.around(np.linspace(0, 7000, 8), decimals=0), fontproperties='Times New Roman', size=26)
plt.yticks(np.around(np.linspace(-200, 200, 9), decimals=0), fontproperties='Times New Roman', size=26)

plt.scatter(mean_A1_list, err_A1_list, color='k', lw=0.5)
plt.plot(np.linspace(np.min(mean_A1_list), np.max(mean_A1_list),10), err_fit_curve1, 'k-', lw=1)
fig = plt.figure()
plt.xlabel(r'A1 测量距离平均值  $\rm(mm)$', fontsize=30)
plt.ylabel(r'测量距离平均值与真实值误差  $\rm(mm)$', fontsize=30)
plt.xlim(0, 7000)
plt.ylim(-200, 150)
plt.xticks(np.around(np.linspace(0, 7000, 8), decimals=0), fontproperties='Times New Roman', size=26)
plt.yticks(np.around(np.linspace(-200, 150, 8), decimals=0), fontproperties='Times New Roman', size=26)
plt.scatter(mean_A2_list, err_A2_list, color='r', lw=0.5)
plt.plot(np.linspace(np.min(mean_A2_list), np.max(mean_A2_list),10), err_fit_curve2, 'r-', lw=1)
fig = plt.figure()
plt.xlabel(r'A2 测量距离平均值  $\rm(mm)$', fontsize=30)
plt.ylabel(r'测量距离平均值与真实值误差  $\rm(mm)$', fontsize=30)
plt.xlim(0, 7000)
plt.ylim(-250, 100)
plt.scatter(mean_A3_list, err_A3_list, color='b', lw=0.5)
plt.plot(np.linspace(np.min(mean_A3_list), np.max(mean_A3_list),10), err_fit_curve3, 'b-', lw=1)
plt.xticks(np.around(np.linspace(0, 7000, 8), decimals=0), fontproperties='Times New Roman', size=26)
plt.yticks(np.around(np.linspace(-250, 100, 8), decimals=0), fontproperties='Times New Roman', size=26)
fig = plt.figure()
plt.xlabel(r'A3 测量距离平均值  $\rm(mm)$', fontsize=30)
```

```
plt.ylabel(r'测量距离平均值与真实值误差 $\rm(mm)$', fontsize=30)
plt.xlim(0, 7000)
plt.ylim(-250, 150)
plt.xticks(np.around(np.linspace(0, 7000, 8), decimals=0), fontproperties='Times New Roman', size=26)
plt.yticks(np.around(np.linspace(-250, 150, 9), decimals=0), fontproperties='Times New Roman', size=26)
plt.scatter(mean_A4_list, err_A4_list, color='g', lw=0.5)
plt.plot(np.linspace(np.min(mean_A4_list), np.max(mean_A4_list),10), err_fit_curve4, 'g-', lw=1)
plt.show()


def modify(dis4):
    """ 根据上面代码获得修正公式系数 """
    dis_mod1 = np.polyval(np.array([2.94138393e-02, -1.45008607e+02]), dis4[0])
    dis_mod2 = np.polyval(np.array([1.98224645e-02, -1.10754487e+02]), dis4[0])
    dis_mod3 = np.polyval(np.array([2.25552478e-02, -1.35738338e+02]), dis4[0])
    dis_mod4 = np.polyval(np.array([2.84936591e-02, -1.61115850e+02]), dis4[0])
    return [dis_mod1, dis_mod2, dis_mod3, dis_mod4]

    global_best_x_list_R, global_best_x_list_W_e1, global_best_x_list_W_e2, global_best_y_list_R,
global_best_y_list_W_e1, global_best_y_list_W_e2 = [], [], [], [], [], []
Z_list, global_best_x_list, global_best_y_list = [], [], []

    for i in range(324):
    filename_R = './输出/第一问 excel/' + str(i+1) + '.正常.xlsx'
    filename_W_e1 = './输出/第一问 excel-异常分两类/' + str(i+1) + '.异常-1.xlsx'
    filename_W_e2 = './输出/第一问 excel-异常分两类/' + str(i+1) + '.异常-2.xlsx'
    filename_T = './Data/附件 2：测试集（实验场景 1）.txt'
    filename_T3 = './Data/附件 3：测试集（实验场景 2）.txt'

    df_R = pd.read_excel(filename_R)
    df_W_e1 = pd.read_excel(filename_W_e1)
    df_W_e2 = pd.read_excel(filename_W_e2)
    df_T = pd.read_table(filename_T, sep=':', names=['x1', 'time', 'x2', 'tagID', 'type', 'length', 'length2',
'Serial No.', 'No.'])
    df_T = df_T.iloc[:, [1, 3, 4, 5, 7, 8]].astype(float)
    df_T3 = pd.read_table(filename_T3, sep=':', names=['x1', 'time', 'x2', 'tagID', 'type', 'length', 'length2',
'Serial No.', 'No.'])
    df_T3 = df_T3.iloc[:, [1, 3, 4, 5, 7, 8]].astype(float)

    # 按照类型分类测试集
    df_T_p = [0, 0, 0, 0]
    df_T3_p = [0, 0, 0, 0]
    for t in range(4):
        df_T_p[t] = df_T.loc[df_T['type'] == t, :]
```

```
        df_T_p[t].index = np.arange(df_T_p[t].shape[0])
        df_T3_p[t] = df_T3.loc[df_T3['type'] == t, :]
        df_T3_p[t].index = np.arange(df_T3_p[t].shape[0])

    #正常平均值
    df_R_mean, df_R_std = [0, 0, 0, 0], [0, 0, 0, 0]
    df_R_mean[0] = df_R.loc[:, 'A0'].mean()
    df_R_mean[1] = df_R.loc[:, 'A1'].mean()
    df_R_mean[2] = df_R.loc[:, 'A2'].mean()
    df_R_mean[3] = df_R.loc[:, 'A3'].mean()

    # 异常平均值，标准差，统计异常维度数目:使用方差排序取较大两个
    df_W_mean_e1, df_W_mean_e2, df_W_std_e1, df_W_std_e2 = [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0,
0, 0]
    df_W_mean_e1[0] = df_W_e1.loc[:, 'A0'].mean()
    df_W_mean_e1[1] = df_W_e1.loc[:, 'A1'].mean()
    df_W_mean_e1[2] = df_W_e1.loc[:, 'A2'].mean()
    df_W_mean_e1[3] = df_W_e1.loc[:, 'A3'].mean()
    df_W_std_e1[0] = df_W_e1.loc[:, 'A0'].std()
    df_W_std_e1[1] = df_W_e1.loc[:, 'A1'].std()
    df_W_std_e1[2] = df_W_e1.loc[:, 'A2'].std()
    df_W_std_e1[3] = df_W_e1.loc[:, 'A3'].std()

    df_W_mean_e2[0] = df_W_e2.loc[:, 'A0'].mean()
    df_W_mean_e2[1] = df_W_e2.loc[:, 'A1'].mean()
    df_W_mean_e2[2] = df_W_e2.loc[:, 'A2'].mean()
    df_W_mean_e2[3] = df_W_e2.loc[:, 'A3'].mean()
    df_W_std_e2[0] = df_W_e2.loc[:, 'A0'].std()
    df_W_std_e2[1] = df_W_e2.loc[:, 'A1'].std()
    df_W_std_e2[2] = df_W_e2.loc[:, 'A2'].std()
    df_W_std_e2[3] = df_W_e2.loc[:, 'A3'].std()

    # GA 遗传算法,4 距离，正常
    for j in range(4):
        A_dis[j] = df_R_mean[j]
        A_dis = A_dis - np.array(modify(A_dis))
    global_best_y = 1e8
    for j in range(10):
        ga = GA(func=optfunc4, n_dim=3, size_pop=100, max_iter=200, prob_mut=0.001, lb=lb, ub=ub,
    precision=1e-7)
        best_x, best_y = ga.run()
        if best_y < global_best_y:
            global_best_x = best_x
            global_best_y = best_y
```

```
        if global_best_y < exit_error:
            break
global_best_x_list_R.append(global_best_x)
global_best_y_list_R.append(global_best_y)
Z_list.append(global_best_x[2])
global_best_x_list.append(global_best_x)
global_best_y_list.append(global_best_y)
print(i, '正常: best_x:', global_best_x, 'best_y:', global_best_y)

"""GA 遗传算法,异常-1"""
limit, exit_error = 1, 1
for j in range(4):
        A_dis[j] = df_W_mean_e1[j]
        #A_dis[j] = fit_coefficient[j, 0] * A_dis[j] + fit_coefficient[j, 1]
A_dis = A_dis - np.array(modify(A_dis))

# 2. 三边初始值计算
res, err = 0, 0
res = triposition(A0, A_dis[0], A1, A_dis[1], A2, A_dis[2], A3, A_dis[3])
err = disErr_scene1(res, A_dis)
if err > limit:
        print(i, '异常-1 初始计算结果: best_x:', res, 'best_y:', err, '四个距离', np.round(A_dis, 2), '大于上限，开始遗传算法求解异常点')
else:
        print(i, '异常-1 初始计算结果: best_x:', res, 'best_y:', err, '四个距离', np.round(A_dis, 2), '正常点')

# 3. 如果异常则遗传优化
final_choose = -1
global_best_x, global_best_y = 0, err
if err > limit:
        for j in range(4):
                minus_object = j    # 修改优化对象
                ga = GA(func=sanbian_gene_opFunc, n_dim=1, size_pop=20, max_iter=10, prob_mut=0.001, lb=0, ub=600, precision=1e-7)
                best_x, best_y = ga.run()

                if best_y < global_best_y:
                        final_choose = j
                        global_best_x = best_x
                        global_best_y = best_y
                if global_best_y < exit_error:
                        break
```

```
# 2. 三边重新计算
if err > limit:
    if final_choose != -1:
        res, err = 0, 0
        A_dis[final_choose] = A_dis[final_choose] - global_best_x
        res = triposition(A0, A_dis[0], A1, A_dis[1], A2, A_dis[2], A3, A_dis[3])
        err = disErr_scene1(res, A_dis)
        print(i, '异常点-1 修正结果:  修正后坐标:', res, 'best_y:', global_best_y, '干扰点',
final_choose, 'best_x  修正值', global_best_x, '四个距离', np.round(A_dis, 2))
        global_best_x_list_W_e1.append(res)
        global_best_y_list_W_e1.append(err)
    else:
        print(i, '异常点-1 修正结果:失败')

# GA  遗传算法,4 距离,  异常-2

for j in range(4):
    A_dis[j] = df_W_mean_e2[j]
    #A_dis[j] = fit_coefficient[j, 0] * A_dis[j] + fit_coefficient[j, 1]
A_dis = A_dis - np.array(modify(A_dis))

# 2. 三边初始值计算
res, err = 0, 0
res = triposition(A0, A_dis[0], A1, A_dis[1], A2, A_dis[2], A3, A_dis[3])
err = disErr_scene1(res, A_dis)
if err > limit:
    print(i, '异常-2 初始计算结果: best_x:', res, 'best_y:', err, '四个距离', np.round(A_dis, 2), '大于
上限，开始遗传算法求解异常点')
else:
    print(i, '异常-2 初始计算结果: best_x:', res, 'best_y:', err, '四个距离', np.round(A_dis, 2), '正常
点')

# 3. 如果异常则遗传优化
final_choose = -1
global_best_x, global_best_y = 0, err
if err > limit:
    for j in range(4):
        minus_object = j   #  修改优化对象

        ga = GA(func=sanbian_gene_opFunc, n_dim=1, size_pop=20, max_iter=10, prob_mut=0.001,
lb=0, ub=600,
                precision=1e-7)
        best_x, best_y = ga.run()
        if best_y < global_best_y:
```

```
                final_choose = j
                global_best_x = best_x
                global_best_y = best_y
            if global_best_y < exit_error:
                break

    # 2. 三边重新计算
    if err > limit:
        if final_choose != -1:
            res, err = 0, 0
            A_dis[final_choose] = A_dis[final_choose] - global_best_x
            res = triposition(A0, A_dis[0], A1, A_dis[1], A2, A_dis[2], A3, A_dis[3])
            err = disErr_scene1(res, A_dis)
            print(i, '异常点-2 修正结果:  修正后坐标:', res, 'best_y:', global_best_y, '干扰点',
    final_choose, 'best_x  修正值',
                  global_best_x, '四个距离', np.round(A_dis, 2))
            global_best_x_list_W_e2.append(res)
            global_best_y_list_W_e2.append(err)
        else:
            print(i, '异常点-2 修正结果:失败')


    # 输出误差Excel
global_best_x_list_array = np.array(global_best_x_list)
global_best_y_list_array = np.array(global_best_y_list).flatten()
df_outerr = {'x':global_best_x_list_array[:, 0], 'y':global_best_x_list_array[:, 1], 'z':global_best_x_list_array[:,
2], 'err_gen':global_best_y_list_array}
df_outerr = pd.DataFrame(df_outerr)
df_outerr.to_excel('./输出/2-输出 xyz 坐标和 Z 误差-遗传算法.xlsx')

# 输出分类Excel
global_best_y_list_R = np.array(global_best_y_list_R).flatten()
global_best_y_list_W_e1 = np.array(global_best_y_list_W_e1).flatten()
global_best_y_list_W_e2 = np.array(global_best_y_list_W_e2).flatten()
df_out_data = {'R':global_best_y_list_R, 'W1':global_best_y_list_W_e1, 'W2':global_best_y_list_W_e2}
df_out = pd.DataFrame(df_out_data)
df_out.to_excel('./输出/第二问遗传算法分类.xlsx')
df_out.to_excel('./输
    # 输出误差Excel
global_best_x_list_We1_array = np.array(global_best_x_list_W_e1)
global_best_y_list_We1_array = np.array(global_best_y_list_W_e1).flatten()
df_outerr1 = {'x':global_best_x_list_We1_array[:, 0], 'y':global_best_x_list_We1_array[:, 1],
'z':global_best_x_list_We1_array[:, 2], 'err_gen':global_best_y_list_We1_array}
df_outerr1 = pd.DataFrame(df_outerr1)
```

```
df_outerr1.to_excel('./输出/2-输出 xyz 坐标和 Z 误差(异常 1)-.xlsx')

global_best_x_list_We2_array = np.array(global_best_x_list_W_e2)
global_best_y_list_We2_array = np.array(global_best_y_list_W_e2).flatten()
df_outerr2 = {'x':global_best_x_list_We2_array[:, 0], 'y':global_best_x_list_We2_array[:, 1],
'z':global_best_x_list_We2_array[:, 2], 'err_gen':global_best_y_list_We2_array}
df_outerr2 = pd.DataFrame(df_outerr2)
df_outerr2.to_excel('./输出/2-输出 xyz 坐标和 Z 误差(异常 2)-.xlsx')
```

Matlab 神经网络代码

```
clear all
clc
data = xlsread('C:\Users\14788\Desktop\Math-Python\输出\2-输出 xyz 坐标和 Z 误差-筛选.xlsx');
data_ceshiji2 = xlsread('C:\Users\14788\Desktop\Math-Python\输出\2-测试集.xlsx');
data_ceshiji2_in = data_ceshiji2(:,1:3);
data_ceshiji2_in = data_ceshiji2_in';

data_yichang2_2 = xlsread('C:\Users\14788\Desktop\Math-Python\输出\2-输出 xyz 坐标和 Z 误差(异常 2).xlsx');
data_yichang2_2_in = data_yichang2_2(:,1:3);
data_yichang2_2_in = data_yichang2_2_in';


train_num = floor(0.9 * size(data,1));
randnum=randperm(size(data,1));

data_train=data(randnum(1:train_num),:);
data_test=data(randnum(train_num+1:end),:);

data_in = data_train(:, 2:4)';
data_out = data_train(:, 6)';
data_intest = data_test(:, 2:4)';
data_outtest = data_test(:, 6)';

inputnum=2;
hiddennum=8;
outputnum=1;
net=newff(data_in,data_out,hiddennum,{'tansig','purelin'},'trainlm');

W1= net. iw{1, 1};
B1 = net.b{1};
```

```
W2 = net.lw{2,1};
B2 = net. b{2};

net.trainParam.epochs=1000;
net.trainParam.lr=0.01;
net.trainParam.goal=0.000000001;
net=train(net,data_in,data_out);
an=sim(net,data_intest); error=an-data_outtest;
error = error';
error_mean = mean(abs(error));
error_std = std(abs(error));
disp(['error_mean = ',num2str(error_mean)]);
disp(['error_std = ',num2str(error_std)]);

an_ceshiji2=sim(net,data_ceshiji2_in);

an_yichang2_2=sim(net,data_yichang2_2_in);
```

3. 任务三源码

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy import spatial, integrate, interpolate, optimize
import sys
import sympy
sys.path.append("Lib/scikit")
from sko.GA import GA   # 项目内置库
from sko.SA import SA   # 项目内置库
from sko.PSO import PSO   # 项目内置库
import Lib.myFunctions as my   # 项目内置库

# 计算定位坐标
def triposition(A0, da, A1, db, A2, dc, A4, dd):
    xa, ya, za = A0[0], A0[1], A0[2]
    xb, yb, zb = A1[0], A1[1], A1[2]
    xc, yc, zc = A2[0], A2[1], A2[2]
    xd, yd, zd = A3[0], A3[1], A3[2]

    x, y, z = sympy.symbols("x, y, z")
    f1 = 2 * x * (xa - xd) + np.square(xd) - np.square(xa) + 2 * y * (ya - yd) + np.square(yd) - np.square(ya) +
```

```
2 * z * (za - zd) + np.square(zd) - np.square(za) - (np.square(dd) - np.square(da))
    f2 = 2 * x * (xb - xd) + np.square(xd) - np.square(xb) + 2 * y * (yb - yd) + np.square(yd) - np.square(yb)
+ 2 * z * (zb - zd) + np.square(zd) - np.square(zb) - (np.square(dd) - np.square(db))
    f3 = 2 * x * (xc - xd) + np.square(xd) - np.square(xc) + 2 * y * (yc - yd) + np.square(yd) - np.square(yc) +
2 * z * (zc - zd) + np.square(zd) - np.square(zc) - (np.square(dd) - np.square(dc))
    result = sympy.solve([f1, f2, f3], [x, y, z])

    locx, locy, locz = result[x], result[y], result[z]
    return [locx, locy, locz]

def disErr_scene1(tag_loc, distance4):
    tag_loc = np.array(tag_loc).astype(float)
    A0_ = np.array(A0)
    A1_ = np.array(A1)
    A2_ = np.array(A2)
    A3_ = np.array(A3)
    dis0 = np.linalg.norm(tag_loc - A0_)
    dis1 = np.linalg.norm(tag_loc - A1_)
    dis2 = np.linalg.norm(tag_loc - A2_)
    dis3 = np.linalg.norm(tag_loc - A3_)

    err = np.fabs(dis0 - distance4[0]) + np.fabs(dis1 - distance4[1]) + np.fabs(dis2 - distance4[2]) +
np.fabs(dis3 - distance4[3])
    return err

def sanbian_gene_opFunc(x):
    A_m = A_dis.copy()
    A_m[minus_object] = A_m[minus_object] - x
    res1 = triposition(A0, A_m[0], A1, A_m[1], A2, A_m[2], A3, A_m[3])
    err1 = disErr_scene1(res1, A_m)
    return err1

def distance4(tag_loc):
    tag_loc = np.array(tag_loc)
    A0 = np.array([0, 0, 1200])
    A1 = np.array([5000, 0, 1600])
    A2 = np.array([0, 3000, 1600])
    A3 = np.array([5000, 3000, 1200])
    return [np.linalg.norm(tag_loc - A0), np.linalg.norm(tag_loc - A1), np.linalg.norm(tag_loc - A2),
np.linalg.norm(tag_loc - A3)]

df_tag = pd.read_excel('./Data/Tag 坐标信息.xlsx')
df_tag_xyz = np.array([df_tag['x'], df_tag['y'], df_tag['z']])
    """ 第三问 """
```

```
A0 = np.array([0, 0, 1200])
A1 = np.array([5000, 0, 1600])
A2 = np.array([0, 3000, 1600])
A3 = np.array([5000, 3000, 1200])

A_dis = [0, 0, 0, 0]
i1, i2, i3 = 0, 0, 0
real_dis_list, A1234_list = [], []
minus_object = 0

def optfunc4(x):
    dis4 = np.array(distance4(x))
    dis4 = np.fabs(dis4 - np.array([A_dis[0], A_dis[1], A_dis[2], A_dis[3]]))
    return dis4[0] + dis4[1] + dis4[2] + dis4[3]
def optfunc4_print(x):
    dis4 = np.array(distance4(x))
    dis4 = np.fabs(dis4 - np.array([A_dis[0], A_dis[1], A_dis[2], A_dis[3]]))
    dis4 = np.round(dis4, 2)
    print(dis4[0], dis4[1], dis4[2], dis4[3])
def optfunc3(x):
    dis3 = np.array(distance3(x, i1, i2, i3))
    dis3 = np.fabs(dis3 - np.array([A_dis[i1], A_dis[i2], A_dis[i3]]))
    return dis3[0] + dis3[1] + dis3[2]

""" 遗传算法 """
def modify(dis4):
    # 使用误差-距离 一次函数拟合公式修正
    dis_mod1 = np.polyval(np.array([2.94138393e-02, -1.45008607e+02]), dis4[0])
    dis_mod2 = np.polyval(np.array([1.98224645e-02, -1.10754487e+02]), dis4[0])
    dis_mod3 = np.polyval(np.array([2.25552478e-02, -1.35738338e+02]), dis4[0])
    dis_mod4 = np.polyval(np.array([2.84936591e-02, -1.61115850e+02]), dis4[0])
    return [dis_mod1, dis_mod2, dis_mod3, dis_mod4]

fit_coefficient = np.ones((4, 2))
fit_coefficient2 = [1.00497993, 1.00649343, 1.00977879, 1.00988594]
fit_coefficient = np.array([[0.97058616, 145.00860654], [0.98017754, 110.75448702], [0.97744475,
135.73833774], [0.97150634, 161.11584952]])
global_best_x_list_R, global_best_x_list_W_e1, global_best_x_list_W_e2, global_best_y_list_R,
global_best_y_list_W_e1, global_best_y_list_W_e2 = [], [], [], [], [], []
Rmean1, Rmean2, Rmean3, Rmean4, W1mean1, W1mean2, W1mean3, W1mean4, W2mean1, W2mean2,
W2mean3, W2mean4 = [], [], [], [], [], [], [], [], [], [], [], []
#range(324)
```

```
filename_T = './Data/附件 2：测试集（实验场景 1）.txt'
filename_T3 = './Data/附件 3：测试集（实验场景 2）.txt'

df_T = pd.read_table(filename_T, sep=':', names=['x1', 'time', 'x2', 'tagID', 'type', 'length', 'length2',
'Serial No.', 'No.'])
df_T = df_T.iloc[:, [1, 3, 4, 5, 7, 8]].astype(float)
df_T3 = pd.read_table(filename_T3, sep=':', names=['x1', 'time', 'x2', 'tagID', 'type', 'length', 'length2',
'Serial No.', 'No.'])
df_T3 = df_T3.iloc[:, [1, 3, 4, 5, 7, 8]].astype(float)

#  按照类型分类测试集
df_T_p = [0, 0, 0, 0]
df_T3_p = [0, 0, 0, 0]
for t in range(4):
    df_T_p[t] = df_T.loc[df_T['type'] == t, :]
    df_T_p[t].index = np.arange(df_T_p[t].shape[0])
    df_T3_p[t] = df_T3.loc[df_T3['type'] == t, :]
    df_T3_p[t].index = np.arange(df_T3_p[t].shape[0])

global_best_x_list, global_best_y_list = [], []
limit, exit_error = 1, 1

for k in np.arange(5):
    # 1. 先修正
    for j in range(4):
        A_dis[j] = df_T3_p[j].loc[k, 'length']
        # A_dis[j] = fit_coefficient[j, 0] * A_dis[j] + fit_coefficient[j, 1]
    A_dis = A_dis - np.array(modify(A_dis))

    global_best_y = 1e8
    for j in range(10):
        ga = GA(func=optfunc4, n_dim=3, size_pop=100, max_iter=200, prob_mut=0.001, lb=[0, 0, 0],
                ub=[5000, 3000, 3000], precision=1e-7)
        best_x, best_y = ga.run()
        # pso = PSO(func=optfunc4, n_dim=3, pop=pop, max_iter=max_iter, lb=lb, ub=ub, w=w, c1=c1,
c2=c2)
        # pso.run()
        # best_x, best_y = pso.gbest_x, pso.gbest_y

        if best_y < global_best_y:
            global_best_x = best_x
            global_best_y = best_y
        if global_best_y < exit_error:
            break
```

```
        global_best_x_list.append(global_best_x)
        global_best_y_list.append(global_best_y)
        print(k, '正常: best_x:', global_best_x, 'best_y:', global_best_y)

for k in np.arange(5, 10):
        # 1. 先修正
        for j in range(4):
                A_dis[j] = df_T3_p[j].loc[k, 'length']
                # A_dis[j] = fit_coefficient[j, 0] * A_dis[j] + fit_coefficient[j, 1]
        A_dis = A_dis - np.array(modify(A_dis))

        # 2. 三边初始值计算
        res, err = 0, 0
        res = triposition(A0, A_dis[0], A1, A_dis[1], A2, A_dis[2], A3, A_dis[3])
        err = disErr_scene1(res, A_dis)
        if err > limit:
                print(k, '测试集初始计算结果结果: best_x:', res, 'best_y:', err, '四个距离', np.round(A_dis, 2), '
大于上限，开始遗传算法求解异常点')
        else:
                print(k, '测试集初始计算结果结果: best_x:', res, 'best_y:', err, '四个距离', np.round(A_dis, 2), '
正常点')

        # 3. 如果异常则遗传优化
        final_choose = -1
        global_best_x, global_best_y = 0, err
        if err > limit:
                for j in range(4):
                        minus_object = j    # 修改优化对象

                        ga = GA(func=sanbian_gene_opFunc, n_dim=1, size_pop=20, max_iter=10, prob_mut=0.001,
lb=0, ub=600,
                                        precision=1e-7)
                        best_x, best_y = ga.run()

                        if best_y < global_best_y:
                                final_choose = j
                                global_best_x = best_x
                                global_best_y = best_y
                        if global_best_y < exit_error:
                                break
        # 2. 三边重新计算
        if err > limit:
                if final_choose != -1:
                        res, err = 0, 0
```

```
                A_dis[final_choose] = A_dis[final_choose] - global_best_x
                res = triposition(A0, A_dis[0], A1, A_dis[1], A2, A_dis[2], A3, A_dis[3])
                err = disErr_scene1(res, A_dis)
                print(k, '测试集修正结果:  修正后坐标:', res, 'best_y:', global_best_y, '干扰点',
final_choose, 'best_x  修正值',
                      global_best_x, '四个距离', np.round(A_dis, 2))
                global_best_x_list.append(res)
                global_best_y_list.append(err)
            else:
                print(k, '测试集修正结果:失败')

# 输出误差Excel
global_best_x_list = np.array(global_best_x_list)
global_best_y_list = np.array(global_best_y_list).flatten()
df_outerr = {'x': global_best_x_list[:, 0], 'y': global_best_x_list[:, 1], 'z': global_best_x_list[:, 2],
             'err_gen': global_best_y_list}
df_outerr = pd.DataFrame(df_outerr)
df_outerr.to_excel('./输出/3-测试集.xlsx')
```

4. 任务四源码

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy import spatial, integrate, interpolate, optimize
import sys
import sympy

sys.path.append("Lib/scikit")
from sko.GA import GA   # 项目内置库
from sko.SA import SA   # 项目内置库
from sko.PSO import PSO   # 项目内置库
import Lib.myFunctions as my   # 项目内置库

"""
场景一
    A0 = np.array([0, 0, 1300])
    A1 = np.array([5000, 0, 1700])
    A2 = np.array([0, 5000, 1700])
    A3 = np.array([5000, 5000, 1300])
场景二
    A0 = np.array([0, 0, 1200])
    A1 = np.array([5000, 0, 1600])
```

```
        A2 = np.array([0, 3000, 1600])
        A3 = np.array([5000, 3000, 1200])
"""

A_dis = [0, 0, 0, 0]
i1, i2, i3 = 0, 0, 0
real_dis_list, A1234_list = [], []
A0 = np.array([0, 0, 1300])
A1 = np.array([5000, 0, 1700])
A2 = np.array([0, 5000, 1700])
A3 = np.array([5000, 5000, 1300])

# 计算定位坐标
def triposition(A0, da, A1, db, A2, dc, A4, dd):
        xa, ya, za = A0[0], A0[1], A0[2]
        xb, yb, zb = A1[0], A1[1], A1[2]
        xc, yc, zc = A2[0], A2[1], A2[2]
        xd, yd, zd = A3[0], A3[1], A3[2]

        x, y, z = sympy.symbols("x, y, z")
        f1 = 2 * x * (xa - xd) + np.square(xd) - np.square(xa) + 2 * y * (ya - yd) + np.square(yd) - np.square(ya) +
2 * z * (za - zd) + np.square(zd) - np.square(za) - (np.square(dd) - np.square(da))
        f2 = 2 * x * (xb - xd) + np.square(xd) - np.square(xb) + 2 * y * (yb - yd) + np.square(yd) - np.square(yb)
+ 2 * z * (zb - zd) + np.square(zd) - np.square(zb) - (np.square(dd) - np.square(db))
        f3 = 2 * x * (xc - xd) + np.square(xd) - np.square(xc) + 2 * y * (yc - yd) + np.square(yd) - np.square(yc) +
2 * z * (zc - zd) + np.square(zd) - np.square(zc) - (np.square(dd) - np.square(dc))
        result = sympy.solve([f1, f2, f3], [x, y, z])

        locx, locy, locz = result[x], result[y], result[z]
        return [locx, locy, locz]

def disErr_scene1(tag_loc, distance4):
        tag_loc = np.array(tag_loc).astype(float)
        A0_ = np.array(A0)
        A1_ = np.array(A1)
        A2_ = np.array(A2)
        A3_ = np.array(A3)
        dis0 = np.linalg.norm(tag_loc - A0_)
        dis1 = np.linalg.norm(tag_loc - A1_)
        dis2 = np.linalg.norm(tag_loc - A2_)
        dis3 = np.linalg.norm(tag_loc - A3_)

        err = np.fabs(dis0 - distance4[0]) + np.fabs(dis1 - distance4[1]) + np.fabs(dis2 - distance4[2]) +
np.fabs(dis3 - distance4[3])
```

```
    return err

def distance4(tag_loc):
    tag_loc = np.array(tag_loc)
    A0 = np.array([0, 0, 1300])
    A1 = np.array([5000, 0, 1700])
    A2 = np.array([0, 5000, 1700])
    A3 = np.array([5000, 5000, 1300])
    return [np.linalg.norm(tag_loc - A0), np.linalg.norm(tag_loc - A1), np.linalg.norm(tag_loc - A2),
np.linalg.norm(tag_loc - A3)]

def distance3(tag_loc, i1, i2, i3):
    i1, i2, i3 = int(i1), int(i2), int(i3)
    tag_loc = np.array(tag_loc)
    A0 = np.array([0, 0, 1300])
    A1 = np.array([5000, 0, 1700])
    A2 = np.array([0, 5000, 1700])
    A3 = np.array([5000, 5000, 1300])
    A_all = np.array([A0, A1, A2, A3])
    return [np.linalg.norm(tag_loc - A_all[i1, :]), np.linalg.norm(tag_loc - A_all[i2, :]),
np.linalg.norm(tag_loc - A_all[i3, :])]

df_tag = pd.read_excel('./Data/Tag 坐标信息.xlsx')
df_tag_xyz = np.array([df_tag['x'], df_tag['y'], df_tag['z']])

""" 第二问 """
def optfunc4(x):
    dis4 = np.array(distance4(x))
    dis4 = np.fabs(dis4 - np.array([A_dis[0], A_dis[1], A_dis[2], A_dis[3]]))
    return dis4[0] + dis4[1] + dis4[2] + dis4[3]
def optfunc4_print(x):
    dis4 = np.array(distance4(x))
    dis4 = np.fabs(dis4 - np.array([A_dis[0], A_dis[1], A_dis[2], A_dis[3]]))
    dis4 = np.round(dis4, 2)
    print(dis4[0], dis4[1], dis4[2], dis4[3])
def optfunc3(x):
    dis3 = np.array(distance3(x, i1, i2, i3))
    dis3 = np.fabs(dis3 - np.array([A_dis[i1], A_dis[i2], A_dis[i3]]))
    return dis3[0] + dis3[1] + dis3[2]

# 校正 4 个距离
err_A1_list, err_A2_list, err_A3_list, err_A4_list = [], [], [], []
mean_A1_list, mean_A2_list, mean_A3_list, mean_A4_list = [], [], [], []
```

```
""" 遗传算法 """
def modify(dis4):
    # 使用误差-距离 一次函数拟合公式修正
    dis_mod1 = np.polyval(np.array([2.94138393e-02, -1.45008607e+02]), dis4[0])
    dis_mod2 = np.polyval(np.array([1.98224645e-02, -1.10754487e+02]), dis4[0])
    dis_mod3 = np.polyval(np.array([2.25552478e-02, -1.35738338e+02]), dis4[0])
    dis_mod4 = np.polyval(np.array([2.84936591e-02, -1.61115850e+02]), dis4[0])
    return [dis_mod1, dis_mod2, dis_mod3, dis_mod4]

fit_coefficient = np.ones((4, 2))
fit_coefficient = np.array([[0.97058616, 145.00860654], [0.98017754, 110.75448702], [0.97744475,
135.73833774], [0.97150634, 161.11584952]])
global_best_x_list_R, global_best_x_list_W_e1, global_best_x_list_W_e2, global_best_y_list_R,
global_best_y_list_W_e1, global_best_y_list_W_e2 = [], [], [], [], [], []
Rmean1, Rmean2, Rmean3, Rmean4, W1mean1, W1mean2, W1mean3, W1mean4, W2mean1, W2mean2,
W2mean3, W2mean4 = [], [], [], [], [], [], [], [], [], [], [], []

"""  三个测试集验证  """
filename_T = './Data/附件 2：测试集（实验场景 1）.txt'
filename_T3 = './Data/附件 3：测试集（实验场景 2）.txt'
filename_T4 = './Data/附件 4：测试集（实验场景 1）.txt'

df_T = pd.read_table(filename_T, sep=':',
                     names=['x1', 'time', 'x2', 'tagID', 'type', 'length', 'length2', 'Serial No.', 'No.'])
df_T = df_T.iloc[:, [1, 3, 4, 5, 7, 8]].astype(float)
df_T3 = pd.read_table(filename_T3, sep=':',
                      names=['x1', 'time', 'x2', 'tagID', 'type', 'length', 'length2', 'Serial No.', 'No.'])
df_T3 = df_T3.iloc[:, [1, 3, 4, 5, 7, 8]].astype(float)
df_T4 = pd.read_table(filename_T4, sep=':',
                      names=['x1', 'time', 'x2', 'tagID', 'type', 'length', 'length2', 'Serial No.', 'No.'])
df_T4 = df_T4.iloc[:, [1, 3, 4, 5, 7, 8]].astype(float)

# 按照类型分类测试集
df_T_p = [0, 0, 0, 0]
df_T3_p = [0, 0, 0, 0]
df_T4_p = [0, 0, 0, 0]
for t in range(4):
    df_T_p[t] = df_T.loc[df_T['type'] == t, :]
    df_T_p[t].index = np.arange(df_T_p[t].shape[0])
    df_T3_p[t] = df_T3.loc[df_T3['type'] == t, :]
    df_T3_p[t].index = np.arange(df_T3_p[t].shape[0])
    df_T4_p[t] = df_T4.loc[df_T4['type'] == t, :]
    df_T4_p[t].index = np.arange(df_T4_p[t].shape[0])
```

```
# 三边判据

limit = 87.6

for i in range(10):
    for j in range(4):
        A_dis[j] = df_T_p[j].loc[i, 'length']
        #A_dis[j] = fit_coefficient[j, 0] * A_dis[j] + fit_coefficient[j, 1]
    A_dis = A_dis - np.array(modify(A_dis))

    # 2. 三边初始值计算
    res, err = 0, 0
    res = triposition(A0, A_dis[0], A1, A_dis[1], A2, A_dis[2], A3, A_dis[3])
    err = disErr_scene1(res, A_dis)
    if err > limit:
        print(i, '测试集 2 计算结果: best_x:', res, 'best_y:', err, '四个距离', np.round(A_dis, 2), '大于上限，干扰点')
    else:
        print(i, '测试集 2 计算结果: best_x:', res, 'best_y:', err, '四个距离', np.round(A_dis, 2), '小于上限，非干扰点')

for i in range(10):
    A0 = np.array([0, 0, 1200])
    A1 = np.array([5000, 0, 1600])
    A2 = np.array([0, 3000, 1600])
    A3 = np.array([5000, 3000, 1200])
    for j in range(4):
        A_dis[j] = df_T3_p[j].loc[i, 'length']
        #A_dis[j] = fit_coefficient[j, 0] * A_dis[j] + fit_coefficient[j, 1]
    A_dis = A_dis - np.array(modify(A_dis))

    # 2. 三边初始值计算
    res, err = 0, 0
    res = triposition(A0, A_dis[0], A1, A_dis[1], A2, A_dis[2], A3, A_dis[3])
    err = disErr_scene1(res, A_dis)
    if err > limit:
        print(i, '测试集 3 计算结果: best_x:', res, 'best_y:', err, '四个距离', np.round(A_dis, 2), '大于上限，干扰点')
    else:
        print(i, '测试集 3 计算结果: best_x:', res, 'best_y:', err, '四个距离', np.round(A_dis, 2), '小于上限，非干扰点')

for i in range(10):
    A0 = np.array([0, 0, 1300])
```

```
    A1 = np.array([5000, 0, 1700])
    A2 = np.array([0, 5000, 1700])
    A3 = np.array([5000, 5000, 1300])

    for j in range(4):
        A_dis[j] = df_T4_p[j].loc[i, 'length']
        #A_dis[j] = fit_coefficient[j, 0] * A_dis[j] + fit_coefficient[j, 1]
    A_dis = A_dis - np.array(modify(A_dis))

    # 2. 三边初始值计算
    res, err = 0, 0
    res = triposition(A0, A_dis[0], A1, A_dis[1], A2, A_dis[2], A3, A_dis[3])
    err = disErr_scene1(res, A_dis)
    if err > limit:
        print(i, '测试集 4 计算结果: best_x:', res, 'best_y:', err, '四个距离', np.round(A_dis, 2), '大于上限，干扰点')
    else:
        print(i, '测试集 4 计算结果: best_x:', res, 'best_y:', err, '四个距离', np.round(A_dis, 2), '小于上限，非干扰点')
```

5. 任务五源码

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy import spatial, integrate, interpolate, optimize
import sys
import sympy
import math
sys.path.append("Lib/scikit")
from sko.GA import GA   # 项目内置库
from sko.SA import SA   # 项目内置库
from sko.PSO import PSO   # 项目内置库
import Lib.myFunctions as my   # 项目内置库

A0 = np.array([0, 0, 1300])
A1 = np.array([5000, 0, 1700])
A2 = np.array([0, 5000, 1700])
A3 = np.array([5000, 5000, 1300])

A_dis = [0, 0, 0, 0]
i1, i2, i3 = 0, 0, 0
real_dis_list, A1234_list = [], []
minus_object = 0
```

```
def distance4(tag_loc):
    tag_loc = np.array(tag_loc)
    A0 = np.array([0, 0, 1300])
    A1 = np.array([5000, 0, 1700])
    A2 = np.array([0, 5000, 1700])
    A3 = np.array([5000, 5000, 1300])
    return [np.linalg.norm(tag_loc - A0), np.linalg.norm(tag_loc - A1), np.linalg.norm(tag_loc - A2),
np.linalg.norm(tag_loc - A3)]

# 计算定位坐标
def triposition(A0, da, A1, db, A2, dc, A4, dd):
    xa, ya, za = A0[0], A0[1], A0[2]
    xb, yb, zb = A1[0], A1[1], A1[2]
    xc, yc, zc = A2[0], A2[1], A2[2]
    xd, yd, zd = A3[0], A3[1], A3[2]

    x, y, z = sympy.symbols("x, y, z")
    f1 = 2 * x * (xa - xd) + np.square(xd) - np.square(xa) + 2 * y * (ya - yd) + np.square(yd) - np.square(ya) +
2 * z * (za - zd) + np.square(zd) - np.square(za) - (np.square(dd) - np.square(da))
    f2 = 2 * x * (xb - xd) + np.square(xd) - np.square(xb) + 2 * y * (yb - yd) + np.square(yd) - np.square(yb)
+ 2 * z * (zb - zd) + np.square(zd) - np.square(zb) - (np.square(dd) - np.square(db))
    f3 = 2 * x * (xc - xd) + np.square(xd) - np.square(xc) + 2 * y * (yc - yd) + np.square(yd) - np.square(yc) +
2 * z * (zc - zd) + np.square(zd) - np.square(zc) - (np.square(dd) - np.square(dc))
    result = sympy.solve([f1, f2, f3], [x, y, z])

    locx, locy, locz = result[x], result[y], result[z]
    return [locx, locy, locz]

def disErr_scene1(tag_loc, distance4):
    tag_loc = np.array(tag_loc).astype(float)
    A0_ = np.array(A0)
    A1_ = np.array(A1)
    A2_ = np.array(A2)
    A3_ = np.array(A3)
    dis0 = np.linalg.norm(tag_loc - A0_)
    dis1 = np.linalg.norm(tag_loc - A1_)
    dis2 = np.linalg.norm(tag_loc - A2_)
    dis3 = np.linalg.norm(tag_loc - A3_)

    err = np.fabs(dis0 - distance4[0]) + np.fabs(dis1 - distance4[1]) + np.fabs(dis2 - distance4[2]) +
np.fabs(dis3 - distance4[3])
    return err

def sanbian_gene_opFunc(x):
```

```
    A_m = A_dis.copy()
    A_m[minus_object] = A_m[minus_object] - x
    res1 = triposition(A0, A_m[0], A1, A_m[1], A2, A_m[2], A3, A_m[3])
    err1 = disErr_scene1(res1, A_m)
    return err1

df_tag = pd.read_excel('./Data/Tag 坐标信息.xlsx')
df_tag_xyz = np.array([df_tag['x'], df_tag['y'], df_tag['z']])
    """ 第五问 """
    def optfunc4(x):
    dis4 = np.array(distance4(x))
    dis4 = np.fabs(dis4 - np.array([A_dis[0], A_dis[1], A_dis[2], A_dis[3]]))
    return dis4[0] + dis4[1] + dis4[2] + dis4[3]

def modify(dis4):
    # 使用误差-距离 一次函数拟合公式修正
    dis_mod1 = np.polyval(np.array([2.94138393e-02, -1.45008607e+02]), dis4[0])
    dis_mod2 = np.polyval(np.array([1.98224645e-02, -1.10754487e+02]), dis4[0])
    dis_mod3 = np.polyval(np.array([2.25552478e-02, -1.35738338e+02]), dis4[0])
    dis_mod4 = np.polyval(np.array([2.84936591e-02, -1.61115850e+02]), dis4[0])
    return [dis_mod1, dis_mod2, dis_mod3, dis_mod4]

fit_coefficient = np.ones((4, 2))
fit_coefficient = np.array([[0.97058616, 145.00860654], [0.98017754, 110.75448702], [0.97744475,
135.73833774], [0.97150634, 161.11584952]])
global_best_x_list_R, global_best_x_list_W_e1, global_best_x_list_W_e2, global_best_y_list_R,
global_best_y_list_W_e1, global_best_y_list_W_e2 = [], [], [], [], [], []
global_best_x_list, global_best_y_list = [], []
location_list = []
""" 三边算法测试集 """
filename_T5 = './Data/附件 5：动态轨迹数据.txt'
df_T5 = pd.read_table(filename_T5, sep=':', names=['x1', 'time', 'x2', 'tagID', 'type', 'length', 'length2',
'Serial No.', 'No.'])
df_T5 = df_T5.iloc[:, [1, 3, 4, 5, 7, 8]].astype(float)

# 按照类型分类测试集
df_T5_p = [0, 0, 0, 0]
for t in range(4):
    df_T5_p[t] = df_T5.loc[df_T5['type'] == t, :]
    df_T5_p[t].index = np.arange(df_T5_p[t].shape[0])

# GA 遗传算法, 测试集合（修改 上面4 点坐标、范围、A_dis[j] = df_T3_p[j].loc[k, 'length']）
limit, exit_error = 100, 1
```

```
for k in range(df_T5_p[0].shape[0]):
    # 1. 先修正
    for j in range(4):
        A_dis[j] = df_T5_p[j].loc[k, 'length']
        # A_dis[j] = fit_coefficient[j, 0] * A_dis[j] + fit_coefficient[j, 1]
    A_dis = A_dis - np.array(modify(A_dis))

    # 2. 三边初始值计算
    res, err = 0, 0
    res = triposition(A0, A_dis[0], A1, A_dis[1], A2, A_dis[2], A3, A_dis[3])
    err = disErr_scene1(res, A_dis)
    if err > limit:
        print(k, '测试集初始计算结果结果: best_x:', res, 'best_y:', err, '四个距离', np.round(A_dis, 2), '
异常点')
    else:
        print(k, '测试集初始计算结果结果: best_x:', res, 'best_y:', err, '四个距离', np.round(A_dis, 2), '
正常点')

    #正常数据遗传算法
    if err <= limit:
        pop = 50; max_iter = 200; lb = [0, 0, 0]; w = 0.8; c1 = 0.5; c2 = 0.5;ub = [5000, 5000, 3000]
        global_best_y = 1e8
        for j in range(10):

            ga = GA(func=optfunc4, n_dim=3, size_pop=100, max_iter=200, prob_mut=0.001, lb=[0, 0, 0],
                    ub=[5000, 3000, 3000], precision=1e-7)
            best_x, best_y = ga.run()

            if best_y < global_best_y:
                global_best_x = best_x
                global_best_y = best_y
            if global_best_y < exit_error:
                break
        #global_best_x_list.append(global_best_x)
        global_best_y_list.append(global_best_y)
        location_list.append(global_best_x)
        print(k, '轨迹点正常结果: best_x:', global_best_x, 'best_y:', global_best_y, '四个距离',
np.round(A_dis, 2))

    # 3. 如果异常则遗传优化
    final_choose = -1
    global_best_x, global_best_y = 0, err
    if err > limit:
        for j in range(4):
```

```
                minus_object = j # 修改优化对象


                ga = GA(func=sanbian_gene_opFunc, n_dim=1, size_pop=20, max_iter=10, prob_mut=0.001,
lb=0, ub=600,
            precision=1e-7)
            best_x, best_y = ga.run()

            global_best_x_list.append(global_best_x)
            global_best_y_list.append(global_best_y)

        # 2. 三边重新计算
        if err > limit:
            if final_choose != -1:
                res, err = 0, 0
                A_dis[final_choose] = A_dis[final_choose] - global_best_x
                res = triposition(A0, A_dis[0], A1, A_dis[1], A2, A_dis[2], A3, A_dis[3])
                err = disErr_scene1(res, A_dis)
                location_list.append(res)
                print(k, '轨迹点修正结果:  修正后坐标:', res, 'best_y:', global_best_y, '干扰点',
final_choose, 'best_x  修正值', global_best_x, '四个距离', np.round(A_dis, 2))
            else:
                print(k, '轨迹点修正结果:失败')

# 输出 Excel
location_list = np.array(location_list)
global_best_y_list = np.array(global_best_y_list)

df_out_data = {'x':location_list[:, 0], 'y':location_list[:, 1], 'z':location_list[:, 2], 'err':global_best_y_list}
df_out = pd.DataFrame(df_out_data)
df_out.to_excel('./输出/第五问轨迹点坐标——最终遗传算法.xlsx')
```