# 深圳工作汇报

柴老师，我与张家铭于五月26日抵达深圳朋瑞翔友公司。这段时间主要给该公司完成了以下工作：

1.基于Levenberg–Marquardt法开发了新的TDOA定位算法。包含Python仿真与C++实现。

2.基于LLSE法开发了Tof模式定位算法。包含Python仿真和C++实现。

3.基于Levenberg–Marquardt法开发了两基站的最小平面定位功能。包含Python仿真和C++实现。

4.基于Savitzky-Golay法编写了目标运动轨迹的滤波算法。包含Python仿真和C++实现。

5.建立了定位精度的评价指标，并编写了处理定位数据，生成可视化图表的Python代码。

6.基于基站的双向测距，编写了基站的位置自标定算法。包含Python代码。

以下是各项工作的细则：

## TDOA定位算法开发

由TDOA测距值解算坐标，实质是解超定的非线性方程组。之前让本科毕设的学妹对比了几种常见的非迭代式计算TDOA问题的算法（大体上都是通过问题转化和构建损失函数，将非线性问题近似线性化，通过有限次的最小二乘得到封闭解），没有选到满意的，于是决定不对问题进行近似线性化，直接使用比较常规的非线性优化方法，对方程组迭代求解，实现计算误差的最小化。

在论文《Positioning Algorithms For Cellular Networks Using TDOA》与论文《UWB Localization System for Indoor Applications: Concept, Realization and Analysis》中，作者结合运算速度、精准度、和数值稳定性等多项指标，在对比了多种算法后，认为Levenberg–Marquardt是最适合求解TDOA问题的算法。于是我进行了该算法的Python仿真和C++实现。

在Python仿真中，复现了原厂使用的SX法，与Levenberg–Marquardt法进行对比。在对生成的TDOA数据加上非视距的影响和噪声时，虽然LM法得到的是理论上的误差最小二乘解，在定位精度上比原厂SX算法也只显示了极微弱的优势。**我之前以为改善坐标的解算就能显著提升定位精度，现在看来我对这个问题的认识是有失误的。**我可能高估了坐标解算算法在定位精度上的重要性。但即便如此，使用Levenberg–Marquardt代替原厂SX算法也带来了一系列好处：

1.首先是**计算速度**上有一定的优势。在每次定位时，我将上一次定位的结果作为迭代初值，由于目标运动时定位频率较高，两次定位的结果不会差距特别远，使得LM法的收敛变得非常迅速。在Centos7虚拟机环境下，使用time指令对比两种算法C++代码的运行速度，LM算法的计算速度甚至比非迭代式的SX算法还要快一点（计算同一组TDOA数据）。

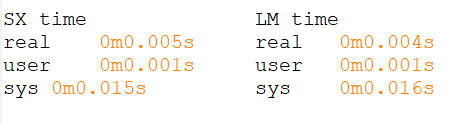


图1-1 运行时间对比

2.其次是**数值稳定性**。原厂的SX算法中，除了需要求解伪逆矩阵，还需要求解二次方程。当时，则无法计算出解。因而当TDOA出现异常数据或者目标处在比较奇怪的位置时，SX算法经常无法计算或者计算出异常解，使定位结果飘飞很远。而LM法不存在无法得出结果的情况，最糟糕的结果无非是在到达设定的最大迭代数前无法求得满意的解，在这种情况下抛出的非最优解并不会距离上次定位结果太远。

3.然后是**可扩展性**。由于LM法是比较通用的非线性优化算法，不需要考虑问题的线性化，求解方程组时定义的损失函数可以很直接，物理意义很明确。



这种情况下可以加入一个权重矩阵，动态调整各基站在定位中所占的权重。我比较直接地认为，在一次定位中，如果某个基站接收到的信号质量较好，那么应该给予它更大的权重。反之，则给予较小的权重。这样在基站数量充足的情况下，应该可以削弱受遮挡的基站对整体定位结果产生的影响。目前公司采用的DW1000芯片里有可以用来衡量接受信号质量的组件，但目前公司这款基站的产品里似乎没有设计这些数据的上报，我认为这部分功能未来是有利用价值的。

我前两天看到咱们学院郭树理老师2020年新发的论文《An Improved PDR/UWB Integrated System for Indoor Navigation Applications》中，已经提出了通过衡量接受信号的质量描述NLOS程度的方法，并且使用一种新的卡尔曼滤波计算权重矩阵。我其实之前就有类似的想法，但是没有能力去实践和探究，最近才发现别人已经做了。

## Tof模式定位算法开发

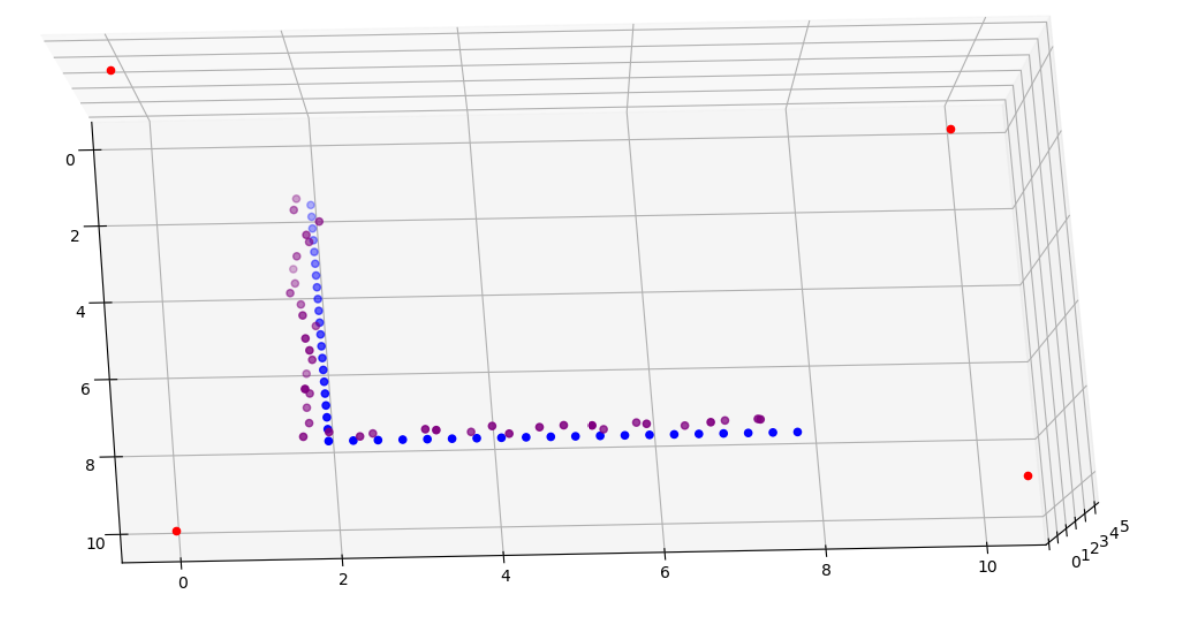
他们公司把之前的定位手环改了一下，现在手环也可以做双向测距了。工作在Tof模式下，可以用双向测距量测手环到各基站间的距离。但是他们不太会由测距值计算坐标，照着CSDN上的博客写了大几百行的垃圾代码，经常出错。

但其实这个问题光靠消去二次项就可以直接线性化，用线性最小二乘就能解决。参考论文《Localization Error Analysis of Indoor Positioning System Based on UWB Measurements》中的线性最小二乘估计，开发了Tof模式的定位算法。只用了十来行。

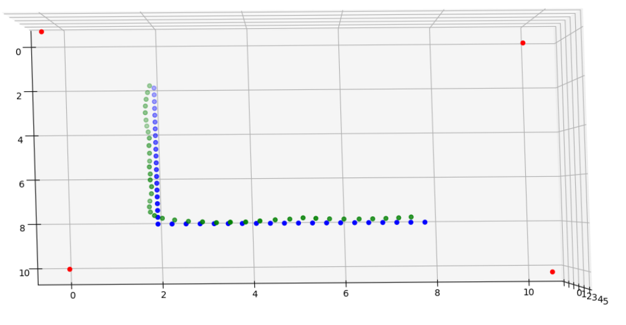
然后基于Levenberg–Marquardt法开发了两基站平面定位功能，解方程的方法和上面解TDOA的方法相同。这家公司之前使用两基站时只能做直线通道定位，现在两基站也可以做平面定位了，可以适应一些基站信号缺失，或成本不足的场景。

## 位置滤波

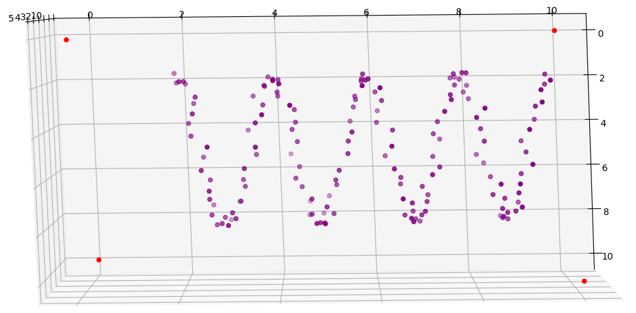
现有资源里没法用传感器做卡尔曼滤波，于是想基于历史定位数据做预测，假装出一个加速度计出来，与当前测量值做滤波融合。故采用Savitzky-Golay法对XYZ坐标进行滤波。该法采用一个滑动的滤波窗口，基于最小二乘法对时序信号进行在线的卷积平滑。Python仿真的结果如下。其中蓝色点为真实值。紫色点为噪声下的使用LM算法的位置测量值，绿色点为滤波之后的值。滤波窗口宽度设置为9个采样点，多项式阶次设置为2。



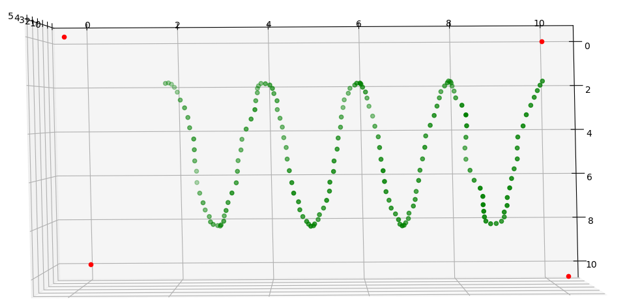
直角轨迹下测量值与真值对比（俯视图）



直角轨迹下滤波值与真值对比（俯视图）



正弦轨迹下的测量值（俯视图）



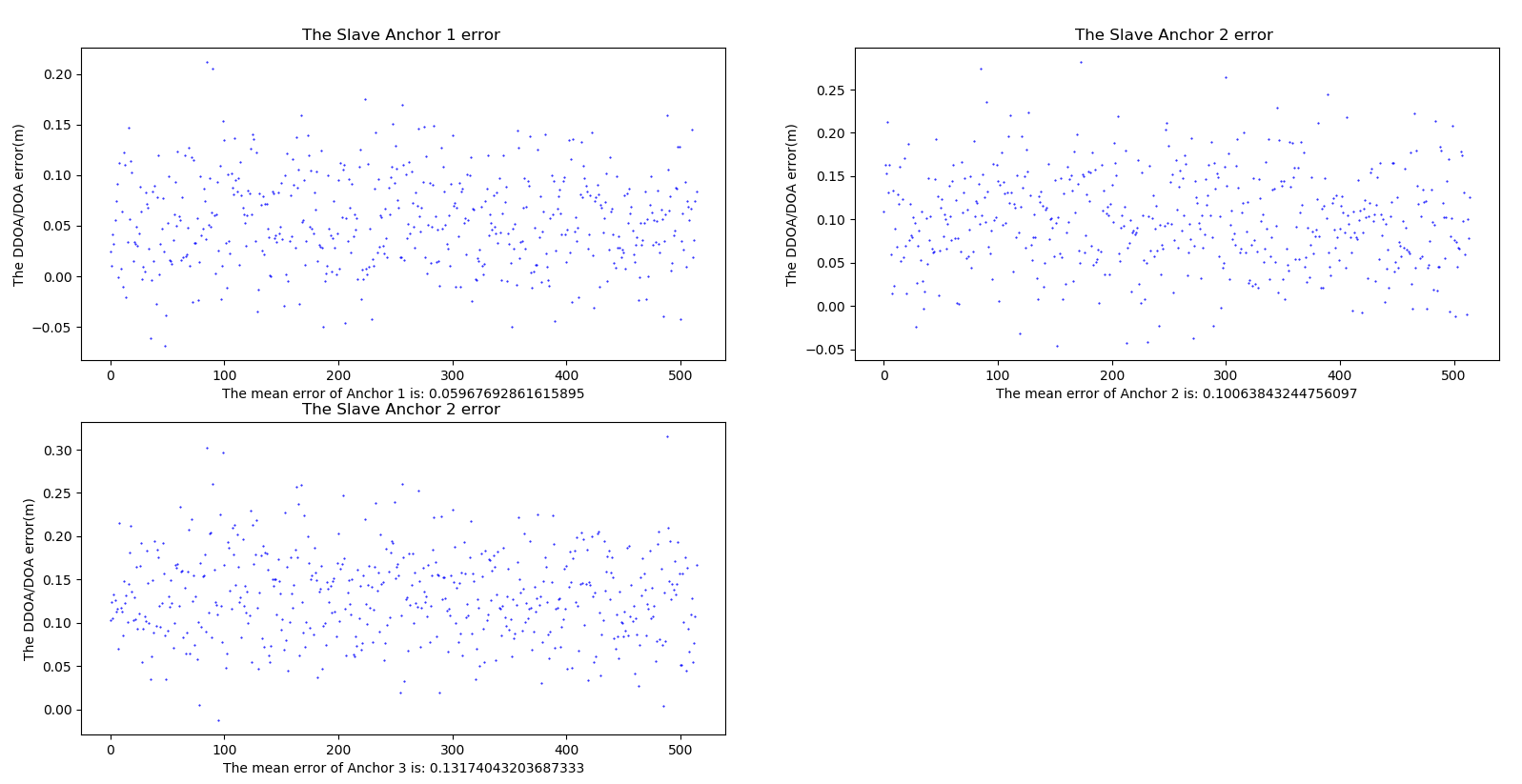
正弦轨迹下滤波值（俯视图）

这块的C++实现是借用了一个Github上的开源库，该库实现了单个窗口的SG滤波，我用boost库里的环形缓冲区做了滑动的滤波窗口，用来对XYZ信号进行实时的滤波处理。这一块其实如果能把滤波窗口和阶次做成自适应的会比较好，但是他们公司的人催的很急，从来第一天就催，所以就没优先研究自适应，先做了一个普通的。

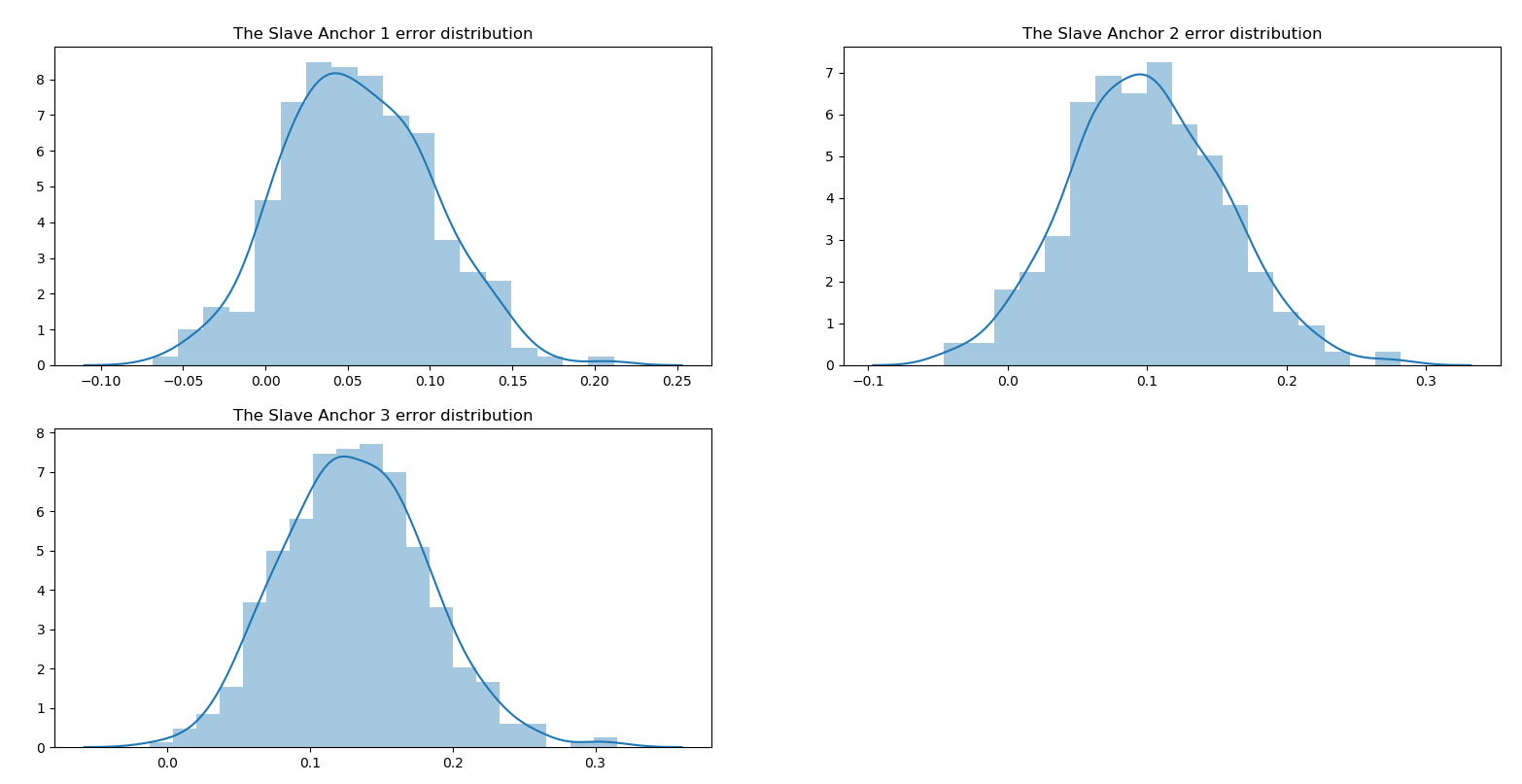
## 定位精度指标建立

这家公司的人对定位精度是没有概念的，不知道什么是准什么是不准，只能看到显示界面上的点飘不飘，跳不跳。于是写了文档给他们科普了均值、方差、均方根误差、圆概率误差等基本概念，确立了误差均值、误差统计分布、定位结果均方根误差等为定位精度指标。又编写了处理Log数据的Python代码，基于Log文件可以生成可视化的图表和统计量：

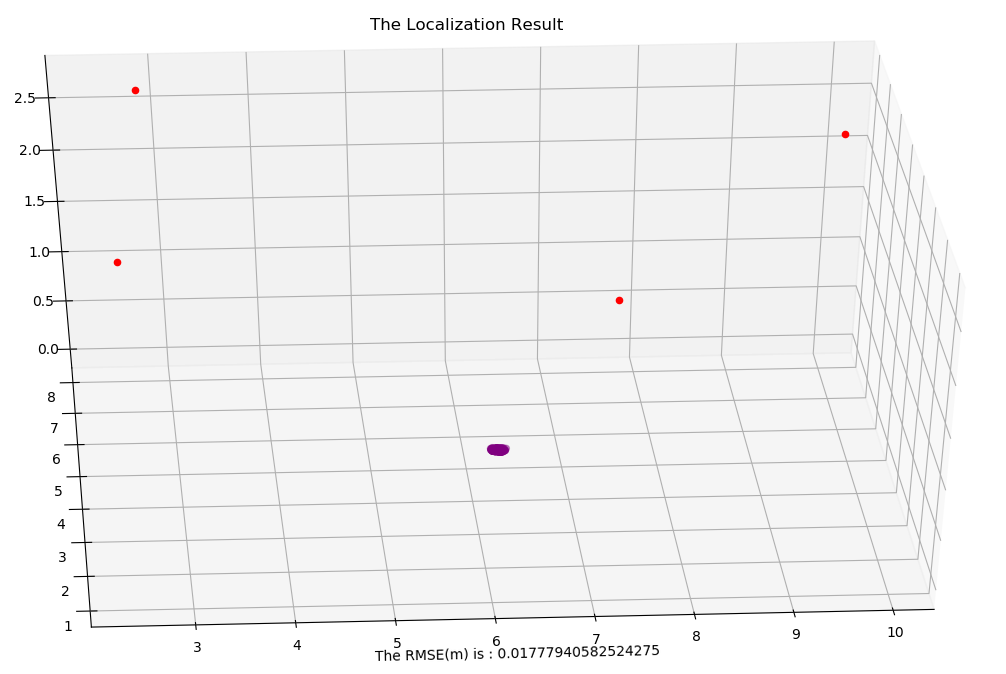
1. 各基站DDOA/DOA测距误差的散点图，以及各基站测距的误差均值：



2.各基站测距误差的统计分布图，用来描述测量噪声的概率分布：



3.定位轨迹回放：

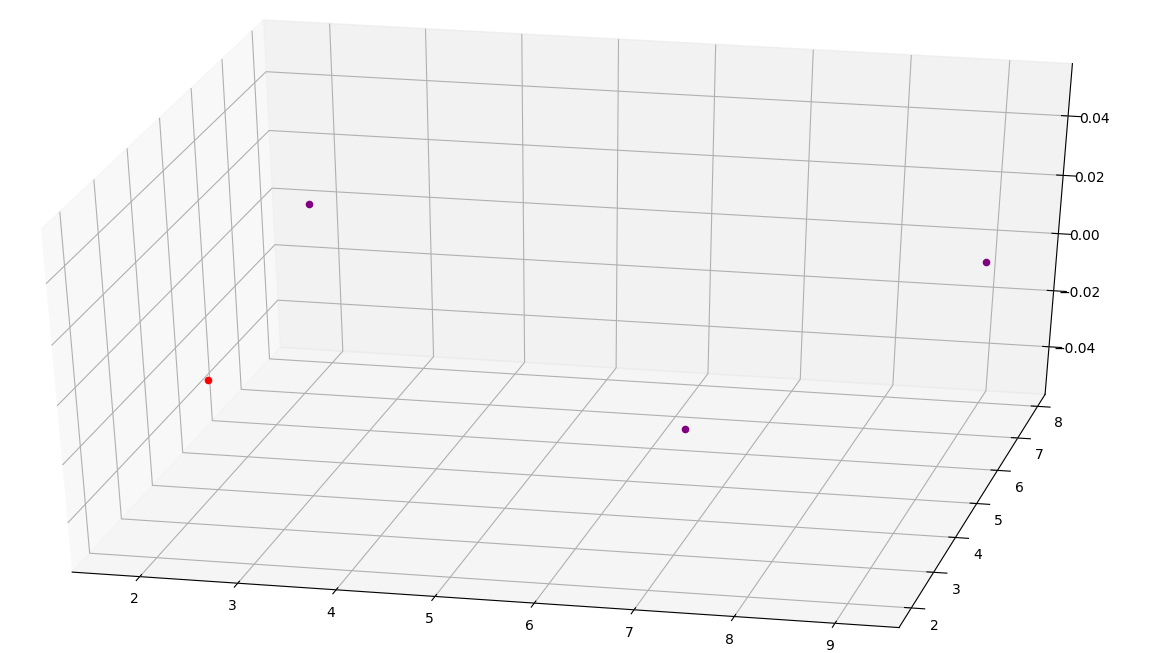


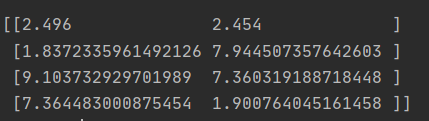
其中红色点为基站，蓝色点为标签位置真值，紫色点为定位轨迹。这批数据是静态测量得到的。处理运动目标的定位数据会画出轨迹。

## 基站位置自标定

我本来是想用老师您之前教我的类似于棋盘标定法一样给基站做自标定。但是他们仍然觉得麻烦。他们的人员其实在设置基站、标定位置的过程中是相当马虎的，很多时候都是用一下基站间的双向测距，然后拖动基站图标在界面上划拉，让几个基站在界面上的距离与测距值尽量差不多就完事。于是我想干脆就拿双向测距做自标定，也不用棋盘了，批量下发测距指令，遍历每个基站间的两两连接，用回报的测距值解方程，解出各基站的坐标。在解非线性方程组时，由于想做成全自动的，所以没有采用基于初值的优化方法，而是直接用了模拟退火做全局搜索。

下图是计算出来这家公司办公室天花板的四个基站位置：





## 这家公司的情况（主观）

从人员上看，这家公司研发力量比较薄弱，员工数量较少，能力也主要是侧重于应用开发，对于定位这件事的认识是较为不足的。并且没有人能看懂原厂算法。在这种比较薄弱的基础下，想推动一些需要成本的技术方案（时间、人力、硬件）对于我和张家铭两个学生来说还是有难度的。我们能做的事情还是比较有限的。