西安交通大學

毕业设计（论文）

题　目　基于到达时间量测的定位算法设计及仿真

电信学部自动化　学院　自动化　专业　自动化 81　班

学生姓名　张杨

学　　号　2186113658

指导教师　蔡远利

设计所在单位　西安交通大学电信学部自动化学院

2022　年　6　月

系　（专业）0504 自动化
系 (专业) 主任　张爱民
批 准 日 期　2021 年 10 月 21 日

西 安 交 通 大 学

毕业设计(论文)任务书

电子与信息学部　学院　0504 自动化　专业　自动化 81　班　　学生　张杨

毕业设计(论文)课题　基于到达时间量测的定位算法设计及仿真

毕业设计(论文)工作自　2021　年 9 月　11　日起至　2022　年　6　月　22　日止

毕业设计(论文)进行地点：　东二楼控制工程研究所（251）

课题的背景、意义及培养目标

在移动定位和无线传感器中，传感器通常需要测量目标信号源的到达时间来对目标的位置进行估计，这一问题广泛存在于军事与民用领域，具有重要的理论意义和应用价值。本毕业设计旨在学习并理解点估计理论的基本方法和原理，在此基础上针对目标定位问题设计优化算法并进行仿真验证，提升学生问题分析、软件设计及程序编写等工作能力。

设计(论文)的原始数据与资料

1、目标定位仿真平台；2、定位算法及优化方法相关文献；3、Python、Matlab 仿真编程环境。

课题的主要任务

1、学习并掌握最小二乘估计、极大似然估计和最大后验估计的基本原理和区别；2、建立定位问题的系统模型以及目标函数；3、针对多传感器定位问题，考虑传感器能够测量目标信号到达时间，学习并分析包括解析算法、迭代算法和半正定松弛算法在内的三类优化算法；4、分析传感器布局、目标位置和量测噪声等因素对定位效果的影响，并在不同条件下对比不同算法的估计精度；5、设计并开发仿真软件。

课题的基本要求(工程设计类题应有技术经济分析要求)

1、采用面向对象编程语言 Python 或 Matlab；2、实现基于到达时间量测的定位算法仿真程序；3、开发用户友好的交互界面，实现数据及参数分析；4、分析传感器布局、目标位置和量测噪声等因

素对定位效果的影响。

完成任务后提交的书面材料要求(图纸规格、数量，论文字数，外文翻译字数等)

1、仿真软件及使用说明；2、毕业设计论文：1.5~2.0 万字；3、英文翻译：5000~7000 字符。

主要参考文献

1.GuvencI,ChongCC.AsurveyonTOAbasedwirelesslocalizationandNLOSmitigationtechniques[J].IEEECommunicationsSurveysTutorials,2009,11(3):107-124.2.BeckA,StoicaP,LiJ.Exactandapproximatesolutionsofsourcelocalizationproblems[J].IEEETransactionsonSignalProcessing,2008,56(5):1770-1778.3.CheungKW,SoHC,MaWK,etal.LeastsquaresalgorithmsfortIme-of-arrival-basedmobilelocation[J].IEEETransactionsonSignalProcessing,2004,52(4):1121-1130.4.YanJ,TiberiusC,BellusciG,etal.FeasibilityofGauss-NewtonmethodforindoorpositioningIC].InProceedingsofIEEE/IONPLANS,2008:660-670.

指导教师：蔡远利

接受设计(论文)任务日期：2021 年 10 月 21 日

（注：由指导教师填写）　　学生签名：

西 安 交 通 大 学

毕业设计(论文)考核评议书

电子与信息学部 学院 0504 自动化 专业 自动化 81 班

指导教师对学生 张杨 所完成的课题为 基于到达时间量测的定位算法设计及仿真 的毕业设计(论文)进行的情况，完成的质量及评分的意见：张杨同学的毕业设计论文对基于时间到达量测的定位算法进行了研究和实现，分析了解析法、迭代法、半正定法等实现方法，开发了仿真分析软件，基于均方根误差和克拉美罗下界，进行了大量仿真计算，分析了测量误差、位置分布、目标位置对算法预测精度的影响。

论文写作认真，格式规范，工作量饱满。毕业设计期间工作积极、主动，较好地完成了规定的毕业设计任务。

建议评分：A-。

指导教师建议成绩： A-

指导教师：

2022 年 06 月 14 日

毕业设计(论文)评审意见书

评审意见： 论文针对时间到达量测问题，分别设计了视线场景、非视线场景、初始时间场景的解析法、迭代法、半正定法等多种算法，并实现了交互式仿真软件开发。

论文写作认真，格式规范，工作量较为饱满。

评阅教师建议成绩： A-

评阅人： 邓逸凡 职称： 助理研究员

2022 年 06 月

毕业设计(论文)答辩结果

_____学院

_____专业

毕业设计(论文)答辩组对学生_____所完成的课题为

_____

_____

的毕业设计(论文)经过答辩，其意见为_____

_____

_____

_____

_____

_____

_____

并确定成绩为_____

毕业设计(论文)答辩组负责人_____

答辩组成员_____ _____

_____ _____

_____ _____

年 月 日

摘　要

无线设备定位算法在现代社会中具有非常广泛的应用。无论是在日常生活中还是在军事领域中，定位算法都有重要的作用，如地图导航、外卖服务、导弹跟踪系统以及无人机侦察等。定位问题有很多基于不同量测方法的算法，如基于时间到达量测的定位算法（Time Of Arrival,TOA）、基于角度到达量测的定位算法（Angle Of Arrival,AOA）、基于时间差到达量测的定位算法（Time Difference Of Arrival,TDOA）、基于接收信号强度的定位算法（Receive Signal Strength,RSS）等。本文主要研究 TOA 算法。

本文首先介绍了时间到达量测问题（Time of Arrival,TOA）并给出了 TOA 问题的数学描述。根据影响因素的不同，TOA 问题可分为视线场景（Line of Sight,NLOS）和非视线场景（None-Line of Sight,NLOS），不同的场景有不同的数学描述。TOA 问题的算法主要是应用最大似然思想和最小二乘思想。基于这两种思想提出的算法可根据目标函数的不同使用不同的解决方法，因此产生了各式各样的 TOA 算法。根据考虑的影响因素的不同，本文依次介绍了 LOS 场景、NLOS 场景以及考虑初始时间场景下的 TOA 问题，依据解决方法的不同，分为解析法、迭代法、半正定法三部分介绍相关算法。

本文以均方根误差（root-mean-square deviation,RMSE）为衡量算法的指标，同时以克拉美罗下界（Cramer-Rao Lower Bound,CRLB）为理论下界，用蒙特卡洛法分析不同测量误差、不同位置分布、不同目标位置对算法预测精度的影响。经过仿真分析，本文得到如下结论：1. 随着基站数目的增加，预测误差下降。2. 随着目标站点接近基站分布的几何中心，预测误差下降。3. 当目标站点位于基站包围区域外时，预测误差大于目标站点位于包围区域内时的误差。4. 最小二乘法及其变式运行时间远小于最大似然法以及高斯牛顿迭代法的运行时间。最后，本文将仿真过程中的各功能集成开发了交互式仿真软件，可以方便的分析各因素对算法准确度的影响。

关键词：时间到达量测；最小二乘估计；最大似然估计；迭代法；解析法

ABSTRACT

Wireless device location algorithms have a very wide range of applications in modern society. Whether in daily life or in the military field, positioning algorithms play an important role, such as map navigation, food delivery services, missile tracking systems, and drone reconnaissance. There are many algorithms based on different measurement methods for the positioning problem, such as the positioning algorithm based on the time of arrival measurement (Time Of Arrival, TOA), the positioning algorithm based on the angle of arrival measurement (Angle Of Arrival, AOA), the arrival measurement based on the time difference The positioning algorithm (Time Difference Of Arrival, TDOA), the positioning algorithm based on the received signal strength (Receive Signal Strength, RSS) and so on. This paper mainly studies the TOA algorithm.

This paper firstly introduces the Time of Arrival (TOA) problem and gives the mathematical description of the TOA problem. According to different influencing factors, TOA problems can be divided into line of sight (NLOS) and non-line of sight (None-Line of Sight, NLOS) scenarios, and different scenarios have different mathematical descriptions. The algorithm of TOA problem mainly applies the idea of maximum likelihood and least squares. The algorithms proposed based on these two ideas can use different solutions according to the different objective functions, so various TOA algorithms have been generated. According to the different influencing factors considered, this paper introduces the LOS scenario, the NLOS scenario and the TOA problem considering the initial time scenario in turn. According to the different solutions, it is divided into three parts: analytical method, iterative method, and positive semi-definite method. The related algorithms are introduced.

In this paper, the root-mean-square deviation (RMSE) is used as the index to measure the algorithm, and the Cramer-Rao Lower Bound (CRLB) is used as the theoretical lower bound, and the Monte Carlo method is used to analyze different measurement errors. , The influence of different location distributions and different target locations on the prediction accuracy of the algorithm. After simulation analysis, this paper draws the following conclusions: 1. With the increase of the number of base stations, the prediction error decreases. 2. As the target site approaches the geometric center of the base station distribution, the prediction error decreases. 3. When the target station is located outside the encircled area of the base station, the prediction error is larger than that when the target station is located in the encircled area. 4. The running time of the least squares method and its variants is much smaller than that of the maximum likelihood method and the Gauss-Newton iteration method. Finally, this paper develops an interactive simulation software that integrates various functions in the simulation process,

which can easily analyze the influence of various factors on the accuracy of the algorithm.

KEY WORDS: Time of arrival measurement; least squares estimation; maximum likelihood estimation; iterative method; analytical method

目　录

1　绪论 1
1.1　应用背景和意义 1
1.2　国内外研究现状 1
1.2.1　LOS 场景下 TOA 问题研究现状 1
1.2.2　其他影响因素下 TOA 问题研究现状 2
1.3　本文工作简介 3
2　问题描述 4
2.1　问题描述 4
2.2　问题区分 5
3　算法研究 7
3.1　基于 LOS 场景的 TOA 算法研究 7
3.1.1　克拉美罗下限 7
3.1.2　迭代法 9
3.1.3　解析法 13
3.1.4　半正定法 17
3.2　基于 NLOS 场景的算法研究 19
3.2.1　NLOS-CRLB 19
3.2.2　识别丢弃法 20
3.2.3　鲁棒预测器 21
3.3　基于初始时间场景的算法研究 22
3.3.1　最小二乘法 23
3.3.2　半正定规划法 23
3.3.3　算法对比 26
4　仿真分析 27
4.1　基站数目对预测误差的影响 27
4.2　基站包围面积对预测误差的影响 27
4.3　目标基站在区域内外对预测误差的影响 28
4.4　目标位置对预测误差的影响 30
4.5　算法运行时间对比 30
4.6　测量误差对预测误差的影响 31
4.7　仿真结果总结 31

5 仿真软件介绍 33
5.1 初始背景界面 33
5.2 参数选择界面 33
5.3 基站位置分布界面 34
5.4 算法选择界面 34
6 结论与展望 35
6.1 结论 35
6.2 展望 36
致谢 37
参考文献 38
附录 A 翻译论文原文 40
附录 B 论文翻译 58
附录 C 源代码 79
附录 C.1 仿真 UI 79
附录 C.2 LOS 算法 97
附录 C.3 NLOS 算法 113
附录 C.4 仿真分析 116

1 绪论

1.1 应用背景和意义

在现代社会中，定位问题具有非常广泛的应用。无论是雷达搜索、激光定位、声纳定位，还是无线网络、蜂窝移动网络以及 GPS（全球定位系统），本质都是定位问题。定位算法的应用不仅给人们的日常生活带来方便，同时在各种应急情况下也发挥着重要作用，是现代社会运行的基石。日常生活中，手机导航可以帮助人们快速到达目标位置，极大的节省了时间。当人们使用外卖服务时，往往需要获取自己的位置信息从而使订单准确送达，或者是使用社交软件中的附近好友功能，也需要用户的定位信息。在极端场景中，apple watch 可以在用户野外遇难时，自动向应急中心发送手机位置。军事中，巡航导弹需要定位目标位置才能准确击打目标，雷达系统也需要通过定位算法确定地方导弹的轨迹来进行拦截，反恐行动中，无人机需要在室内定位恐怖分子的位置才能进行有效打击。由此可见，定位问题具有重要的研究意义。

1.2 国内外研究现状

在过去的几十年里，定位问题已经得到了广泛的研究，此前已有许多文献提出了优秀的方法[1][2][3][4] 如：基于时间到达量测的定位算法（Time Of Arrival）、基于角度到达量测的定位算法（Angle Of Arrival）、基于时间差到达量测的定位算法（Time Difference Of Arrival）、基于接收信号强度的定位算法（Receive Signal Strength）等。由于在导航系统、检测系统、无线通信系统中的广泛应用，基于 TOA 信息的目标定位问题得到了广泛的关注和研究。

TOA 问题主要可以分为两个场景，即视线场景（line Of Sight,LOS）和非视线场景（None-Line Of Sight,NLOS）。早期大部分算法主要研究 LOS 场景下的目标位置求解问题。

1.2.1 LOS 场景下 TOA 问题研究现状

1964 年，Marchand 在文献 [5] 中第一次用最大似然的思想求解目标位置，基于噪音分布为高斯分布的假设，求使得概率最大的解，即为目标位置。此外，该文献还进一步考虑了噪声分布等因素。

1972 年，Turin 在文献 [6] 中第一次提出用最小二乘法求解目标位置定位问题，即求使误差最小的解。至此，解决定位问题的两大思想都已提出。后续文献在这两个思想的前提下，提出了各式各样的改进算法，同时引入了更多的影响因素，不断深入 TOA 定位问题的研究。

2004 年，Chang 在文献 [7] 中给出了 TOA 问题在 LOS 场景下的克拉美罗下限

（Cramer-Rao Lower Bound,CRLB），这是算法准确度的理论极限，也是衡量算法优劣的重要指标。

文献 [8] 基于最大似然思想，提出了改进的最大似然算法并应用在 TOA 问题上。该改进算法的主要贡献是引入迭代法的思想，通过选根公式求解最优目标位置。

文献 [9][10] 基于最小二乘思想给出了 TOA 问题在视线场景（Line Of Sight,LOS) 下的算法。在理论层面分析了影响最小二乘法的准确度。该算法通过分析得知误差与测量距离有关，因此提出了两个变式以提高预测的准确度，分别是求平均法降低误差影响，以及选取最短距离作为参考距离。最小二乘法在该问题的应用具有极大的实践意义，不仅耗时短，而且复杂度低，在较低精度要求下具有极大的实际价值。此外，为了解决当基站数目增多，矩阵大小增大导致的求逆问题，文献 [3] 和文献 [11] 应用最速梯度法和高斯牛顿迭代法进行求解，进一步提高了最小二乘的实际价值。

由上述介绍可以知道，TOA 问题的两个主要解决思路是最大似然和最小二乘。最大似然法利用测量噪音为高斯分布的先验信息求最大概率，最小二乘思路是使误差最小，这两种思路对应的求解方法可为迭代法、解析法和半正定法，因而衍生了不同的算法。求解算法主要取决于目标函数的形式，当目标函数为测量距离平方与实际距离平方差的最小二乘时，往往使用解析法，这是因为平方使得欧氏距离变换为更易于处理的二次多项式形式。当目标函数为测量距离与实际距离差的最小二乘时，往往使用迭代法。

随着计算机硬件的不断提升，之前一些难以应用的求解算法被引入 TOA 问题。文献 [12] 给出了半正定法在 TOA 问题上的应用。半正定法的引入，给了算法求解的第三种思路，半正定松弛法可以将非凸二次优化问题转化为凸优化并求出较优解，应用范围非常广泛。对于 LOS 场景下的 TOA 问题，可以给出比最小二乘法更高的求解精度。

1.2.2 其他影响因素下 TOA 问题研究现状

针对 LOS 场景之外的情况，也有许多进展。文献 [13] 提出利用残差法进行 NLOS 基站的识别丢弃的思想在解决 NLOS 场景下具有重要意义。无论权重怎样调，在基站数目足够的情况下，只选择 LOS 的基站的准确度都要高于包含 NLOS 的准确度。文献 [14] 给出了 NLOS 场景下的鲁棒预测器，利用 Huber 预测器，对异常值进行处理，进一步提高了算法的鲁棒性。[9] 则将中位数最小二乘法的思想应用到 NLOS 场景的 TOA 定位问题中，可以抵抗最多 50% 的异常值。文献 [15] 则提出了利用半正定规划法解决带初始时间的 TOA 问题，使得算法更具有实际意义，进一步提高了算法实际应用的价值。

除了 NLOS 误差外，文献 [3] 还指出, 影响 TOA 预测精度的影响因素有快速衰落信道、低信噪比 (signal-to-noise ratio,SNR)、多径效应和多用户干扰等，并针对这些因素给出改善建议如，提高时钟频率，提高信噪比，过采样，慢变通道等。此外还对比分析了 AOA、TOA、TDOA 问题误差的理论下限。该文献不仅对 TOA 算法的实际应用有重要意义而且为为后续算法提供了理论支持和参考标准。

随着人们对实时定位要求的不断提高，针对 TOA 实时定位的研究也越来越多。文献 [16] 介绍了一种改进卡尔曼滤波的方法动态预测移动站点，适用于 LOS 和 NLOS 场景下的预测，具有较大的实际应用价值。此外，文献 [17] 探讨了在信号频率、初相、幅度未知条件下的到达时间估计算法。该方法利用 Haar 小波对信号进行处理作为到达时间精估计，进而提高了预测精度。这篇文献从接收信号的角度提高定位准确度，提供了一个新的视角，具有重要意义。在实际当中，目标往往位于地球表面，此时可以利用目标高度约束信息来进一步提升目标的定位精度。文献 [18] 把目标高度约束引入到基于到达时间的地面目标定位中，提升了目标定位精度。

尽管 TOA 问题取得了很多进展，但仍然有许多值得探讨的地方。在工程实际应用中，各种场景对定位问题的要求不同，因此研究不同算法在不同场景下的优劣具有重要意义。此外，算法往往假设的是理想情况，而实际中受限于经费、环境干扰，往往不能在基站数目与硬件性能方面同时获得最优条件，研究数目与硬件性能对算法精度的影响同样有重要意义。更重要的是，现实情况中除了发送信号初始时间偏差，还有基站位置误差等因素干扰。在这些情况下，如果没有全球定位系统 (Clobakl Position System,GPS)，使用原有的算法将变得不切实际。这些实际应用中的工程问题具有重要的研究意义。

1.3 本文工作简介

本文主要分为如下部分，首先介绍 TOA 在 LOS 场景，NLOS 场景以及考虑初始时间场景下的各类算法，包括迭代法、解析法以及半正定法。其次利用仿真比较分析各算法的优劣。接着分析了测量误差、移动站点位置、基站分布对预测精度的影响。最后开发了包括 LOS 场景、NLOS 场景并集成各类算法的仿真软件。该仿真软件可以更改测量误差，基站分布，选择定位算法，通过该仿真软件可以分析不同场景下的算法效果。

本文主要按如下结构分布。1. 问题陈述。该部分对 TOA 进行数学建模，公式化该问题并定义了各变量的表示；2. 算法研究部分. 该部分首先针对 LOS 场景，介绍了应用解析法，迭代法，半正定法的算法并进行讨论。然后针对 NLOS 场景介绍了识别丢弃算法并于最小二乘法对比分析。接着介绍了初始时间场景下的半正定算法并进行了分析；3. 仿真分析部分。该部分利用仿真分析了相同情况下各算法的误差比较以及运行时间比较。此外还探讨了基站分布、基站数目、目标站点位置等对算法准确度的影响；4. 仿真软件介绍部分。该部分介绍了基于 TOA 的各算法的集成仿真平台；5. 结论与展望部分。

2 问题描述

2.1 问题描述

如图2-1所示，最少三个基站即可预测出目标站点位置。令 **x** 为目标站点实际位置，$\hat{\mathbf{x}} = \begin{bmatrix} \hat{x} & \hat{y} \end{bmatrix}^T$ 为预测位置，$\mathbf{x}_i = \begin{bmatrix} x_i & y_i \end{bmatrix}^T$ 为第 i 个基站的位置，$\hat{d}_i$ 是移动终端 MT 的位置与第 i 个基站之间的预测位置，可以建模表示为

$$\hat{d}_i = d_i + b_i + n_i + d_0 = ct_i, i = 1, 2, ...N \tag{2-1}$$

其中，t_i 表示第 i 个 FT 的 TOA，c 是光的传播速度，d_i 是 MT 和第 i 个 FT 之间的实际距离。$n_i \sim \mathcal{N}(0, \sigma_i^2)$ 是附加的方差为 σ_i^2 的高斯白噪声（AWGN），d_0 是初始时间误差导致的测量距离误差，b_i 是有路径阻碍引入的正偏差，由下式给出

$$b_i = \begin{cases} 0, \text{ if } i \text{ th FT is LOS} \\ \psi_i, \text{ if } i \text{ th FT is NLOS} \end{cases} \tag{2-2}$$

对于 NLOS 的基站 FTs，偏差通常为指数函数。令

$$\mathbf{d} = d(x) = \begin{bmatrix} d_1, d_2, ..., d_N \end{bmatrix}^T \tag{2-3}$$

表示基站与移动终端之间实际距离的向量，

$$\hat{\mathbf{d}} = \begin{bmatrix} \hat{d}_1, \hat{d}_2, ..., \hat{d}_N \end{bmatrix}^T \tag{2-4}$$

表示测量距离，

$$\mathbf{b} = \begin{bmatrix} b_1, b_2, ..., b_N \end{bmatrix}^T \tag{2-5}$$

表示偏差向量。同样，令

$$\mathbf{Q} = \mathbb{E}\left[\mathbf{n}\mathbf{n}^T\right] = diag \begin{bmatrix} \sigma_1^2, \sigma_2^2, ..., \sigma_N^2 \end{bmatrix}^T \tag{2-6}$$

表示误差矩阵 $\mathbf{n} = [n_1, n_2, ..., n_N]^T$ 的协方差矩阵。
在没有测量误差和 NLOS 偏差时，MT 的位置可由基站 FT 以及测量距离确定的圆的交点确定，

$$(x - x_i)^2 + (y - y_i)^2 = d^2, i = 1, 2, ..., N \tag{2-7}$$

然而实际情况中，测量误差和 NLOS 偏差存在并导致这些圆的交点不在一点上。这时表示为，

$$(x-x_i)^2+(y-y_i)^2=\hat{d}^2, i=1,2,....,N \tag{2-8}$$

为了有更紧凑的表达，本文定义，

$$s=x^2+y^2, k_i=x_i^2+y_i^2 \tag{2-9}$$

公式2-8可以重写为,

$$s+k_i-2\left(xx_i+yy_i\right)=\hat{d}^2, i=1,2,....,N \tag{2-10}$$

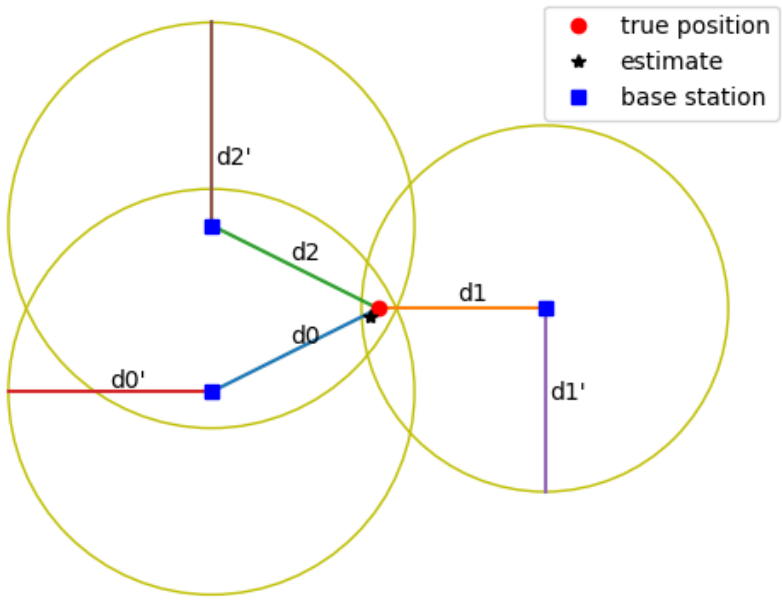


图 2-1 无线定位示意图

2.2 问题区分

TOA 定位问题实际上为已知测量距离 d_i，求目标基站位置 $\mathbf{x}$ 的问题。区别在目标函数的表达形式上，主要可分为两种形式，第一种形式是求测量距离与实际距离差值的平方，

$$\hat{\mathbf{x}}=\arg\min_{\mathbf{x}}\left\{\sum_{i=1}^{N}\left(\hat{d}_i-\|\mathbf{x}-\mathbf{x}_i\|\right)^2\right\} \tag{2-11}$$

第二种是求测量距离的平方与实际距离平方之差的平方，

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \left\{ \sum_{i=1}^{N} \left(\hat{d}_i^2 - \|\mathbf{x} - \mathbf{x}_i\|^2 \right)^2 \right\} \tag{2-12}$$

针对这两种不同的算法表达，有不同的求解方法，因此分为迭代法、解析法、半正定法三类方法。

3 算法研究

3.1 基于 LOS 场景的 TOA 算法研究

在本部分，本文首先讨论 LOS 场景下的 TOA 问题，忽略 NLOS 偏差、初始时间误差等因素的影响，以最简单的形式关注问题的本质。本文主要探讨在 LOS 场景的克拉美罗下限（CRLB）以及迭代法，解析法和半正定法（SDR）在 TOA 问题上的具体应用。

3.1.1 克拉美罗下限

克拉美罗下限通常用来表示预测值理论方差的最小值，是衡量信息量的重要指标。

为了确定 LOS 场景下的克拉美罗下限[7][8]，本文假设测量噪声是独立且一致分布的，高斯噪声为零均值且方差为 σ^2。在该前提下，条件概率密度函数如下

$$\begin{aligned} p\left(\hat{d}_1, \hat{d}_2, \ldots, \hat{d_N} \mid \mathbf{x}\right) &= \left(2\pi\sigma^2\right)^{-N/2} \\ &\times \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{N}\left(\hat{d}_i - \|\mathbf{x}_i - \mathbf{x}\|\right)^2\right) \end{aligned} \tag{3-1}$$

对应的 FISHER 信息矩阵为,

$$[\mathbf{I}(\mathbf{x})]_{ij} = -\mathrm{E}\left[\frac{\partial^2 \ln P(\hat{\mathbf{d}} \mid \mathbf{x})}{\partial \mathbf{x}_i \partial \mathbf{x}_j}\right] \tag{3-2}$$

代入上式后，

$$\begin{aligned} \mathbf{I}(\mathbf{x}) &= \begin{bmatrix} \sum_{i=1}^{N} \frac{(x-x_i)^2}{\sigma_i^2 d_i^2} & \sum_{i=1}^{N} \frac{(x-x_i)(y-y_i)}{\sigma_i^2 d_i^2} \\ \sum_{i=1}^{N} \frac{(x-x_i)(y-y_i)}{\sigma_i^2 d_i^2} & \sum_{i=1}^{N} \frac{(y-y_i)^2}{\sigma_i^2 d_i^2} \end{bmatrix} \\ &= \begin{bmatrix} \sum_{i=1}^{N} \frac{\cos^2(\alpha_i)}{\sigma_i^2} & \sum_{i=1}^{N} \frac{\cos(\alpha_i)\sin(\alpha_i)}{\sigma_i^2} \\ \sum_{i=1}^{N} \frac{\cos(\alpha_i)\sin(\alpha_i)}{\sigma_i^2} & \sum_{i=1}^{N} \frac{\sin^2(\alpha_i)}{\sigma_i^2} \end{bmatrix} \end{aligned} \tag{3-3}$$

其中，α_i 为 MT 与第 i 个 FT 之间的夹角。CRLB 由 $\mathbf{I}(\mathbf{x})^{-1}$ 得到。因此，对于任何无偏估计器估计出的 MT 的位置 $\mathbf{x}$,

$$\mathrm{E}_{\mathbf{x}}\left[(\hat{\mathbf{x}} - \mathbf{x})(\hat{\mathbf{x}} - \mathbf{x})^T\right] \geq \mathbf{I}^{-1}(\mathbf{x}) \tag{3-4}$$

此外，CRLB 可能与另外一个被称作几何稀释精度（GDOP）的测量值相关，当不同测量基站的噪声一致分布，即 $\sigma_i^2 = \sigma^2$ 时,GDOP 可以表达为，

$$\mathrm{GDOP} = \frac{\mathrm{RMSE}_{\mathrm{loc}}}{\mathrm{RMSE}_{\mathrm{range}}} = \frac{\sigma_{\mathrm{loc}}}{\sigma} \tag{3-5}$$

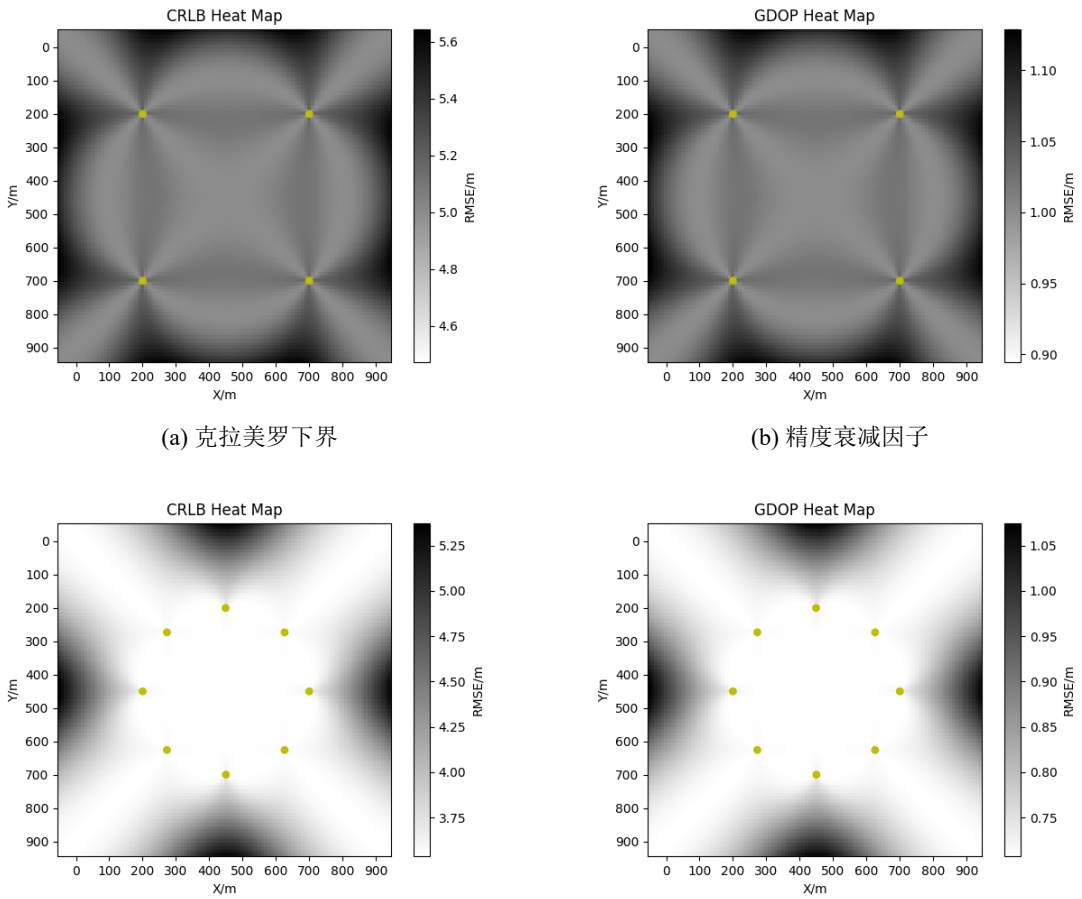
其中 $\mathrm{RMSE}_{\mathrm{loc}}$ 和 $\mathrm{RMSE}_{\mathrm{range}}$ 是位置预测和范围预测的均方根误差，相应的 σ_{loc} 是位置预测的标准差。GDOP 高度依赖于基站 FTs 和移动站 MT 的位置分布，且 GDOP 值小于 3 时通常是可取的，大于 6 时则意味着 FT 的几何位置不好。如果位置预测器可以实现 CRLB，则 GDOP 由下式给出，

$$\begin{aligned}\mathrm{GDOP} &= \frac{\sqrt{\mathrm{trace}\left[\mathbf{I}^{-1}(\mathbf{x})\right]}}{\sigma}, \\ &= \sqrt{\mathrm{trace}\left[\tilde{\mathbf{I}}^{-1}(\mathbf{x})\right]},\end{aligned} \tag{3-6}$$

其中，

$$\tilde{\mathbf{I}}(\mathbf{x}) = \begin{bmatrix} \sum_{i=1}^{N}\cos^2(\alpha_i) & \sum_{i=1}^{N}\cos(\alpha_i)\sin(\alpha_i) \\ \sum_{i=1}^{N}\cos(\alpha_i)\sin(\alpha_i) & \sum_{i=1}^{N}\sin^2(\alpha_i) \end{bmatrix} \tag{3-7}$$

当本文固定四个 FT 在位置 [0,0],[0,50],[50,0],[50,50] 时，改变 MT 的位置，CRLB 的结果如图3-1所示。



(a) 克拉美罗下界　　(b) 精度衰减因子

(c) 克拉美罗下界 1　　(d) 精度衰减因子 1

图 3-1　克拉美罗下界和精度衰减因子热图

由图3-1可知，CRLB 和 GDOP 呈一致分布。当基站分布为方形时，靠近中间位置的 CRLB 更小。从公式3-3可以看出，x 和 y 的对应方差为矩阵 $\mathbf{I}(\mathbf{x})^{-1}$（1，1）和（2，

2）项，而这与 $\frac{(x-x_i)^2}{d_i^2}$ 正相关。当目标位置处于各基站中点时，横坐标之差占总距离之比的和最大，求逆则为最小，因此目标站点越靠近中间位置 CRLB 越小，上图与本文的理论分析一致。当目标站点分布在基站区域外时，情况相反，当目标站点靠近测量基站时，CRLB 的值要小于处于测量基站中间的位置。这是因为当靠近测量基站时，存在于基站连线的延长线上，因此该部分的夹角极小，对应的 CRLB 就会偏小。

3.1.2 迭代法

迭代法也称辗转法，是一种不断用变量的旧值递推新值的过程，与迭代法相对应的是直接法 (或者称为一次解法)，即一次性解决问题。迭代算法是用计算机解决问题的一种基本方法，它利用计算机运算速度快、适合做重复性操作的特点，让计算机对一组指令 (或一定步骤) 进行重复执行，在每次执行这组指令 (或这些步骤) 时，都从变量的原值推出它的一个新值 [19]。本部分介绍了使用迭代法求解的最大似然法 [20]，并提出了高斯牛顿迭代法在 TOA 问题中的应用。

1) 高斯牛顿迭代法

高斯牛顿迭代法是一种经典的迭代法，它利用了目标函数的二阶海森矩阵，因此具有较快的收敛速度。高斯牛顿迭代式为，

$$\mathbf{x_k} = \mathbf{x_{k-1}} - \mathbf{H^{-1}J} \tag{3-8}$$

其中，**H** 为二阶海森矩阵，**J** 为一阶雅可比矩阵。对于 TOA 问题，目标函数可以写为

$$f(\mathbf{x}) = \sum_{i=1}^{N}\left(\hat{d}_i - \sqrt{(x_i - x)^2 + (y_i - y)^2}\right)^2 \tag{3-9}$$

当移动基站位置固定在 (400,300) 时，绘制上述公式在 $x \in (0, 1000)$,$y \in (0, 600)$ 的函数值 3D 图。如图3-2。 由3-2可知，只要初始位置选择在合适的范围，可以通过有限迭代求得最优解。

关于初始值的求解，本文使用线性化方法。本文取等式2-8前三项，二三项分别减去第一项后得两线性等式。几何意义为第二个第三个基站与第一个基站确定的圆相交得两直线。求解后得到的初始值即为两直线的交点。求解公式为，

$$\begin{aligned} y &= \frac{f - \frac{lc}{a}}{e - \frac{lb}{a}} \\ x &= \frac{c - by}{a} \end{aligned} \tag{3-10}$$

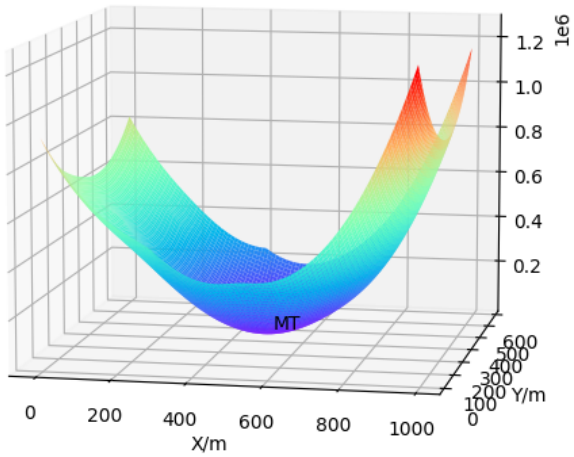


图 3-2　目标函数 3D 图

其中，

$$
\begin{aligned}
&a = 2\left(x_1 - x_0\right), \quad b = 2\left(y_1 - y_0\right) \\
&l = 2\left(x_2 - x_0\right), \quad e = 2\left(y_2 - y_0\right) \\
&c = \hat{d_0}^2 - \hat{d_1}^2 - \left({x_0}^2 - {x_1}^2 + {y_0}^2 - {y_1}^2\right) \\
&f = \hat{d_0}^2 - \hat{d_2}^2 - \left({x_0}^2 - {x_2}^2 + {y_0}^2 - {y_2}^2\right)
\end{aligned} \tag{3-11}
$$

然而，求解初始值时存在三个基站位于同一点的情况。在该情况下，无法用求解公式3-11求解。针对这种情况，利用如下公式进行求解，

$$
\begin{aligned}
x &= \frac{\left({d_1}^2 - {d_0}^2\right) - \left({x_1}^2 - {x_0}^2\right)}{2\left(x_0 - x_1\right)} \\
y &= \sqrt{{d_0}^2 - \left(x - {x_0}^2\right)} + y_0
\end{aligned} \tag{3-12}
$$

求得初始值后，代入公式3-8。该目标函数的一阶雅可比矩阵为

$$
\mathbf{J} = \left[\frac{\partial f\left(\mathbf{x}\right)}{\partial x}, \frac{\partial f\left(\mathbf{x}\right)}{\partial y}\right]^T \tag{3-13}
$$

二阶海森矩阵为

$$
\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f(\mathbf{x})}{\partial x^2} & \frac{\partial^2 f(\mathbf{x})}{\partial x \partial y} \\ \frac{\partial^2 f(\mathbf{x})}{\partial x \partial y} & \frac{\partial^2 f(\mathbf{x})}{\partial y^2} \end{bmatrix} \tag{3-14}
$$

在求出目标函数的雅可比矩阵和海森矩阵后，设置阈值 θ, 当 $\|\mathbf{x_k} - \mathbf{x_{k-1}}\| < \theta$ 时跳出。经过仿真，通常 θ 设置在 10^{-3} 左右。

综上，高斯牛顿迭代法的步骤如下, 关于高斯牛顿迭代法的仿真结果在**仿真展示**部分。

算法 3-1 Gauss-Newton Iteration

1) 设置超参数阈值 θ 和迭代次数 N；

2) 利用公式3-11求出初始位置 (x_0, y_0)；

3) 将位置 (x_0, y_0) 代入公式3-13,3-14中求出对应的雅可比矩阵和海森矩阵；

4) 将雅可比矩阵与海森矩阵代入公式3-8中求出下一个位置 $\mathbf{x_k}$；

5) 判断 $\|\mathbf{x_k} - \mathbf{x_{k-1}}\| < \theta$ 以及迭代次数是否大于 N，若是返回 x_k, 否则重复步骤 3-5。

2) 最大似然迭代法

高斯牛顿迭代法的目标函数是基于最小二乘写出，目的是求出最小的误差。而最大似然迭代法与高斯牛顿不同，最大似然法是基于概率进行求解。该方法基于各基站测量的噪音误差为一致高斯分布的假设，在已经测量的数据情况下，求使得已有数据的概率最大的目标位置。公式表达为，

$$\begin{aligned} p\left(\hat{d}_1, \hat{d}_2, \ldots, \hat{d_N} \mid \mathbf{x}\right) &= \left(2\pi\sigma^2\right)^{-N/2} \\ \times \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{N}\left(\hat{d}_i - \|\mathbf{x}_i - \mathbf{x}\|\right)^2\right) \end{aligned} \tag{3-15}$$

目标位置为，

$$\hat{\mathbf{x}} = \arg\max_{x} P\left(\hat{\mathbf{d}}|\mathbf{x}\right) \tag{3-16}$$

为求得目标位置 $\mathbf{x}$, 对公式3-15求导，最后等价于

$$\begin{aligned} \sum_{i=1}^{N}\frac{\left(d_i - \hat{d}_i\right)(x - x_i)}{d_i} &= 0 \\ \sum_{i=1}^{N}\frac{\left(d_i - \hat{d}_i\right)(y - y_i)}{d_i} &= 0 \end{aligned} \tag{3-17}$$

将原问题转化为求方程组3-17的解。关于求方程组的解的问题，由以下步骤组成。首先因为原方程需要实际距离 d_i，而这由目标的真实位置决定，因此该方程组需要近似求解。文献 [20] 提出了一种求解方法，利用平方差公式，

$$d_i - \hat{d}_i = \frac{{d_i}^2 - {\hat{d}_i}^2}{d_i + \hat{d}_i} \tag{3-18}$$

代入后如下，

$$\begin{aligned}\sum_{i=1}^{N}\frac{\left(d_i^2-\hat{d}_i^2\right)(x-x_i)}{d_i\left(d_i+\hat{d}_i\right)}=0\\ \sum_{i=1}^{N}\frac{\left(d_i^2-\hat{d}_i^2\right)(y-y_i)}{d_i\left(d_i+\hat{d}_i\right)}=0\end{aligned} \tag{3-19}$$

转化为矩阵等式，

$$2\begin{bmatrix}\sum g_ix_i & \sum g_iy_i\\ \sum h_ix_i & \sum h_iy_i\end{bmatrix}\begin{bmatrix}x\\ y\end{bmatrix}=\begin{bmatrix}\sum g_i\left(s+k_i-\hat{d}_i^2\right)\\ \sum h_i\left(s+k_i-\hat{d}_i^2\right)\end{bmatrix} \tag{3-20}$$

其中，

$$g_i=\frac{x-x_i}{d_i\left(d_i+\hat{d}_i\right)},\quad h_i=\frac{y-y_i}{d_i\left(d_i+\hat{d}_i\right)} \tag{3-21}$$

因为未知参数 x,y 同样出现在了 g_i,h_i,s 中，所以并不能直接进行最小二乘求解。因此，该方法将 g_i,h_i,s 视为已知参数，从而公式3-20可以视为线性公式。而要将 g_i,h_i,s 视为已知，需要代入初始 x,y 的值。初始值可由公式3-11求解。在求得初始值后，代入公式3-20中求出 g_i,h_i。

在求出 g_i,h_i 后，可以利用最小二乘公式求出用 s 表示的 x,y 值，将其代入公式2-9中。接下来利用选根方法：如果只有一个根为正，则该根即为解。如果两个均为正，则代入公式3-15中选择概率大的根为解。如果两个解均为负值或者复数，则取实部的绝对值。在迭代 q 次后，选取概率最大的值为预测值。

综上，最大似然迭代法的步骤为，通过仿真可知，最大似然迭代法的误差较小，这

算法 3-2　Maximum Likelihood Iteration

1) 利用公式3-11求出初始位置 (x_0,y_0)；
2) 将初始位置 (x_0,y_0) 代入3-21中，求出 g_i,h_i 的值；
3) 将 g_i,h_i,s 代入3-21中，求出 g_i,h_i,s 的值；
4) 求出公式3-20的最小二乘解；
5) 根据选根准则选择正确的位置；
6) 迭代 q 次后选择概率最大的值为预测值。

可以由该算法利用了噪声一致分布的先验条件解释，相比于高斯牛顿迭代法，该算法有更好的效果，当实际应用中噪声一致分布时可以考虑应用最大似然迭代法。

3.1.3 解析法

解析法又称为分析法，它是应用解析式去求解数学模型的方法。数学中用解析式表示函数或任意数学对象的方法叫解析法。这是一种直接求解的方法，相比于迭代法通常不需要重复计算，但是有可能出现无法求解或计算复杂度大的情况。当预测的未知参数概率分布未知时，用最小二乘法表示目标函数是非常有效的方法。对于 LOS 场景下的 TOA 问题，最小二乘法如下。

$$\begin{aligned}\hat{\mathbf{x}} &= \arg\min_{\mathbf{x}} \{R_{es}(\mathbf{x})\} \\ &= \arg\min_{\mathbf{x}} \left\{ \sum_{i=1}^{N} \beta_i \left(\hat{d}_i - \|\mathbf{x} - \mathbf{x}_i\| \right)^2 \right\}\end{aligned} \tag{3-22}$$

将3-22改写为矩阵形式，如下

$$\mathbf{A}_1 \boldsymbol{\theta} = \frac{1}{2}\mathbf{p}_1 \tag{3-23}$$

其中,

$$\mathbf{A}_1 = \begin{bmatrix} x_1 & y_1 & -0.5 \\ x_2 & y_2 & -0.5 \\ \vdots & \vdots & \\ x_N & y_N & -0.5 \end{bmatrix} \tag{3-24}$$

$$\boldsymbol{\theta} = \begin{bmatrix} x \\ y \\ s \end{bmatrix}, \mathbf{p}_1 = \begin{bmatrix} k_1 - \hat{d}_1^2 \\ k_2 - \hat{d}_2^2 \\ \vdots \\ k_n - \hat{d}_N^2 \end{bmatrix} \tag{3-25}$$

且 $s = x^2 + y^2$ 同样是未知的，因为包含 MT 位置 $\mathbf{x}$ 的二次项，所以需要选根，但这是不准确的。因此本文引入了近似线性最小二乘法 [21]。

1) LLS-1

因为本文有 N 个基站，所以共有 N 个2-8等式。本文用剩余的 N-1 个等式减去第 r 个等式，获得如下矩阵表达式，

$$\mathbf{A}_2 \boldsymbol{x} = \frac{1}{2}\mathbf{p}_2, \tag{3-26}$$

其中，

$$\mathbf{A}_2 = \begin{bmatrix} x_1 - x_r & y_1 - y_r \\ x_2 - x_r & y_2 - y_r \\ \vdots & \vdots \\ x_N - x_r & y_N - y_r \end{bmatrix}, \mathbf{p}_2 = \begin{bmatrix} \hat{d}_r^2 - \hat{d}_1^2 - k_{r,1} \\ \hat{d}_r^2 - \hat{d}_2^2 - k_{r,2} \\ \vdots \\ \hat{d}_r^2 - \hat{d}_N^2 - k_{r,N} \end{bmatrix} \tag{3-27}$$

上式 LS 的解析解为，

$$\mathbf{x} = \frac{1}{2}\left(\mathbf{A}_2^T \mathbf{A}_2\right)^{-1} \mathbf{A}_2^T \mathbf{p}_2 \tag{3-28}$$

注意到，2-8表示的几何意义是以基站为圆心，以测量距离为半径的圆，如图2-1所示。而相减后的等式则表示两个圆交点所确定的直线，如图3-3。 为了获得对 LLS 的更深

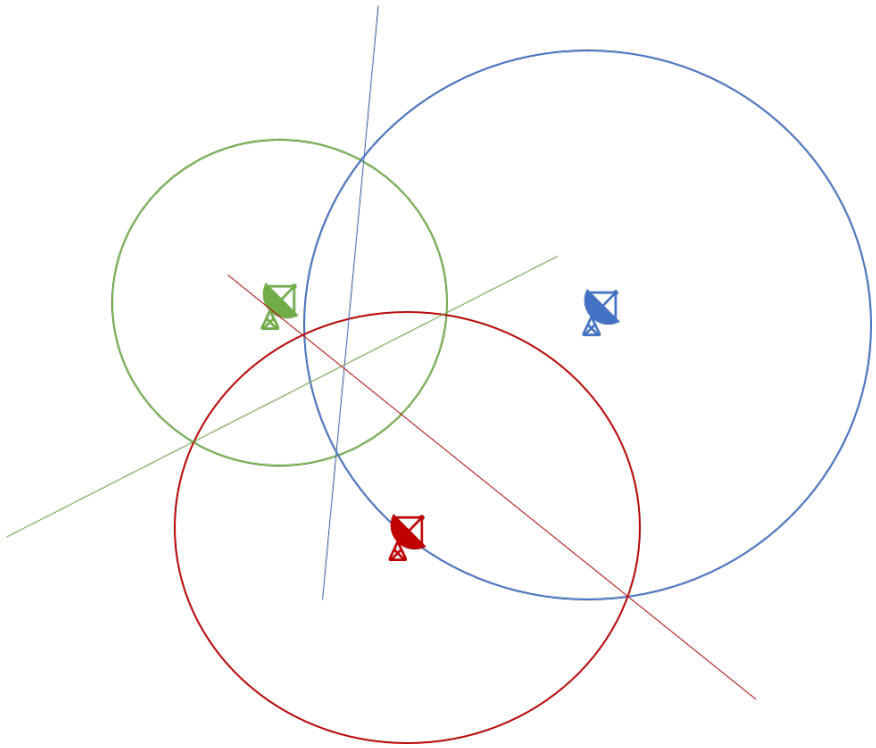


图 3-3　线性最小二乘示意图

入的理解，本文将 $\mathbf{p}_2$ 分解为,

$$\mathbf{p}_2 = \mathbf{p}_2^{(c)} + \mathbf{p}_2^{(n)} \tag{3-29}$$

将2-1代入 $\mathbf{p}_2$ 后, 得到下式，

$$\mathbf{p}_2^{(c)} = \begin{bmatrix} d_r^2 - d_1^2 - k_r + k_1 \\ d_r^2 - d_2^2 - k_r + k_2 \\ \vdots \\ d_r^2 - d_N^2 - k_r + k_N \end{bmatrix} \tag{3-30}$$

$$\mathbf{p}_2^{(n)} = \begin{bmatrix} 2d_r n_r - 2d_1 n_1 + n_r^2 - n_1^2 \\ 2d_r n_r - 2d_2 n_2 + n_r^2 - n_2^2 \\ \vdots \\ 2d_r n_r - 2d_N n_N + n_r^2 - n_N^2 \end{bmatrix} \tag{3-31}$$

由3-31可以看出，增加的噪声与测量距离有关，因此为了提高 LLS 的预测效果，有了接下来的改进。

2) LLS-Average

第一种改进是求测量距离的平均，得到平均后的公式，然后所有公式减去该公式进行求解，可以表示为，

$$\bar{d}_i = \hat{d}_i^2 - \frac{1}{N}\sum_{j=1}^{N} \hat{d}_j^2, \quad i = 1, 2, \ldots, N \tag{3-32}$$

3) LLS-RS

同理，因为误差与测量距离有关，第二种改进方案可以选择减去测量距离最短的基站，该基站的下标为。

$$r = \arg\min_i \left\{ \hat{d}_i \right\}, i = 1, 2, ...N \tag{3-33}$$

通过仿真发现，LLS-RS 和 LLS-Average 比 LLS-1 效果略好。具体分析在**仿真展示部分**。

4) SR-LS 法

另一种应用解析法的方法是对测量范围的平方使用最小二乘法，使其作为目标函数，如下，

$$(\mathrm{SR-LS}): \quad \min_{\mathbf{x}} \sum_{i=1}^{m} \left(\|\mathbf{x} - \mathbf{x}_i\|^2 - \hat{d}_i^{\,2} \right)^2 \tag{3-34}$$

目标函数3-34是非凸的，因此可以应用半正定法进行处理。首先将原式写为带约束最优化问题，

$$\min_{\mathbf{x}\in\mathbb{R}^n, \alpha\in\mathbb{R}} \left\{ \sum_{i=1}^{m} \left(\alpha - 2\mathbf{a}_i^T\mathbf{x} + \|\mathbf{a}_i\|^2 - r_i^2 \right)^2 : \|\mathbf{x}\|^2 = \alpha \right\} \tag{3-35}$$

令 $\mathbf{y} = (\mathbf{x}^T, \alpha)^T$, 上式可以重写为，

$$\min_{\mathbf{y}\in\mathbb{R}^{n+1}} \left\{ \|\mathbf{A}\mathbf{y} - \mathbf{b}\|^2 : \mathbf{y}^T\mathbf{D}\mathbf{y} + 2\mathbf{f}^T\mathbf{y} = 0 \right\} \tag{3-36}$$

其中，

$$\mathbf{A}=\begin{pmatrix}-2\mathbf{a}_1^T & 1\\ \vdots & \vdots\\ -2\mathbf{a}_m^T & 1\end{pmatrix},\mathbf{b}\ =\begin{pmatrix}r_1^2-\|\mathbf{a}_1\|^2\\ \vdots\\ r_m^2-\|\mathbf{a}_m\|^2\end{pmatrix} \tag{3-37}$$

且，

$$\mathbf{D}=\begin{pmatrix}\mathbf{I}_n & \mathbf{0}_{n\times 1}\\ \mathbf{0}_{1\times n} & 0\end{pmatrix},\mathbf{f}\ =\begin{pmatrix}\mathbf{0}\\ -0.5\end{pmatrix} \tag{3-38}$$

本文假设 $\mathbf{A}$ 是行满秩, 这意味着 $\mathbf{A}^T\mathbf{A}$ 是非奇异矩阵。

公式3-36只是重写了原式仍然是非凸的，现有方法只能求解次优解。一种方法是直接舍弃二次约束条件，这样可以直接由最小二乘公式求得 $\mathbf{y}^*=(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$，而对应的预测位置 $\mathbf{x}$ 则为 $\mathbf{y}^*$ 前两项元素。

文献 [12] 提出了另一种准确求解 SR-LS 的方法。公式3-36是约束条件为二次约束，目标函数为二次函数的最优化问题，这类问题被称作广义信任域子问题 (GTRS)，而 GTRS 问题往往可以通过有效的方法求得最优解。具体来说，当且仅当存在 $\lambda\in R$ 满足如下条件时，$\mathbf{y}$ 是最优解，

$$\begin{aligned}\left(\mathbf{A}^T\mathbf{A}+\lambda\mathbf{D}\right)\mathbf{y}&=\mathbf{A}^T\mathbf{b}-\lambda\mathbf{f}\\ \mathbf{y}^T\mathbf{D}\mathbf{y}+2\mathbf{f}^T\mathbf{y}&=0\\ \mathbf{A}^T\mathbf{A}+\lambda\mathbf{D}&\succeq\mathbf{0}.\end{aligned} \tag{3-39}$$

该问题的最优解由下式给出，

$$\hat{\mathbf{y}}(\lambda)=\left(\mathbf{A}^T\mathbf{A}+\lambda\mathbf{D}\right)^{-1}\left(\mathbf{A}^T\mathbf{b}-\lambda\mathbf{f}\right) \tag{3-40}$$

而 λ 是下式的解，

$$\varphi(\lambda)=0,\quad \lambda\in I \tag{3-41}$$

而且函数 φ 由下式

$$\varphi(\lambda)\equiv\hat{\mathbf{y}}(\lambda)^T\mathbf{D}\hat{\mathbf{y}}(\lambda)+2\mathbf{f}^T\hat{\mathbf{y}}(\lambda) \tag{3-42}$$

使得 $\mathbf{A}^\mathbf{T}\mathbf{A}+\lambda\mathbf{D}$ 半正定的 λ 的取值范围为，

$$I=\left(-\frac{1}{\lambda_1\left(\mathbf{D},\mathbf{A}^T\mathbf{A}\right)},\infty\right) \tag{3-43}$$

其中，$\lambda_i\left(\mathbf{A},\mathbf{B}\right)$ 表示 $\lambda_i\left(\mathbf{B}^{-1/2},\mathbf{A}\mathbf{B}^{-1/2}\right),i=1,...,n$ 且 $\lambda_i(M)$ 表示矩阵 $\mathbf{M}$ 的第 i 个特征值（按递减顺序）。

同时，由定理可知，$\Phi(\lambda)$ 在区间 I 上是严格递减的，因此可以使用二分法来求最优 λ。综上，求解 SR-LS 的步骤如下，在实际复现时，会出现许多问题，本文重点介

算法 3-3 Gauss-Newton Iteration

1) 求出 λ 的取值范围 I；

2) 使用二分法求出最优 λ；

3) 将最优 λ 代入3-40求得 **y**, 预测位置 **x** 为 **y** 前两项。

绍如下几点，

1) 在求解最佳 λ 过程中，会出现 $\mathbf{A^T A} + \lambda \mathbf{D}$ 为奇异矩阵的情况导致程序报错，因此需要单独处理，直接令3-42为无穷；

2) 在求解 λ 过程中，可能出现 $\lambda = 0$ 的情况这时应修改 λ 为一个极小值，避免除以 0 报错；

3) 二分法需要在有限集合，因此上限实际不能设置为正无穷，应先求得下限，再不断扩大上限至函数3-42值为负；

4) 可以采用插值法改进步骤 b) 的二分法提高查找效率。

3.1.4 半正定法

半正定规划是凸优化问题的一个分支，它具有线性目标函数（由用户指定的最大化或最小化函数），且其定义在半正定矩阵构成的凸锥与仿射空间的交集上，即光谱面。而许多非凸二次规划的问题往往可以通过半正定松弛的方法转化为凸优化且具有较优解，因此半正定规划的应用范围非常广泛。

半正定规划问题通常表示为，

$$\begin{aligned} &\min_{x^1,\ldots,x^n \in \mathbb{R}^n} \sum_{i,j=1}^{n} c_{i,j} \left(x^i \cdot x^j\right) \\ &s.t. \sum_{i,j=1}^{n} a_{i,j,k} \left(x^i \cdot x^j\right) \leq b_k, \quad k = 1, \ldots, m \end{aligned} \tag{3-44}$$

其中 $c_{i,j}, a_{i,j,k}, b_k$ 是实数，$x^i \cdot x^j$ 是 x^i 和 x^j 的内积，s.t. 后面是需满足的限制式。

从上式看不出半正定这个名称从何而来。事实上，从下面的等价描述会发现，半正定这个词来自于将线性规划中的实变数的非负限制式改为矩阵变数的半正定限制式。将 S^n 设为收集所有对称矩阵所形成的空间，并且在空间上赋予内积，

$$\langle A, B \rangle_{\mathbb{S}^n} = \operatorname{tr}\left(A^T B\right) = \sum_{i,j=1}^{n} A_{ij} B_{ij} \tag{3-45}$$

其中 tr 代表矩阵的迹。因此上述的半正定规划可以改写成，

$$\begin{aligned}&\min_{X\in\mathbb{S}^n}\langle C,X\rangle_{\mathbb{S}^n}\\&s.t.\ \langle A_k,X\rangle_{\mathbb{S}^n}\le b_k,\quad k=1,\dots,m\\&\quad X\succeq 0\end{aligned}\tag{3-46}$$

对于 LOS 场景下 TOA 问题的具体应用，本文介绍了半正定算法 R-LS[12]。

1) R-LS 法

对于 TOA 问题，目标函数3-60是非凸函数，因此很难求得准确解。基于这个原因，文献 [12] 提出利用半正定松弛法将 TOA 问题转化为半正定规划问题，如下。
首先将公式3-60重写为带约束最优解问题，

$$\begin{aligned}&\min_{\mathbf{x},\mathbf{g}}\sum_{i=1}^{m}\left(\hat{d}_i-g_i\right)^2\\&s.t.g_i^2=\|\mathbf{x}-\mathbf{a}_i\|^2,\quad i=1,\dots,m\end{aligned}\tag{3-47}$$

然后将未知向量转化为未知矩阵,

$$\mathbf{G}=\begin{pmatrix}\mathbf{g}\\1\end{pmatrix}\begin{pmatrix}\mathbf{g}^T & 1\end{pmatrix},\mathbf{X}=\begin{pmatrix}\mathbf{x}\\1\end{pmatrix}\begin{pmatrix}\mathbf{x}^T & 1\end{pmatrix}\tag{3-48}$$

向量与其转置相乘产生的矩阵一定是秩为一的半正定矩阵，因此之后用 $\mathbf{G},\mathbf{X}$ 重写公式3-47,

$$\begin{aligned}&\min_{\mathbf{X},\mathbf{G}}\sum_{i=1}^{m}\left(G_{ii}-2r_iG_{m+1,i}+r_i^2\right)\\&\text{s.t. }G_{ii}=\mathrm{Tr}\left(\mathbf{C}_i\mathbf{X}\right),\quad i=1,\dots,m\\&\mathbf{G}\succeq\mathbf{0},\mathbf{X}\succeq\mathbf{0}\\&G_{m+1,m+1}=X_{n+1,n+1}=1\\&\mathrm{rank}(\mathbf{X})=\mathrm{rank}(\mathbf{G})=1\end{aligned}\tag{3-49}$$

其中，

$$\mathbf{C}_i=\begin{pmatrix}\mathbf{I} & -\mathbf{a}_i\\-\mathbf{a}_i^T & \|\mathbf{a}_i\|^2\end{pmatrix},\quad i\ =1,\dots,m.\tag{3-50}$$

因为 $rank(X) = rank(G) = 1$ 为非凸约束，因此舍弃，即为半正定松弛。
最后的优化问题为，

$$\begin{aligned} \min_{\mathbf{X},\mathbf{G}} \sum_{i=1}^{m} \left(G_{ii} - 2r_i G_{m+1,i} + r_i^2\right) \\ \text{s.t. } G_{ii} = \mathrm{Tr}\left(\mathbf{C}_i \mathbf{X}\right), \quad i = 1, \ldots, m \\ \mathbf{G} \succeq \mathbf{0}, \mathbf{X} \succeq \mathbf{0} \\ G_{m+1,m+1} = X_{n+1,n+1} = 1. \end{aligned} \tag{3-51}$$

SDR 问题可以通过内点法进行求解，当且仅当最终求得的 G, X 秩为一时，所求最优解与原最优解重合。

3.2 基于 NLOS 场景的算法研究

在典型环境中，尤其是在室内场景中，MT 和某些 FT 之间的 LOS 可能是被阻塞的（即对于某些 i，$b_i > 0$）。这些 NLOS 的 FTs 可能会严重降低定位准确性。NLOS 缓解的最简单方法是通过识别和丢弃 NLOS 的 FTs，并通过使用一种上一节提到的 LOS 技术定位 MT 的位置。然而，事实上总有可能误报（将 LOS 的 FT 识别为 NLOS）和漏检（将 NLOS FTs 识别为 LOS），而这会降低定位精度。在本节中，首先介绍了 NLOS 场景下的 CRLB，然后介绍了两种处理 NLOS 的 FTs 的方法。

3.2.1 NLOS-CRLB

在 NLOS 场景中，CRLB 取决于是否存在有关 NLOS 偏差的任何先验信息。首先本文认为没有关于 NLOS 的先验信息偏差，除了假设 NLOS 的 FTs 是已知的。然后公式3-3中的 FIM 的扩展版本由 [22] 给出，

$$\mathbf{I}\left(\mathbf{x}_b\right) = \tilde{\mathbf{A}}\mathbf{I}(\mathbf{d})\tilde{\mathbf{A}}^T \tag{3-52}$$

其中，

$$\mathbf{x}_b = \left[x, y, b_1, b_2, \ldots b_{N_{NL}}\right]^T \tag{3-53}$$

是一个 $(N_{NL} + 2)$×1 的未知参数向量（包括 NLOS 偏差值）且，

$$\begin{aligned} \tilde{\mathbf{A}} &= \frac{\partial \mathbf{d}}{\partial \mathbf{x}_b} \\ &= \begin{bmatrix} \frac{\partial d_1}{\partial x} & \frac{\partial d_2}{\partial x} & \cdots & \frac{\partial d_{N_{NL}}}{\partial x} & \cdots & \frac{\partial d_N}{\partial x} \\ \frac{\partial d_1}{\partial y} & \frac{\partial d_2}{\partial y} & \cdots & \frac{\partial d_{NL}}{\partial y} & \cdots & \frac{\partial d_N}{\partial y} \\ \frac{\partial d_1}{\partial b_1} & \frac{\partial d_2}{\partial b_1} & \cdots & \frac{\partial d_{NL}}{\partial b_1} & \cdots & \frac{\partial d d_N}{\partial b_1} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial d_1}{\partial b_{N_{NL}}} & \frac{\partial d_2}{\partial b_{N_{NL}}} & \cdots & \frac{\partial d_{N_{NL}}}{\partial b_{N_{NL}}} & \cdots & \frac{\partial d_N}{\partial b_{N_{NL}}} \end{bmatrix} \end{aligned} \tag{3-54}$$

并且，

$$\mathbf{I}(\mathbf{d}) = \mathrm{E}_{\mathbf{d}}\left[\frac{\partial \ln f(\hat{\mathbf{d}} \mid \mathbf{d})}{\partial \mathbf{d}}\left(\frac{\partial \ln f(\hat{\mathbf{d}} \mid \mathbf{d})}{\partial \mathbf{d}}\right)^{T}\right] \tag{3-55}$$

正如在 [22] 中讨论的，$\tilde{\mathbf{A}}$ 可以写成由 LOS 和 NLOS 组成的形式，

$$\tilde{\mathbf{A}} = \begin{bmatrix} \tilde{\mathbf{A}}_{NL} & \tilde{\mathbf{A}}_{L} \\ \mathbf{I}_{N_{NL}} & \mathbf{0}_{N_{NL},N_L} \end{bmatrix} \tag{3-56}$$

类似地，$I(d)$ 可以写成它的 NLOS 和 LOS 组成如 [22]，

$$\mathbf{I}(\mathbf{d}) = \begin{bmatrix} \Lambda_{NL} & \mathbf{0} \\ \mathbf{0} & \Lambda_{L} \end{bmatrix} \tag{3-57}$$

其中，$\Lambda_{NL} = diag(\sigma_1^{-2}, ..., \sigma_N^{-2})$ 且 $\Lambda_L = diag(\sigma_{N_{NL+1}}^{-2}, ..., \sigma_{N_N}^{-2})$，最终 $I(x_b)$ 由下式得到，

$$\mathbf{I}\left(\mathbf{x}_b\right) = \begin{bmatrix} \tilde{\mathbf{A}}_{NL}\mathbf{\Lambda}_{NL}\tilde{\mathbf{A}}_{NL}^{T} + \tilde{\mathbf{A}}_{L}\mathbf{\Lambda}_{L}\tilde{\mathbf{A}}_{L}^{T} & \tilde{\mathbf{A}}_{NL}\mathbf{\Lambda}_{NL} \\ \mathbf{\Lambda}_{NL}\mathbf{A}_{NL}^{T} & \mathbf{\Lambda}_{NL} \end{bmatrix} \tag{3-58}$$

而最终的 CRLB 由下式得到，

$$E\left[\left(\hat{\mathbf{x}}_b - \mathbf{x}_b\right)\left(\hat{\mathbf{x}}_b - \mathbf{x}_b\right)^{T}\right] \geq \mathbf{I}^{-1}\left(\mathbf{x}_b\right) = \left(\tilde{\mathbf{A}}_{L}\mathbf{\Lambda}_{L}\tilde{\mathbf{A}}_{L}^{T}\right)^{-1} \tag{3-59}$$

3.2.2 识别丢弃法

首先，为了对比 NLOS 的 FT 影响的效果，本部分先讨论没有进行 NLOS 处理的情况下最小二乘法的效果。这当然是有意义的，因为很可能存在全部都是 NLOS 基站 FTs 的情况，这种情况下，不能简单地进行舍弃或者加权，而只能用 LOS 的方法进行求解。

$$\begin{aligned} \hat{\mathbf{x}} &= \arg\min_{\mathbf{x}}\left\{R_{es}(\mathbf{x})\right\} \\ &= \arg\min_{\mathbf{x}}\left\{\sum_{i=1}^{N}\left(\hat{d}_i - \|\mathbf{x} - \mathbf{x}_i\|\right)^2\right\} \end{aligned} \tag{3-60}$$

本文利用上式对预测位置进行求解，可以预测到，与全部都是 LOS 的情况相比，NLOS 情况下的 LLS 的效果要差。本文使用蒙特卡洛方法运行 100 次，在同样场景下固定基站位置和种类，仅用 LOS 的 FTs 和包括 NLOS 的 FTs 分别进行最小二乘预测，如
由图3-4可以看出，无论怎样减小 NLOS 的 FTs 对预测结果的影响，在有足够的 LOS 的 FTs 的情况下都不如直接舍弃所有的 NLOS 的 FTs 来的有效，如何正确识别并舍弃 NLOS 基站是消除影响的关键。

文献 [13] 提出残差法，从结果倒推可能的 LOS 组合。主要思想是有 NLOS 的预测

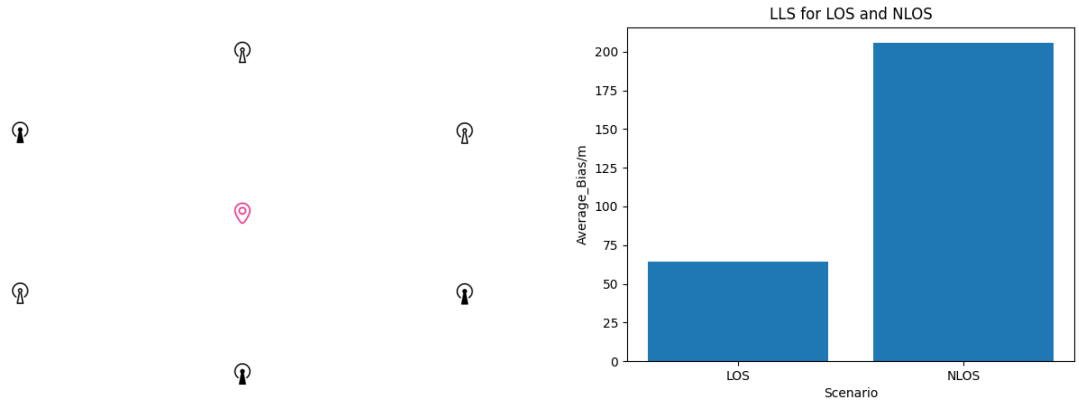


图 3-4　最小二乘在 NLOS 场景和 LOS 场景对比

结果要比没有 NLOS 的预测结果差，因此使用最小二乘法进行预测。从 N 个基站中选出所有的 N-1 个组合，分别进行预测获得目标位置 $\hat{x}$，选择预测效果最好（即残差最小）的组合，如下，

$$Res(S_k) = \frac{\sum_{i=1}^{N-1} \|\hat{\mathbf{x}} - \mathbf{x_i}\|}{N-1} \tag{3-61}$$

将剩余的一个基站舍弃。重复上述步骤，直到仅剩三个基站（预测位置要求的最小数量）。最后从所有组合中选择残差最小的一组作为最后的结果。

显然该方法是以时间为代价来提高预测的准确度。具体步骤如下，

算法 3-4　Gauss-Newton Iteration

1) 从 N 个基站中选出所有 N-1 个基站的组合，分别用最小二乘法进行预测，获得预测位置 $\hat{x}$ 后，利用公式3-61进行比较，选取残差最小的一组；

2) 重复上述步骤直到预设的终止条件；

3) 选择残差最小的一组的预测结果作为最终结果。

3.2.3　鲁棒预测器

1)　Huber-M 预测器

Huber-M 预测器是 ML 类型的预测器，是考虑用于缓解 NLOS 影响的稳健估计器。如前几节所述，ML 算法试图最大化下述形式的函数,

$$\prod_{i=1}^{N} f(x_i) \tag{3-62}$$

等价于最小化 $\sum_{i=1}^{N} -logf(x_i)$。当有异常值出现时，ML 算法不能准确预测，因此 Huber 于 1964 引入了下述形式的函数用以生成一个 ML 算法的普遍形式，被称作 M 预测器

$$\sum_{i=1}^{N} \rho(x_i) \tag{3-63}$$

其中，$\rho(.)$ 是凸函数，对于 Huber-M 预测器，$\rho(.)$ 被定义为，

$$\rho(\nu) = \begin{cases} \nu^2/2 & |\nu| \leq \xi \\ \xi|\nu| - \xi^2/2 & |\nu| > \xi \end{cases} \tag{3-64}$$

它不是严格凸的，因此，最小化目标函数产生多个相近的解。在文献 [14] 中，M 预测器用来预测 MT 的位置，产生如下公式，

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \left\{ \sum_{i=1}^{N} \rho\left(\left(\hat{d}_i - \|\mathbf{x} - \mathbf{x}_i\| \right) / \sigma_i \right) \right\} \tag{3-65}$$

2) 最小中位数平方

在文献 [9] 中，最小中值二乘 (LMS) 技术被用于 NLOS 缓解，这是最常用的鲁棒拟合算法。它可以容忍在没有噪声的情况下高达 50% 的异常值。使用 LMS 解决方案的 MT 的预测位置由文献 [9] 给出，

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \left\{ med_i \left(\hat{d}_i - \|\mathbf{x} - \mathbf{x}_i\| \right)^2 \right\} \tag{3-66}$$

其中 $med_i(\theta(i))$ 是所有可能的 $\theta(i)$ 的中位数 i 的值。

由于公式3-66是计算密集的，文献 [9] 提出了一个较低复杂度的实现。它使用 x_i 的随机子集来获得几个候选 $\hat{x}$，并且选择残差中位数最小的那个作为解决方案。需要注意的是，在安全场景下 [9] 使用 LMS 算法，其中一些测量可能在存在对本地化的某些攻击时成为异常值系统。

3.3 基于初始时间场景的算法研究

在实际应用中，影响 TOA 问题测量精度的因素有很多，往往不止是视线与非视线的影响。例如，很多情况下移动站 MT 和基站 FT 以及基站 FT 与基站 FT 之间不一定是同步的，这意味着接收站点往往并不知道信号发送的起始时间，即最后获得的到达时间测量值还包括初始时间误差。为方便分析初始时间的影响，本文假设没有非视线 NLOS 误差。测量量 t_i，

$$t_i = \frac{1}{c}\|\mathbf{x_i} - \mathbf{y}\| + t_0 + n_i, i = 1, 2, ..., N \tag{3-67}$$

其中，c 是光速，t_0 是未知的初始时间，n_i 是噪声。
将其转化为距离形式如下，

$$d_i = \|\mathbf{x_i} - \mathbf{y}\| + d_0 + n_i, i = 1, 2, ..., N \tag{3-68}$$

3.3.1 最小二乘法

在一些现有方法中，往往采用等式相减的方法抵消初始时间 t_0 的影响，而这会导致信噪比有 3dB 的损失。在不做任何处理的情况下，本文使用最小二乘法，将初始时间视为未知参数，可以得到下式，

$$\left(\hat{\mathbf{y}}, \hat{d_0}\right) = \arg\min_{\mathbf{y}, d_0} \sum_{i=1}^{N} \left(d_i - \|\mathbf{x}_i - \mathbf{y}\| - d_0\right)^2 \tag{3-69}$$

将其转换为矩阵形式，

$$\mathbf{A}_3 \boldsymbol{x} = \frac{1}{2}\mathbf{p}_3 \tag{3-70}$$

其中 $\mathbf{x} = [x, y, d_0]^T$ 且，

$$\mathbf{p}_3 = \begin{bmatrix} x_1^2 - x_r^2 + y_1^2 - y_r^2 + d_r^2 - d_1^2 \\ x_2^2 - x_r^2 + y_2^2 - y_r^2 + d_r^2 - d_2^2 \\ \vdots \\ x_N^2 - x_r^2 + y_N^2 - y_r^2 + d_r^2 - d_N^2 \end{bmatrix} \tag{3-71}$$

$$\mathbf{A}_3 = \begin{bmatrix} 2\left(x_1 - x_r\right) & 2\left(y_1 - y_r\right) & 2\left(d_r - d_1\right) \\ 2\left(x_2 - x_r\right) & 2\left(y_2 - y_r\right) & 2\left(d_r - d_2\right) \\ \vdots & \vdots & \vdots \\ 2\left(x_N - x_r\right) & 2\left(y_N - y_r\right) & 2\left(d_r - d_N\right) \end{bmatrix}, \tag{3-72}$$

最后由最小二乘求解公式得到预测位置，

$$\mathbf{x} = \frac{1}{2}\left(\mathbf{A}_3^T \mathbf{A}_3\right)^{-1} \mathbf{A}_3^T \mathbf{p}_3 \tag{3-73}$$

3.3.2 半正定规划法

文献 [15] 提出了一种二步规划法。本文注意到，wLS 的预测要同时求出未知位置 $\mathbf{x}$ 和 t_0。二步规划法没有同时求解这个未知参数，而是分步求解。本文由公式3-67可以得知，初始时间与未知参数之间满足等式关系。具体来说，对于零均值的噪声，初始时间 t_0 可以表示为，

$$\hat{t}_0 = \frac{1}{N}\sum_{i=1}^{N}\left(t_i - \frac{1}{c}\|\mathbf{x}_i - \mathbf{y}\|\right) \tag{3-74}$$

本文把上式代入公式3-69中，可以得到下式，

$$\sum_{i=1}^{N}\left[t_i-\frac{1}{c}\left\|\mathbf{x}_i-\mathbf{y}\right\|-\frac{1}{N}\sum_{j=1}^{N}\left(t_j-\frac{1}{c}\left\|\mathbf{x}_j-\mathbf{y}\right\|\right)\right]^2 \tag{3-75}$$

目标函数是非凸的因此需要用半正定松弛法转换。

首先，引入辅助变量,

$$\tau_i=\frac{1}{c}\left\|\mathbf{x}_i-\mathbf{y}\right\|,\quad 1\le i\le N \tag{3-76}$$

同时，$\mathbf{t}=[t_1,t_2,....t_N]^T,\tau=[\tau_1,\tau_2,....\tau_N]^T,Q=\tau\tau^T,G=I-\mathbf{1}\mathbf{1}^T$，用这些符号重写目标函数如下

$$\mathrm{Tr}\left[G^TG\left(\mathbf{Q}-2t\tau^T+t^T\right)\right] \tag{3-77}$$

本文注意到上式描述的问题有两个约束条件，$\tau_i=\frac{1}{c}\left\|\mathbf{x}_i-\mathbf{y}\right\|$ 和 $Q=\tau\tau^T$。而这两个约束条件都是非凸的。因此，本文下一步的目标就是放松约束条件使问题变为凸优化的同时保持和原问题紧密相连。

首先考虑辅助变量 τ，注意到，

$$\left(\mathbf{x}_i-\mathbf{y}\right)^T\left(\mathbf{x}_j-\mathbf{y}\right)=\begin{bmatrix}\mathbf{x}_i\\-1\end{bmatrix}^T\begin{bmatrix}\mathbf{I}&\mathbf{y}\\\mathbf{y}^T&y_s\end{bmatrix}\begin{bmatrix}\mathbf{x}_j\\-\mathbf{1}\end{bmatrix} \tag{3-78}$$

其中，$y_s=\mathbf{y}^T\mathbf{y}$, 因此对于约束条件 Q，可以被写为，

$$\begin{aligned}\mathbf{Q}_{ii}=\tau_i^2&=\frac{1}{c^2}\left(\mathbf{x}_i-\mathbf{y}\right)^T\left(\mathbf{x}_i-\mathbf{y}\right)\\&=\frac{1}{c^2}\begin{bmatrix}\mathbf{x}_i\\-1\end{bmatrix}^T\begin{bmatrix}\mathbf{I}&\mathbf{y}\\\mathbf{y}^T&y_s\end{bmatrix}\begin{bmatrix}\mathbf{x}_i\\-1\end{bmatrix}\end{aligned} \tag{3-79}$$

现在这一约束条件是凸约束了，本文使用了变量 $Q,\mathbf{y},y_s$。紧接着，本文对于变量 Q 的其余元素，应用柯西不等式得，

$$\left\|\mathbf{x}_i-\mathbf{y}\right\|\left\|\left|\mathbf{x}_j-\mathbf{y}\right\|\right.\ge\left|\left(\mathbf{x}_i-\mathbf{y}\right)^T\left(\mathbf{x}_j-\mathbf{y}\right)\right| \tag{3-80}$$

因此，该约束条件可以写为，

$$\begin{aligned}\mathbf{Q}_{ij}=\tau_i\tau_j&=\frac{1}{c^2}\left\|\mathbf{x}_i-\mathbf{y}\right\|\left\|\mathbf{x}_j-\mathbf{y}\right\|\\&\ge\frac{1}{c^2}\left|\begin{bmatrix}\mathbf{x}_i\\-1\end{bmatrix}^T\begin{bmatrix}\mathbf{I}&\mathbf{y}\\\mathbf{y}^T&y_s\end{bmatrix}\begin{bmatrix}\mathbf{x}_j\\-1\end{bmatrix}\right|\end{aligned} \tag{3-81}$$

这一约束也是凸约束了。

本文现在仍有两个非线性非凸得约束条件 $Q=\tau\tau^T$ 和 $y_s=\mathbf{y}\mathbf{y}^T$，对于这两个约束，本

文使用半正定松弛是他们变为凸约束，他们可以被写为，

$$\begin{bmatrix} \mathbf{Q} & \tau \\ \tau^T & 1 \end{bmatrix} \succeq 0, \quad \begin{bmatrix} \mathbf{I} & \mathbf{y} \\ \mathbf{y}^T & y_s \end{bmatrix} \succeq 0 \tag{3-82}$$

本文现在把原问题转换为了一个凸优化问题，

$$\begin{aligned} &\min_{\mathbf{y}, y_s, \tau, \mathbf{Q}} \operatorname{Tr}\left[G^T G\left(\mathbf{Q} - 2t\tau^T + tt^T\right)\right] \\ &s.t. \begin{bmatrix} \mathbf{Q} & \tau \\ \tau^T & 1 \end{bmatrix} \succeq 0, \quad \begin{bmatrix} \mathbf{I} & \mathbf{y} \\ \mathbf{y}^T & y_s \end{bmatrix} \succeq 0 \\ &\mathbf{Q}_{ii} = \frac{1}{c^2} \begin{bmatrix} \mathbf{x}_i \\ -1 \end{bmatrix}^T \begin{bmatrix} \mathbf{I} & \mathbf{y} \\ \mathbf{y}^T & y_s \end{bmatrix} \begin{bmatrix} \mathbf{x}_i \\ -1 \end{bmatrix} \\ &\mathbf{Q}_{ij} \geq \frac{1}{c^2} \left| \begin{bmatrix} \mathbf{x}_i \\ -1 \end{bmatrix}^T \begin{bmatrix} \mathbf{I} & \mathbf{y} \\ \mathbf{y}^T & y_s \end{bmatrix} \begin{bmatrix} \mathbf{x}_j \\ -1 \end{bmatrix} \right| \end{aligned} \tag{3-83}$$

尽管已经转变为了凸优化问题，这个问题仍然没有准确解，因为当 $\frac{1}{c}\|\mathbf{x_i} - \mathbf{y}\|$ 增大而 t_0 减小或相反情况时，最小二乘法3-69的值不会改变。因此，本文需要增加一个惩罚函数去避免这种不确定性。本文引入了罚函数 $\eta \sum_{i=1}^{N} \sum_{j=1}^{N} Q_{ij}$ 且 $\eta > 0$ 是罚函数因子。

最后，该问题被写为，

$$\begin{aligned} &\min_{\mathbf{y}, y_s, \tau, \mathbf{Q}} \operatorname{Tr}\left[G^T G\left(\mathbf{Q} - 2t\tau^T + tt^T\right)\right] + \eta \sum_{i=1}^{N} \sum_{j=1}^{N} Q_{ij} \\ &s.t. \begin{bmatrix} \mathbf{Q} & \tau \\ \tau^T & 1 \end{bmatrix} \succeq 0, \quad \begin{bmatrix} \mathbf{I} & \mathbf{y} \\ \mathbf{y}^T & y_s \end{bmatrix} \succeq 0 \\ &\mathbf{Q}_{ii} = \frac{1}{c^2} \begin{bmatrix} \mathbf{x}_i \\ -1 \end{bmatrix}^T \begin{bmatrix} \mathbf{I} & \mathbf{y} \\ \mathbf{y}^T & y_s \end{bmatrix} \begin{bmatrix} \mathbf{x}_i \\ -1 \end{bmatrix} \\ &\mathbf{Q}_{ij} \geq \frac{1}{c^2} \left| \begin{bmatrix} \mathbf{x}_i \\ -1 \end{bmatrix}^T \begin{bmatrix} \mathbf{I} & \mathbf{y} \\ \mathbf{y}^T & y_s \end{bmatrix} \begin{bmatrix} \mathbf{x}_j \\ -1 \end{bmatrix} \right| \\ &i = 1, 2, ..., N, j = i + 1, ..., N \end{aligned} \tag{3-84}$$

公式 3-84可以用内点法进行求解，本篇文章使用 cvxpy 库的默认求解器求解。在公式 3-84中，η 的选择对预测结果有很大的影响，本文很自然的可以想到，权重系数应该和测量距离有关。因此，η 应该正比于测量距离的平均值，

$$\eta = \alpha \times \frac{1}{N} \sum_{i=1}^{N} t_i \tag{3-85}$$

而适当的 α 可以在仿真时由试验得出，本文会在仿真部分给出。

3.3.3 算法对比

半正定规划法的放缩非常紧密，因此效果比单纯的最小二乘法要好上许多，本篇文章通过蒙特卡洛仿真得出两种算法的均方根误差的平均值，如图3-5。

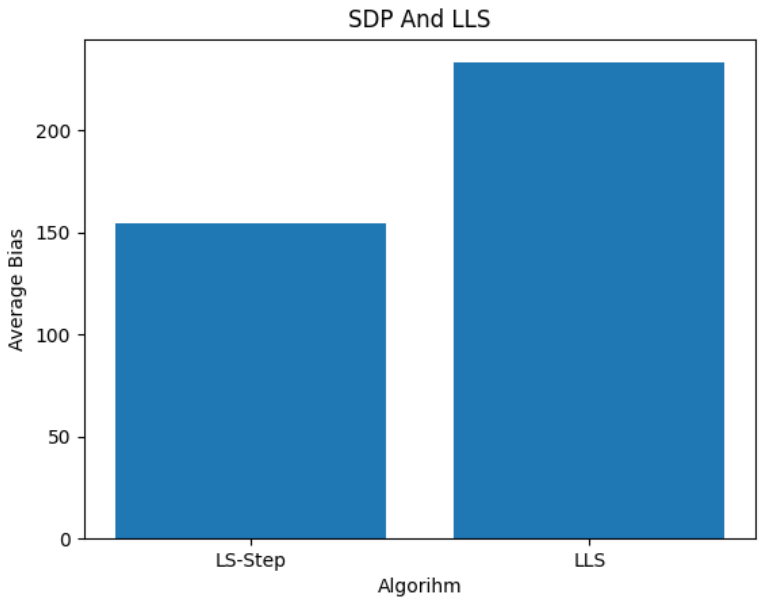
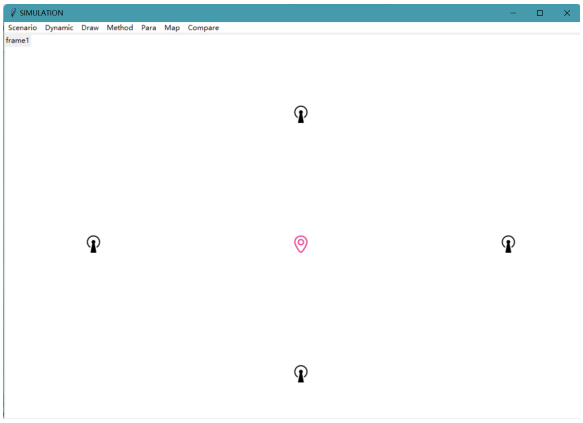


图 3-5 SDP 算法和 LLS 算法对比

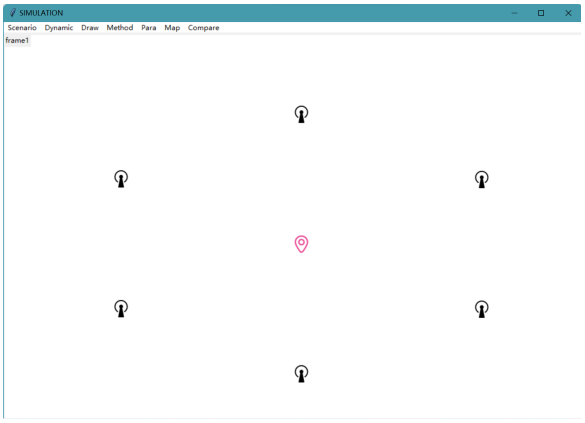
4 仿真分析

4.1 基站数目对预测误差的影响

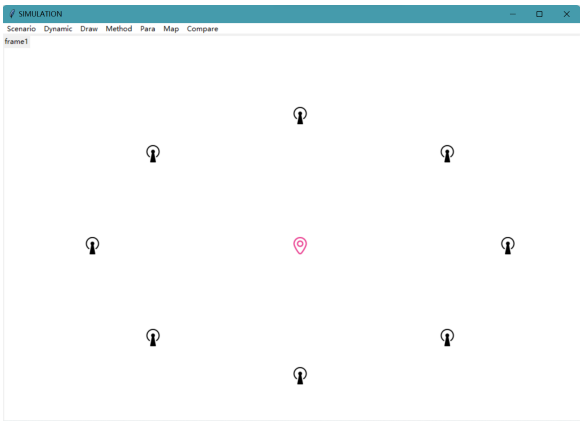
为了分析基站数目对预测结果的影响，我们考虑 LOS 场景下基站沿圆形分布时，不断增加基站数目，观察对应的预测误差如何变化。直观上，我们认为随着基站数目的增多，预测结果会逐渐变好，我们将基站按圆形分布不断增加基站数目，如图4-1。接着使用 LLS 算法每个数目场景下用蒙特卡洛仿真 10000 次测得平均误差，如图4-2。



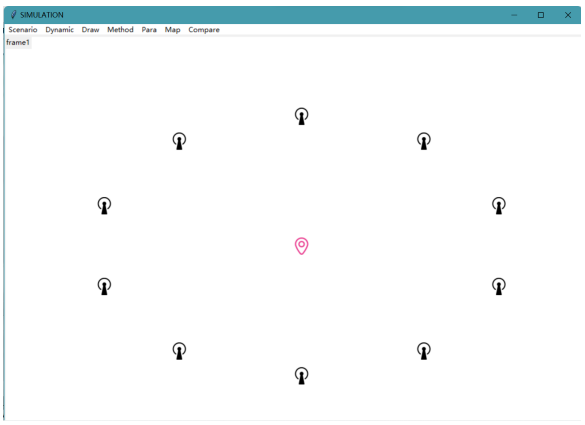
(a) 4



(b) 6



(c) 8



(d) 10

图 4-1 基站数目

由图4-2可以清楚的看出随着基站数目的增加，测量误差逐渐减小，这是因为算法可用的信息增多。

4.2 基站包围面积对预测误差的影响

我们接下来探讨，当预测的总面积不变时（即总共 $10000m^2$ 的面积），基站分布的包含面积对预测误差会有什么影响。当固定数目（11）基站围成的圆形半径逐渐扩

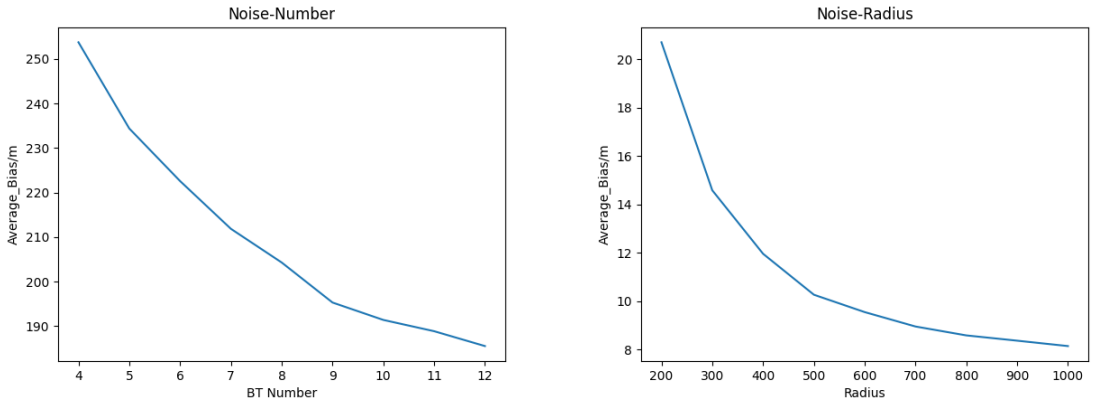


图 4-2　Noise-Number and Noise-radius

大时，如图4-3。我们继续通过蒙特卡洛进行 10000 次仿真进行对比，通过仿真得出误

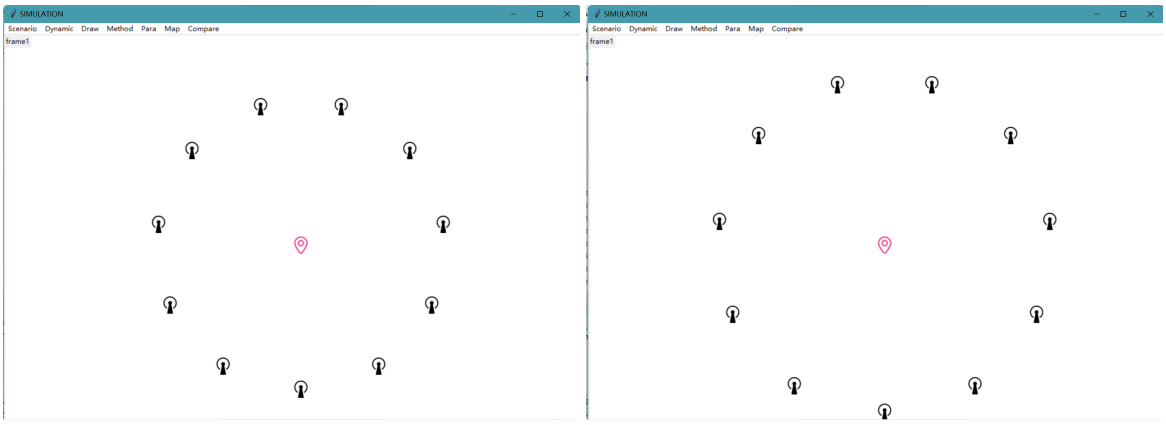


图 4-3　包含面积对比

差随圆半径增大的变化，如图4-2。由图4-2可以看出，随着半径增大预测误差在逐渐减小。这是因为测量误差的高斯分布与测量距离无关时，包含面积大时基站距离目标站点的平均距离增大，相应的测量误差占比下降，因此预测准确度上升。

4.3　目标基站在区域内外对预测误差的影响

为进一步分析，我们接着讨论目标位置处于基站分布区域内与分布区域外时，平均预测误差如何变化。如图4-4。

我们继续通过蒙特卡洛进行 10000 次仿真进行对比，通过仿真得出误差与目标位置在基站分布区域内外的关系。如图4-5。由图4-5可以看出，当目标基站处于内部区域时，预测误差相对要小。这是因为处于内侧时，各基站与目标位置的距离相差不多，每个基站都能提供可靠的信息不会出现距离过大的情况。同时，也能解释上部分基站区域面积的问题，当基站区域面积较大时，目标位置更有可能落入基站区域内。

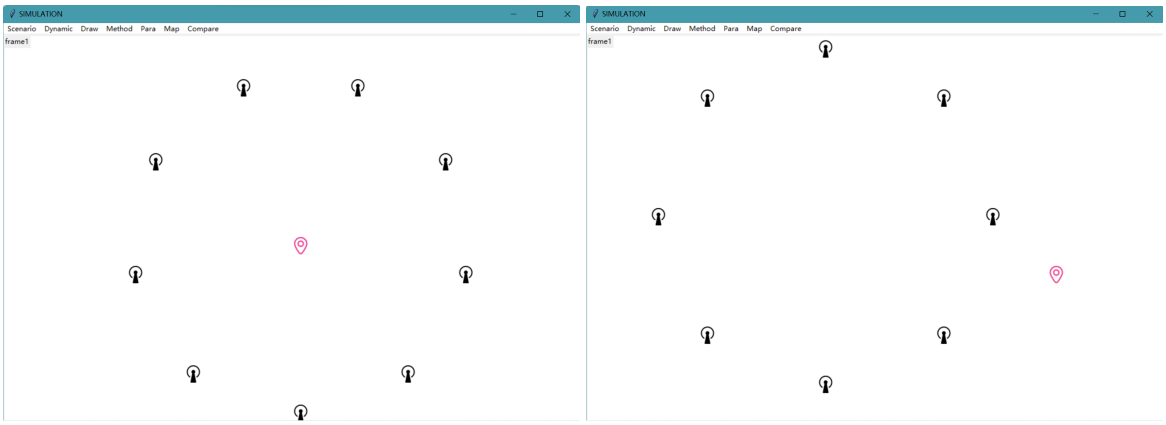


图 4-4　目标站点内外分布示意图

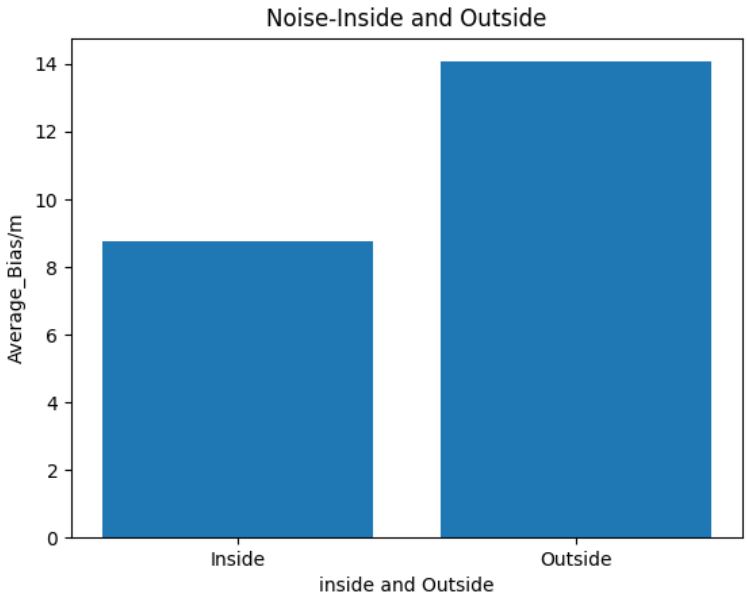
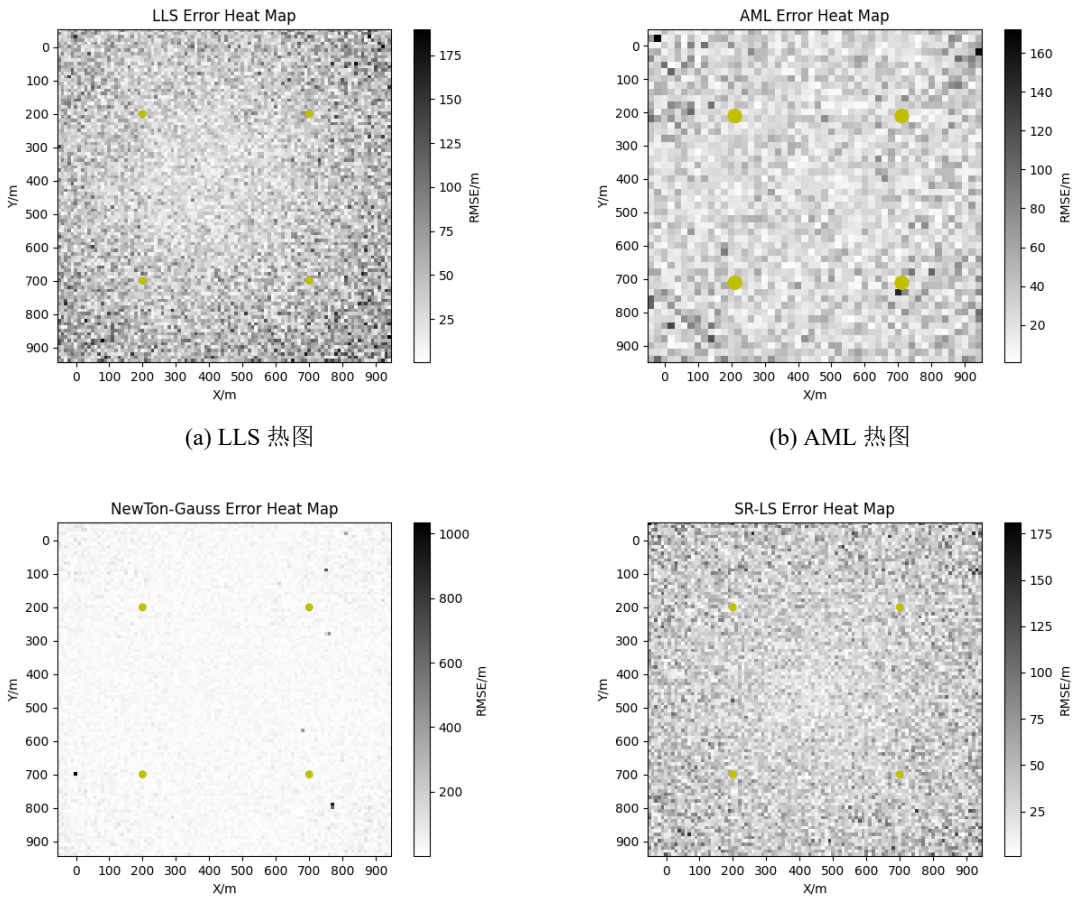


图 4-5　目标站点内外分布误差对比图

4.4 目标位置对预测误差的影响

由公式3-3可知，预测站点相对于基站的位置分布对算法的克拉美罗下限有影响，因此有必要分析不同算法受预测位置变化的影响。本文在 LOS 场景下固定四个基站位置（[200,50],[700,50],[200,550],[700,550]）移动 MT 的位置绘制均方根误差热图

由图4-6可知，当 MT 位于基站的分布中心时均方根误差较小，总体趋势随着 MT 向基



(a) LLS 热图　(b) AML 热图

(c) Newton-Gauss 热图　(d) SR-LS 热图

图 4-6　LLS,AML,Newton-Gauss,SR-LS 热图

站的靠近，均方根误差逐渐增大，这与 CRLB 的理论分析一致。直观的，可以想见当 MT 与各基站的距离差不多时，测量误差较为均匀，不会出现过大值，当 MT 偏离中心时，会出现某一站点的测量误差过大，从而影响了整体的测量效果。当 MT 到达基站位置时，意味着远离对角的基站，因此均方根误差逐渐增大。

4.5 算法运行时间对比

众所周知，衡量一个算法的好坏不仅仅在于准确度，运行的时间复杂度同样很重要。当硬件设施不满足或者对实时性要求较高时，允许牺牲部分准确度以换取较快的运行速度的情况也不少见。因此，本文对比了各类算法的平均运行时间。

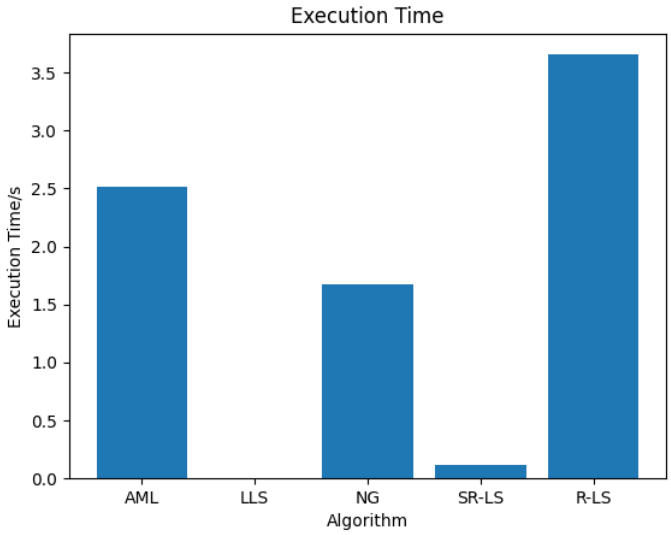


图 4-7　算法运行时间对比图

由图4-7可知，在相同测量数据情况下，最小二乘法相比于迭代法与半正定法都是最节省时间的，而半正定法消耗的时间又远远大于迭代法。对比这三种算法我们知道，最小二乘法仅仅包含一次矩阵运算，而迭代法需要多次重复并包含选根过程，半正定法则需要调用数值求解包进行规划运算，涉及多次矩阵运算，因此耗费时间最长。综上，当对精度要求不高时，利用最小二乘法进行位置预测耗时最短。

4.6　测量误差对预测误差的影响

很显然的，影响算法准确度地一个重要因素就是测量误差。随着测量误差增加，算法的准确度会逐渐降低，但是各算法随着准确度具体是如何变化的，我们通过仿真得到下图4-8。

由图4-8可知，在相同条件下 AML 算法和高斯牛顿迭代的误差要小于最小二乘法。

4.7　仿真结果总结

综上，在经过大量仿真后，我们可以得出结论。为了获得更高的准确率，我们应该在允许范围内尽可能多的布置基站，根据图4-2可知，预测误差会随着基站数目的增加而下降。此外，当基站数目固定时，基站的分布应尽可能的覆盖预测范围，由图4-5可知当目标位置位于基站区域内时，预测误差会好于位于基站区域外。同时，由图4-6可知，预测位置接近基站区域中心时，预测精度会逐渐上升，由此可知圆形分布情况下预测效果会更好。此外，由图4-7可知最小二乘法在计算时间上具有极大的优势，当精度要求不是过高时，使用最小二乘法可以极大的节省时间。由图4-8我们可知，预测的精度会随基站测量误差的增加而上升，这是理所当然的。因此，我们应尽可能的选择较好的硬件。

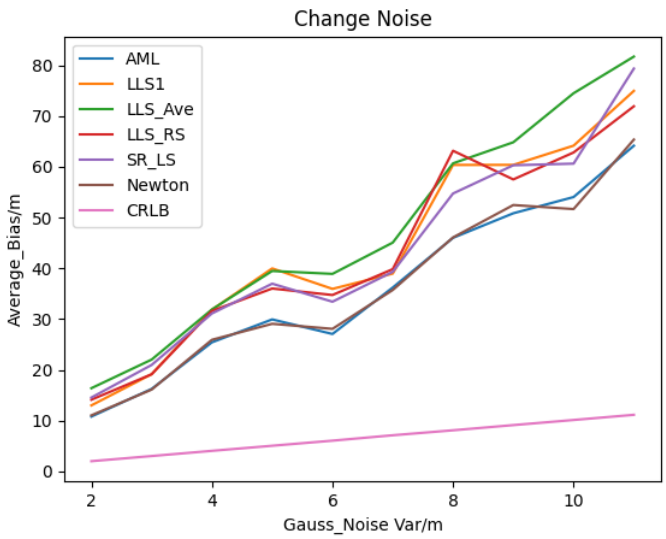


图 4-8　测量噪声对误差影响图

5　仿真软件介绍

为了可视化基站预测及方便操作各种算法，实现位置预测的仿真软件对于提高效率有着重要作用，本文利用 python 实现了位置预测的可视化，可以手动修改误差的分布参数，基站位置，移动站点的位置，选择合适的算法，设置 NLOS，LOS 的基站等功能。

5.1　初始背景界面

首先是初始界面，初始界面的设置经过修改变为了如下图所示的场景，所有的选项都在目录栏，主体仅为场景背景，使得仿真页面看上去简洁。

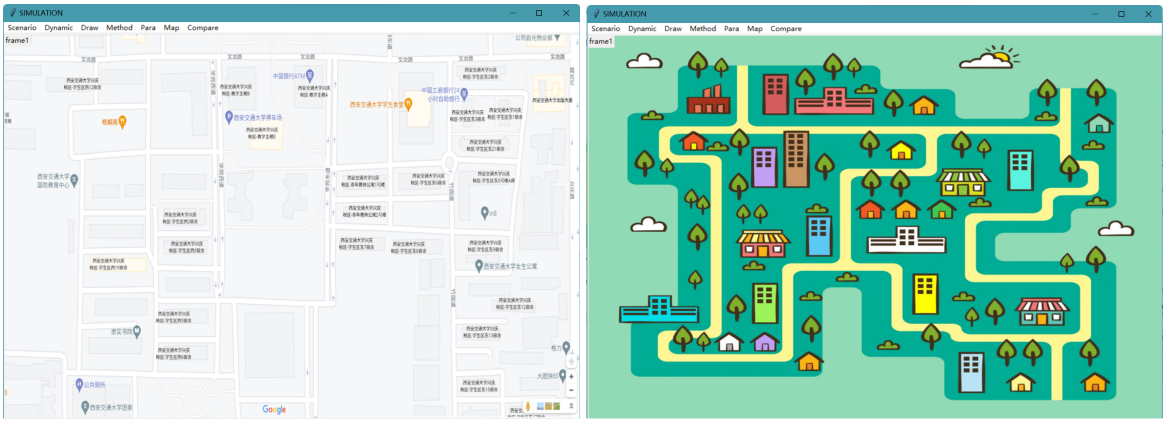


图 5-1　初始界面

5.2　参数选择界面

在初始界面目录栏设置了参数选择一项，可以更改高斯分布参数（LOS），指数分布参数（NLOS），基站数量，以及移动站点的实际位置，如图5-2。

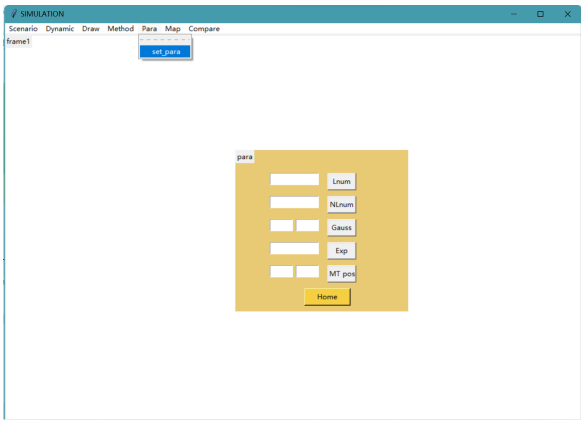


图 5-2　设置参数

5.3 基站位置分布界面

为了分析基站分布对算法的影响，还需要确定各基站的位置，该仿真软件实现了圆形分布，手动分布以及随机分布三种位置设置方式，满足不同的需求。如下， 其中

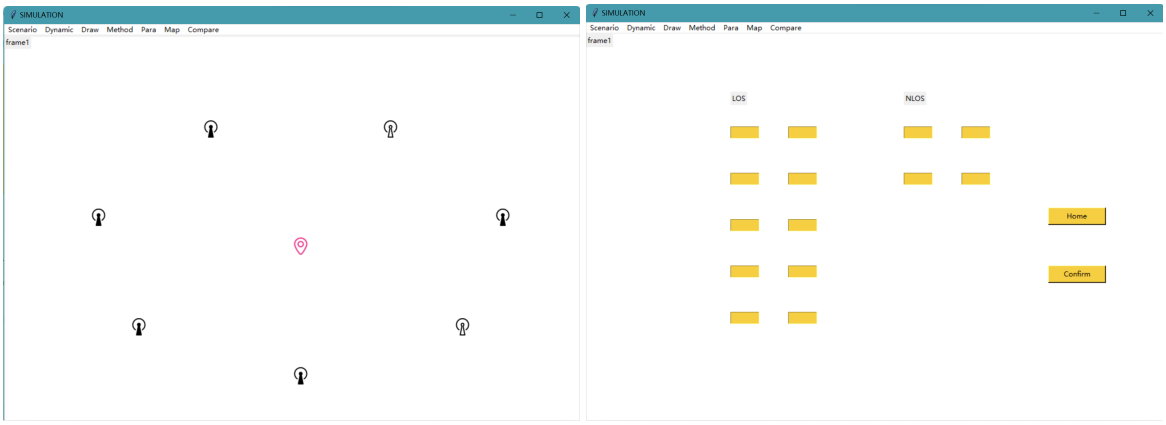


图 5-3 基站位置设置界面

实基站为 NLOS，空基站为 LOS，可以通过改变 LOS 和 NLOS 类型基站的参数以及设置位置实现不同的分布要求。

5.4 算法选择界面

最后就是对于相同场景下不同算法的选择，该仿真软件实现了 AML,LLS,R-LS,SR-LS,Newton-Gauss,RSWG 等类型的算法，预测基站位置与实际基站位置直接对比，效果明显。如图。 如上图所示，其中实基站为 NLOS，空基站为 LOS，可以通过改变

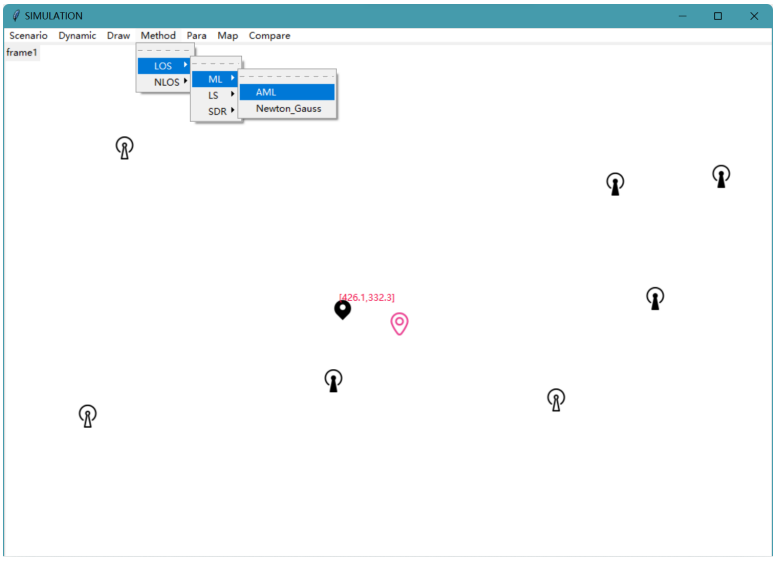


图 5-4 算法选择界面

LOS 和 NLOS 类型基站的参数以及设置位置实现不同的分布要求。

6 结论与展望

6.1 结论

本篇文章介绍了 TOA 问题在 LOS 场景，NLOS 场景以及初始时间场景的经典算法，并用仿真分析对比各算法的优劣以及各种因素对算法预测准确度的影响。LOS 场景是最简单也是最重要的场景。它提炼了 TOA 问题的最本质特征，说明 TOA 问题实质上是优化问题，区别在于目标函数的不同以及求解方法的不同。关于目标函数，主要思路是最大似然和最小二乘，分别希望求出使测量数据概率最大的解和测量误差最小的解。关于求解法，主要方法为迭代法，解析法和半正定法。

在此基础上，我们首先介绍了高斯牛顿迭代法 Gauss-Newton 以及应用迭代思想的最大似然法 AML，然后介绍了最小二乘法 LLS 并包括基于最小二乘的两个改进方法，最后介绍两个半正定法 SR-LS 和 R-LS。这些方法是 TOA 问题的经典解决方法，各有优劣。在介绍完 LOS 场景的算法后，我们接着扩展问题引入非视线场景（NLOS），这是在现实中更常见的场景，我们主要介绍了最小二乘法以及识别丢弃法。最后我们考虑了基站的初始时间使得应用更加贴近真实场景，并介绍了对应的最小二乘法和半正定法。在介绍算法之后，我们进行了仿真并比较各算法。

通过仿真我们得到如下结论。对于 LOS 场景，在相同参数下，AML 算法和 Gauss-Newton 算法的准确度要优于半正定法的 SR-LS 法同时由于最小二乘法的 LLS-Average 和 LLS-Reference，而 LLS-1 预测准确度最低。同时，最小二乘法和 SR-LS 的耗时要远远小于 AML 算法和 Gauss-Newton 迭代法的耗时。由此我们可以得出结论，当对实时性的要求高时，可以选用最小二乘法，而对准确度的要求高时，若已知噪音分布，可以选用 AML 算法若噪音分布未知，则选用高斯牛顿迭代法。

对于 NLOS 场景我们知道，仅仅使用最小二乘法进行求解误差远远大于使用识别丢弃的算法，而该算法显而易见的牺牲了时间来换取更高的准确度。同时，我们也知道，在 LOS 的 FT 数目足够时，最好是舍弃所有的 NLOS 的 FT 测得的数据。当对实时性要求较高时，可以考虑单纯应用最小二乘法求解。

对于考虑初始时间的场景，我们介绍并仿真了应用半正定法的算法，LS-Steps。我们同样使用最简单的最小二乘法进行对比，可以看出 SDP 使得求解的准确度有较大的提升。

此外，我们还分析了位置分布等因素对预测准确度的影响，我们选择最小二乘法作为衡量算法，改变基站分布，基站数目，移动站点位置等，得到如下结论。

随着基站数目的增加。测量误差会不断减小，这是因为有更多的可用信息，而当基站数目和预测范围一定时，基站分布包围的面积越大，预测的误差越小。此外，当移动站点分布在基站包围区域内时，预测的准确度要高于站点在基站区域外，这也部分

解释了基站面积得出的结论。最后，当基站分布为矩形时，站点越靠近矩形中心，误差越小，这是因为每个基站测量的数据都能有效的利用，而不会导致某个基站的测量出现较大误差。

6.2 展望

实际应用中，绝大部分情况都包含有 NLOS 误差，因此只针对 LOS 场景的分析可以提供借鉴意义，但是距离实际应用还有一段距离。现实场景中，NLOS 误差分布并不是已知的，不同场景下的 NLOS 误差可能有不同分布，即使分布相同也有可能有不同的分布参数。因此，确定不同场景的 NLOS 误差模型系数，有利于进一步提高算法准确度，是需要进一步研究的问题。

除 NLOS 误差，实际应用中还可能包括基站位置误差，初始时间误差，多用户多信道误差等因素，针对这些因素的研究有利于进一步提高实际应用的准确度。综合考虑这些因素的算法，则更加具有应用价值，而这类算法有待进一步研究。

此外，定位问题不止 TOA 一种方法，相应的还有 TDOA，RSS，AOA 等方法，这些方法获取测量数据的思路不同，因此有各自的优势，本文虽然仅仅讨论了 TOA 情况下的各种算法，但是这些算法的思想可以应用到其他方法中。

致　　谢

个人的成长离不开集体的支持，没有人可以脱离他人的帮助独自成长。无论是学习还是生活，我们始终要怀揣感恩之心，感谢他人为自己的付出。如果我自己能有一些小小的进步，这一定是老师的帮助、家庭的关怀、朋友的支持共同作用的结果。

首先，感谢我的老师。本文是在蔡远利老师和姜浩楠老师的共同指导下完成的。蔡远利老师高屋建瓴，经验丰富，对我的毕设题目有很深的理解，可以从很高的层面指出我的问题并给我提供解决的方向。姜老师认真负责，在准备毕设的过程中，一直耐心地指导我。从推荐经典文献，到提供解决思路，再到论文的修改审核，姜浩楠老师以认真负责的态度陪伴我毕设的整个过程。多谢蔡远利老师和姜浩楠老师的辅导，我才能顺利完成论文的写作。在此向这两位老师表示最真挚的感谢！

此外，感谢我的父母。是他们的无私奉献让我得以继续走在求学路上；是他们无条件的支持，让我在迷茫时能有继续前进的动力。借此机会，向我的父母表示真挚的感谢！

最后，感谢我的朋友与同学。大学的生活不只有求学，还有与朋友一起度过的丰富生活。在空闲时，一起去打球；在结束期末后，一起去聚餐；在遇到困难时，彼此互相帮助。这段时光很精彩，谢谢你们！

参考文献

[1] Güvenç I, Chong C C. A survey on TOA based wireless localization and NLOS mitigation techniques[J]. IEEE Communications Surveys and Tutorials, 2009, 11(3) : 107 – 124.

[2] Sayed A H, Tarighat A, Khajehnouri N. Network-based wireless location: Challenges faced in developing techniques for accurate wireless location information[J]. IEEE Signal Processing Magazine, 2005, 22(4) : 24 – 40.

[3] Gustafsson F, Gunnarsson F. Mobile positioning using wireless networks: Possibilities and fundamental limitations based on available wireless network measurements[J]. IEEE Signal Processing Magazine, 2005, 22(4) : 41 – 53.

[4] Gezici S. A survey on wireless position estimation[J]. Wireless Personal Communications, 2008, 44(3) : 263 – 282.

[5] Marchand N. Error Distributions of Best Estimate of Position from Multiple Time Difference Hyperbolic Networks[J]. IEEE Transactions on Aerospace and Navigational Electronics, 1964, ANE-11(2) : 96 – 100.

[6] Turin G L, Jewell W S, Johnston T L. Simulation of Urban Vehicle-Monitoring Systems[J]. IEEE Transactions on Vehicular Technology, 1972, 21(1) : 9 – 16.

[7] Chang C, Sahai A. Estimation bounds for localization[C] // 2004 First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, IEEE SECON 2004. 2004 : 415 – 424.

[8] Chan Y T, Tsui W Y, So H C, et al. Time-of-arrival based localization under NLOS conditions[J]. IEEE Transactions on Vehicular Technology, 2006, 55(1) : 17 – 24.

[9] Li Z, Trappe W, Zhang Y, et al. Robust statistical methods for securing wireless localization in sensor networks[C] // 2005 4th International Symposium on Information Processing in Sensor Networks, IPSN 2005 : Vol 2005. 2005 : 91 – 98.

[10] Venkatesh S, Buehrer R M. A linear programming approach to NLOS error mitigation in sensor networks[C] // Proceedings of the Fifth International Conference on Information Processing in Sensor Networks, IPSN ’06 : Vol 2006. 2006 : 301 – 308.

[11] Caffery J, Stiiber G L. Subscriber location in CDNA cellular networks[J]. IEEE Transactions on Vehicular Technology, 1998, 47(2) : 406 – 416.

[12] Beck A, Stoica P, Li J. Exact and approximate solutions of source localization problems[J]. IEEE Transactions on Signal Processing, 2008, 56(5) : 1770 – 1778.

[13] Chen P C. A non-line-of-sight error mitigation algorithm in location estimation[C] // IEEE Wireless Communications and Networking Conference, WCNC : Vol 1. [S.l.] : Institute of Electrical and Electronics Engineers Inc., 1999 : 316 – 320.

[14] Sun G L, Guo W. Bootstrapping M-estimators for reducing errors due to non-line-of-sight (NLOS) propagation[J]. IEEE Communications Letters, 2004, 8(8) : 509 – 510.

[15] Xu E, Ding Z, Dasgupta S. Source localization in wireless sensor networks from signal time-of-arrival measurements[J]. IEEE Transactions on Signal Processing, 2011, 59(6) : 2887 – 2897.

[16] 王长强, 徐爱功, 隋心. 超宽带双向到达时间测距的室内动态定位方法 [J]. 测绘科学, 2017, 42(12) : 151 – 156.

[17] 胡国兵, 刘渝, 邓振淼. 基于 Haar 小波变换的信号到达时间估计 [J]. 系统工程与电子技术, 2009(7) : 1615 – 1619.

[18] 邓兵, 孙正波, 杨乐. 带目标高度约束信息的ＴＯＡ无源定位 [J]. 西安电子科技大学学报（自然科学版）, 2017, 44(3) : 133 – 137.

[19] Baidu. 半迭代法 [EB/OL]. 2022.
http://baike.baidu.com/view/20510213.html.

[20] Chan Y T, Hang H Y C, Ching P C. Exact and approximate maximum likelihood localization algorithms[J]. IEEE Transactions on Vehicular Technology, 2006, 55(1): 10 – 16.

[21] Caffery J J. New approach to the geometry of TOA location[C] // IEEE Vehicular Technology Conference: Vol 4. [S.l.]: IEEE, 2000: 1943 – 1949.

[22] Qi Y, Kobayashi H, Suda H. Analysis of wireless geolocation in a non-line-of-sight environment[J]. IEEE Transactions on Wireless Communications, 2006, 5(3): 672 – 681.

附录 A　翻译论文原文

IEEE COMMUNICATIONS SURVEYS & TUTORIALS, VOL. 11, NO. 3, THIRD QUARTER 2009 107

A Survey on TOA Based Wireless Localization and NLOS Mitigation Techniques

İsmail Güvenç, *Member, IEEE,* and Chia-Chin Chong, *Senior Member, IEEE*

***Abstract*—Localization of a wireless device using the time-of-arrivals (TOAs) from different base stations has been studied extensively in the literature. Numerous localization algorithms with different accuracies, computational complexities, *a-priori* knowledge requirements, and different levels of robustness against non-line-of-sight (NLOS) bias effects also have been reported. However, to our best knowledge, a detailed unified survey of different localization and NLOS mitigation algorithms is not available in the literature. This paper aims to give a comprehensive review of these different TOA-based localization algorithms and their technical challenges, and to point out possible future research directions. Firstly, fundamental lower bounds and some practical estimators that achieve close to these bounds are summarized for line-of-sight (LOS) scenarios. Then, after giving the fundamental lower bounds for NLOS systems, different NLOS mitigation techniques are classified and summarized. Simulation results are also provided in order to compare the performance of various techniques. Finally, a table that summarizes the key characteristics of the investigated techniques is provided to conclude the paper.**

***Index Terms*—Cramer-Rao Lower Bound, Location Estimation, NLOS Mitigation, Positioning, Time-of-Arrival.**

I. Introduction

RECENTLY, location awareness has received great deal of interest in many wireless systems such as cellular networks, wireless local area networks, and wireless sensor networks due its capability to provide wide range of add-on applications. Location-based services such as location based advertisement, location based social networking, and E911 emergency services have become more important in order to enhance the future lifestyle. For example, the location based advertisement allows users to selectively receive promotional advertisement by strategically placing messaging near where buyer behavior can be most immediately influenced. For instance, a user will receive electronics sales items and coupons only when he/she is entering a shopping mall. On the other hand, location based social networking may further enhance the Internet based social networking services such as Facebook, Friendsters, MySpace, etc. by allowing users forming groups based on their social preference and interest. For the E911 emergency services, user will be able to make emergency call that allows local authority to track and locate the user position under both indoor and outdoor scenarios with high accuracy. The aforementioned example applications offered by location awareness will enable ubiquitous and context aware network services which necessitate the location of the wireless device to be accurately estimated.

Even though location estimation problems have been investigated extensively in the literature in the last few decades, there are still some open issues that remain unresolved. One of the key challenges in localization is the efficiency and preciseness of the estimation in dense cluttered non-line-of-sight (NLOS) scenarios. NLOS scenarios occur when there is an obstruction between transmitter (TX) and receiver (RX) which are commonly encountered in modern wireless system deployment for both indoor (e.g., residential, office, shopping malls, etc.) and outdoor (e.g., metropolitan, urban area, etc.) environments. In such circumstances, the use of the global positioning system (GPS) becomes impractical if not impossible.

Several previous works have been reported in the literature (e.g., [1]–[5]) that provide extensive review on localization using angle-of-arrival (AOA), time-of-arrival (TOA), time-difference-of-arrival (TDOA), and received-signal-strength (RSS) techniques. However, none of these works investigate the impact of NLOS mitigation techniques in detail in order to improve the performance degradation incurred by the blockage of the direct path. In this paper, we provide a comprehensive survey for TOA based localization techniques which are applicable for both LOS and NLOS scenarios. For other techniques such as AOA, TDOA, and fingerprint-based methods, interested readers are referred to [6]–[8] and the references therein.

The goal of this paper are two folds. Firstly, to provide a unified overview of different TOA based localization techniques and related NLOS mitigation approaches. Secondly, to study the trade-offs and inter-relations among the various localization techniques. Note that specific techniques required to estimate the TOA of the first arriving path for TOA-based ranging are outside the scope of this paper; interested readers are referred to [9]–[17].

The paper is organized as follows. Section II briefly reviews different location estimation techniques; namely, TOA, TDOA, AOA, RSS, and pattern-matching based approaches. In Section III, the TOA based localization scenario is outlined and the system model as well as the problem definition are presented. For the rest of the sections, Section IV and Section V are dedicated to LOS scenarios while Sections VI-X are dedicated to NLOS scenarios. Section IV provides fundamental lower bounds for LOS systems and summarizes some of the key maximum likelihood (ML) based techniques

Manuscript received 9 December 2007; revised 2 June 2008.
The authors are with DOCOMO Communications Laboratories USA, 3240 Hillview Avenue, Palo Alto, CA 94304, USA (e-mail: iguvenc@docomolabs-usa.com, cchong@docomolabs-usa.com).
Digital Object Identifier 10.1109/SURV.2009.090308.

1553-877X/09/$25.00 © 2009 IEEE

Authorized licensed use limited to: Xian Jiaotong University. Downloaded on October 19,2021 at 12:08:56 UTC from IEEE Xplore. Restrictions apply.

that can achieve close to these fundamental bounds. Section V provides a unified discussion of different least squares (LS) location estimation techniques for TOA-based systems. Section VI provides fundamental lower bounds and some ML techniques for NLOS systems, Section VII is on LS techniques in NLOS scenarios, Section IX is on robust estimators in NLOS scenarios, and Section X is a review of identify and discard type of NLOS mitigation techniques. Section XI studies the impact of FT distribution on the localization accuracy, and finally, Section XII summarizes the available techniques and provides some concluding remarks.

II. Overview of Different Localization Techniques

In general, there are two phases towards realization of highly accurate location awareness applications. Firstly, through accurate ranging i.e., an action of estimating the distance between two nodes, and secondly, through localization i.e., an action of determining the exact location of an unknown node from known nodes (anchors) by means of the intersection of three or more measured ranges from known nodes. In this section, a brief overview of five commonly used localization techniques will be discussed. A summary is also presented in Table I.

For the TOA based technique, the distance information is extracted from the propagation delay between a TX and an RX. This technique can be further classified into TOA one-way ranging and TOA two-way-ranging. The former requires perfect synchronization between TX and RX, while the latter does not require synchronization between TX and RX. This technique is more common in cellular networks since the receiving nodes are typically synchronized to base stations.

For the TDOA based technique, the difference between TOAs in several RXs is used to reconstruct a TX's position. This could either be based on the difference in the times at which a single signal from a single node arrives at three or more nodes, or based on the difference in the times at which multiple signals from a single node arrive at another node. This requires highly precise synchronization between the RXs, but not precise synchronization between TX and RXs. TDOA based technique is more commonly used in wireless sensor networks.

For the AOA based technique, the distance between nodes is reconstructed from the angle between them. The major disadvantage of this technique is that it requires new hardware i.e., adoption of antenna arrays and a minimum distance between the antenna elements which results additional costs and larger node sizes. Furthermore, this technique is highly sensitive to multipath, NLOS conditions, and array precision.

For the RSS based technique, the distance is measured based on the attenuation introduced by the propagation of the signal from TX to RX. Since the relation between distance and attenuation depends on channel behavior, an accurate propagation model is required for reliable distance estimation. The main advantage of this technique is its low cost and the fact that most RXs are capable of estimating the RSS. However, the MT mobility and unpredictable variations in channel behavior can occasionally lead to large errors in distance evaluation. Also, this technique is very susceptible to noise and interference. Thus, the RSS technique is not an accurate method, and its adoption is confined to applications that require coarse estimation.

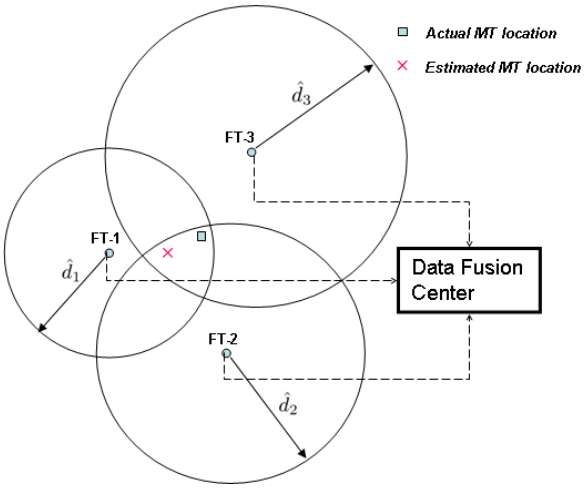


Fig. 1. Illustration of a simple scenario for wireless localization.

Finally, for pattern-matching based techniques, fingerprint information of the measured radio signals at different geographical locations are utilized for position estimation. Such information should be location-sensitive and can be collected during a training (off-line) phase in a database. During the real-time (on-line) phase, the fingerprint information can be used to locate the mobile node. A fingerprint database can be simply composed of received signal strengths from/at different reference nodes and at different mobile locations. However, it may as well capture more detailed fingerprint information, such as the mean excess delay, root mean square (RMS) delay spread, maximum excess delay, total received power, number of multipath components etc. A challenge with such systems is that due to changes in the channel and environment, the fingerprint database may become unreliable, and may need to be updated frequently.

III. System Model for TOA-Based Location Estimation

Consider a wireless network as in Fig. 1 where there are N fixed terminals (FTs)[1], $\hat{\mathbf{x}} = [\hat{x}\ \hat{y}]^T$ is the estimate of the mobile terminal (MT) location, $\mathbf{x}_i = [x_i\ y_i]^T$ is the position of the ith FT, $\hat{d}_i$ is the measured distance between the MT and the ith FT commonly modeled as

$$\hat{d}_i = d_i + b_i + n_i = ct_i, \quad i = 1, 2, ..., N\ , \tag{1}$$

where t_i is the TOA of the signal at the ith FT, c is the speed of light, d_i is the real distance between the MT and the ith FT, $n_i \sim \mathcal{N}\big(0, \sigma_i^2\big)$ is the additive white Gaussian noise (AWGN) with variance σ_i^2, and b_i is a positive distance bias introduced

[1]FT is usually a base station in cellular networks, an anchor node in sensor networks, or an access point in wireless local area networks.

Authorized licensed use limited to: Xian Jiaotong University. Downloaded on October 19,2021 at 12:08:56 UTC from IEEE Xplore. Restrictions apply.

TABLE I
OVERVIEW OF DIFFERENT LOCALIZATION ALGORITHMS.

Localization Technique	Summary and Characteristics	Strength and Weakness	Usage and Applicability
TOA	Uses distance information between FT and MT.	One-way ranging requires perfect synchronization, while two-way ranging does not.	More common in cellular networks.
TDOA	Difference between TOAs in several FTs are utilized.	Needs highly precise synchronization between MTs, while not precise synchronization between FTs and MTs.	More common in wireless sensor networks.
AOA	Uses the angle information to construct the lines between MT and FTs and use their intersection to find MT location	Requires new hardware (antenna arrays). This means additional costs and larger node sizes.	More appropriate for FTs rather than MTs due to large size. Otherwise MT size has to be able to accommodate an antenna array.
RSS	Distance is estimated based on the attenuation introduced by propagation of the signal from FT to MT.	An accurate propagation model is needed for reliable distance estimation. It is low cost due to most RX being able to estimate RSS. MT mobility and channel variation may yield large errors.	Since it has low-precision characteristic, typically used in applications which require coarse estimate.
Pattern Matching	Fingerprint information of measured radio signal at different geographical locations are utilized.	Needs an off-line training stage to obtain a database. Also, this database may be unreliable if the channel and environment changes with time.	Mostly used in wireless local area networks with RSS as the metric used in the database. Also considered for cellular systems.

due to the blockage of direct path given by

$$b_i = \begin{cases} 0 \ , & \text{if } i\text{th FT is LOS} \ , \\ \psi_i \ , & \text{if } i\text{th FT is NLOS} \ . \end{cases} \tag{2}$$

For NLOS FTs, the bias term ψ_i was modeled in different ways in the literature such as exponentially distributed [18], [19], uniformly distributed [20], [21], Gaussian distributed [22], constant along a time window [23], or based on an empirical model from measurements [24], [25]. Typically, the model depends on the wireless propagation channel and the specific technology under consideration (e.g., cellular networks, wireless sensor networks, etc.).

Let

$$\mathbf{d} = \mathbf{d}(\mathbf{x}) = [d_1, d_2, ..., d_N]^T \ , \tag{3}$$

be a vector of actual distances between the MT and the FTs,

$$\hat{\mathbf{d}} = [\hat{d}_1, \hat{d}_2, ..., \hat{d}_N]^T \ , \tag{4}$$

be a vector of measured distances, and

$$\mathbf{b} = [b_1, b_2, ..., b_N]^T \ , \tag{5}$$

be a bias vector. Also let

$$\mathbf{Q} = \mathrm{E}[\mathbf{n}\mathbf{n}^T] = \mathrm{diag}[\sigma_1^2, \sigma_2^2, ..., \sigma_N^2]^T \ , \tag{6}$$

to denote the covariance of noise vector $\mathbf{n} = [n_1, n_2, ..., n_N]^T$ with the assumption that all the noise terms are zero mean and independent Gaussian random variables.

In the absence of noise and NLOS bias, the true distance d_i between the MT and the ith FT defines a circle around the ith FT corresponding to possible MT locations

$$(x - x_i)^2 + (y - y_i)^2 = d_i^2, \quad i = 1, 2, ..., N \ , \tag{7}$$

where all the circles intersect at the same point, and solving these expressions jointly gives the exact MT location. However, the noisy measurements and NLOS bias at different FTs yield circles which do not intersect at the same point (see Fig. 1), resulting in the inconsistent equations as follows

$$(x - x_i)^2 + (y - y_i)^2 = \hat{d}_i^2, \quad i = 1, 2, ..., N \ . \tag{8}$$

In order to have more compact expressions throughout the paper, we further define the following terms

$$s = x^2 + y^2 \ , \ k_i = x_i^2 + y_i^2 \ . \tag{9}$$

The problem of TOA-based location estimation can be defined as the estimation of the MT's location $\mathbf{x}$ from the noisy (and possibly biased) distance measurements in (4) given the FT locations $\mathbf{x}_i$; in other words, given the set of equations in (8). Various localization techniques were proposed in the literature in order to estimate the MT location from (8) in LOS and NLOS scenarios. In the following sections, we will review these algorithms and discuss their trade-offs.

IV. LOS SCENARIOS: FUNDAMENTAL LIMITS AND ML SOLUTIONS

In this section, we will overview the fundamental lower bounds and ML type of algorithms for LOS scenarios (i.e., $b_i = 0$ for all i). First, we define below the ML algorithm that maximizes the conditional probability of the measured distances $\hat{\mathbf{d}}$. Then, the Cramer-Rao lower bound (CRLB) will be derived in the following section. Two other sub-optimum ML type of algorithms asymptotically achieving the CRLB will also be described.

A. Maximum Likelihood Algorithm

In the absence of NLOS bias (i.e., $b_i = 0$ for all i), the conditional probability density function (PDF) of $\hat{\mathbf{d}}$ in (4) given $\mathbf{x}$ can be expressed as follows [26], [27]

$$P(\hat{\mathbf{d}}|\mathbf{x}) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left\{ -\frac{(\hat{d}_i - d_i)^2}{2\sigma_i^2} \right\} \tag{10}$$

$$= \frac{1}{\sqrt{(2\pi)^N \det(\mathbf{Q})}} \exp\left\{ -\frac{\mathbf{J}}{2} \right\} \ , \tag{11}$$

where

$$\mathbf{J} = \left[\hat{\mathbf{d}} - \mathbf{d}(\mathbf{x})\right]^T \mathbf{Q}^{-1} \left[\hat{\mathbf{d}} - \mathbf{d}(\mathbf{x})\right] \ , \tag{12}$$

Authorized licensed use limited to: Xian Jiaotong University. Downloaded on October 19,2021 at 12:08:56 UTC from IEEE Xplore. Restrictions apply.

with $\mathbf{Q}$ as given in (6). Then, the ML solution for $\mathbf{x}$ is the one that maximizes $P(\hat{\mathbf{d}}|\mathbf{x})$, i.e.,

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} P(\hat{\mathbf{d}}|\mathbf{x}) \ . \tag{13}$$

Note that solving for $\mathbf{x}$ from (13) requires a search over possible MT locations which is computationally intensive.

For the special case of $\sigma_i^2 = \sigma^2$ for all i, the ML solution in (13) is equivalent to minimizing $\mathbf{J}$. In order to find the minimum value of $\mathbf{J}$, the gradient of $\mathbf{J}$ with respect to $\mathbf{x}$ is equated to zero, yielding [26]

$$\sum_{i=1}^{N} \frac{(d_i - \hat{d}_i)(x - x_i)}{d_i} = 0 \ , \tag{14}$$

$$\sum_{i=1}^{N} \frac{(d_i - \hat{d}_i)(y - y_i)}{d_i} = 0 \ , \tag{15}$$

which are non-linear equations. Hence, $\mathbf{x}$ can not be solved in closed form from (14) and (15) using a linear least squares (LS) algorithm. Also, both (14) and (15) depend on d_i, which are unknown. Even though a closed form ML solution is not possible, approximate and iterative ML techniques can be derived as will be discussed in Section IV-C and Section IV-D, which may asymptotically achieve the CRLB.

B. Cramer-Rao Lower Bounds

Given the conditional PDF of $\hat{\mathbf{d}}$ as in (10), we may derive the CRLB for TOA based location estimation. The CRLB, in general, can be defined as the theoretical lower bound on the variance of any unbiased estimator of an unknown parameter. The CRLB for the TOA based location estimation mainly depends on the following factors:

- Positions of the FTs ($\mathbf{x}_i$),
- True position of the MT ($\mathbf{x}$), and
- Measurement noise variances (σ_i^2).

The CRLB is calculated using the Fisher information matrix (FIM), whose elements are defined as

$$[\mathbf{I}(\mathbf{x})]_{ij} = -\mathrm{E}\Big[\frac{\partial^2 \ln P(\hat{\mathbf{d}}|\mathbf{x})}{\partial \mathbf{x}_i \partial \mathbf{x}_j}\Big] \ . \tag{16}$$

Then, using the PDF given in (10), the FIM can be calculated as [27], [28]

$$\mathbf{I}(\mathbf{x}) = \begin{bmatrix} \sum_{i=1}^{N} \frac{(x-x_i)^2}{\sigma_i^2 d_i^2} & \sum_{i=1}^{N} \frac{(x-x_i)(y-y_i)}{\sigma_i^2 d_i^2} \\ \sum_{i=1}^{N} \frac{(x-x_i)(y-y_i)}{\sigma_i^2 d_i^2} & \sum_{i=1}^{N} \frac{(y-y_i)^2}{\sigma_i^2 d_i^2} \end{bmatrix} , \tag{17}$$

$$= \begin{bmatrix} \sum_{i=1}^{N} \frac{\cos^2(\alpha_i)}{\sigma_i^2} & \sum_{i=1}^{N} \frac{\cos(\alpha_i)\sin(\alpha_i)}{\sigma_i^2} \\ \sum_{i=1}^{N} \frac{\cos(\alpha_i)\sin(\alpha_i)}{\sigma_i^2} & \sum_{i=1}^{N} \frac{\sin^2(\alpha_i)}{\sigma_i^2} \end{bmatrix} , \tag{18}$$

where α_i defines the angle from the ith FT to the MT, and the CRLB is given by $\mathbf{I}^{-1}(\mathbf{x})$. Thus, for an estimate $\hat{\mathbf{x}}$ of the MT location obtained with any unbiased estimator, we have

$$\mathrm{E}_{\mathbf{x}}\big[(\hat{\mathbf{x}} - \mathbf{x})(\hat{\mathbf{x}} - \mathbf{x})^T\big] \geq \mathbf{I}^{-1}(\mathbf{x}) \ . \tag{19}$$

The CRLB can be related to another important measurement metric referred to as the geometric dilution of precision

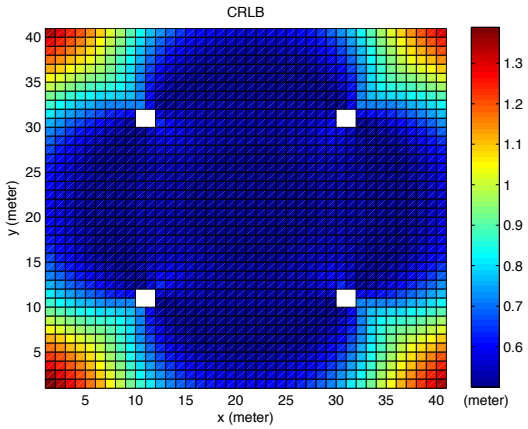


Fig. 2. The CRLB for a simple localization scenario where there are four FTs at the corners of a square room of size 40×40 meters.

(GDOP) in the literature. For identical noise variances $\sigma_i^2 = \sigma^2$ at different FTs, the GDOP can be defined as

$$\mathrm{GDOP} = \frac{\mathrm{RMSE}_{\mathrm{loc}}}{\mathrm{RMSE}_{\mathrm{range}}} = \frac{\sigma_{\mathrm{loc}}}{\sigma} \ , \tag{20}$$

where $\mathrm{RMSE}_{\mathrm{loc}}$ and $\mathrm{RMSE}_{\mathrm{range}}$ are the root mean square error (RMSE) of the location estimate and the range estimate, respectively, and σ_{loc} is the standard deviation of the location estimate. GDOP depends highly on the positions of the FTs and the MT location. While GDOP values smaller than three are usually preferable, values larger than six may imply a very bad geometry of the FTs. If the employed location estimator can achieve the CRLB, the GDOP is given by

$$\mathrm{GDOP} = \frac{\sqrt{\mathrm{trace}\Big[\mathbf{I}^{-1}(\mathbf{x})\Big]}}{\sigma} \ , \tag{21}$$

$$= \sqrt{\mathrm{trace}\Big[\tilde{\mathbf{I}}^{-1}(\mathbf{x})\Big]} \ , \tag{22}$$

where

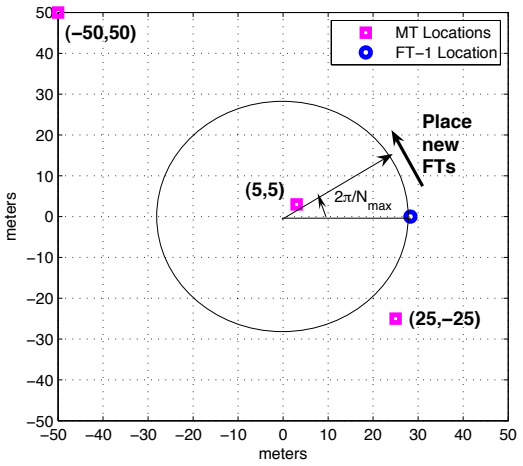
$$\tilde{\mathbf{I}}(\mathbf{x}) = \begin{bmatrix} \sum_{i=1}^{N} \cos^2(\alpha_i) & \sum_{i=1}^{N} \cos(\alpha_i)\sin(\alpha_i) \\ \sum_{i=1}^{N} \cos(\alpha_i)\sin(\alpha_i) & \sum_{i=1}^{N} \sin^2(\alpha_i) \end{bmatrix} . \tag{23}$$

The relation between the achievable localization accuracy and the geometry between the locations of the MT and the FTs is apparent from (22).

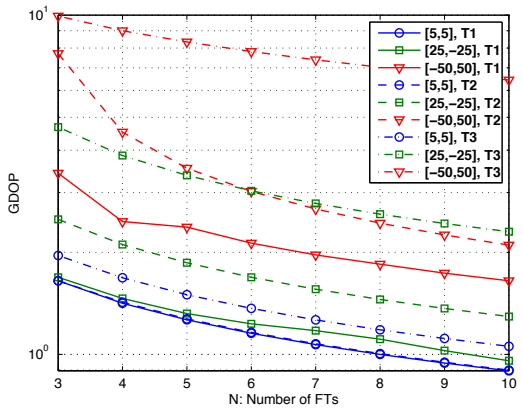
1) Simulation Results: The CRLBs for a simple wireless localization scenario at different MT locations is illustrated in Fig. 2. Four FTs are positioned at $[0, 0]$ m, $[0, 20]$ m, $[20, 0]$ m, and $[20, 20]$ m and we have $\sigma^2 = 0.5$ for all the FTs. The CRLB becomes lower when the MT is closer to the center of the room. Also, at four specific MT locations, the FIM in (16) becomes singular and does not have a matrix inverse, which explains the white spots in Fig. 2.

In order to see the typical values that GDOP may take in wireless localization, some example node topologies are simulated in Fig. 3. Three MT locations are considered, namely $[5, 5]$ m, $[25, -25]$ m, and $[-50, 50]$ m. FT-1 location is

Authorized licensed use limited to: Xian Jiaotong University. Downloaded on October 19,2021 at 12:08:56 UTC from IEEE Xplore. Restrictions apply.



(a) Topology for GDOP simulations.



(b) GDOP versus the number of FTs for different topologies $T1$, $T2$, and $T3$.

Fig. 3. GDOPs for different topologies and MT locations.

fixed to $[20\sqrt{2}, 0]$ m and new FTs are added counter-clockwise around the illustrated circle. Three topologies are considered, namely, T1, T2, and T3, and for each topology, a new FT is placed at an increment of $2\pi/N_{\max}$, $\pi/N_{\max}$, and $\pi/2N_{\max}$ radians, respectively, where $N_{\max} = 10$ denotes the maximum number of FTs. The GDOP is less than two for an MT located at $[5, 5]$ m for all the topologies, and becomes better as more FTs are deployed. For an MT located at $[-50, 50]$ m, GDOP is worst, and it may be as large as 10 for T3. The results show that as we increase the number of FTs, it is possible to have GDOP values smaller than 1, which implies that the standard deviation of the location estimate becomes smaller than the standard deviation of the distance measurements.

C. Two-Step ML Algorithm

While the CRLB gives a lower bound on the best achievable accuracy, it may practically be difficult to approach it in realistic scenarios. One of the earlier TOA based techniques in the literature that approaches the CRLB under certain conditions is introduced in [29]. It is a two-step ML algorithm, where the ML solution for the MT location estimate can be obtained as

$$\hat{\boldsymbol{\theta}} = \frac{1}{2}(\mathbf{A}_1^T \boldsymbol{\Psi}^{-1} \mathbf{A}_1)^{-1} \mathbf{A}_1^T \boldsymbol{\Psi}^{-1} \mathbf{p}_1 \ , \tag{24}$$

where $\mathbf{A}_1$ and $\mathbf{p}_1$ are as defined in (35), and

$$\boldsymbol{\theta} = [x, y, s]^T \ , \tag{25}$$

$$\boldsymbol{\Psi} = \mathrm{E}[\psi\psi^T] = \mathbf{BQB} \ , \tag{26}$$

$$\psi = \mathbf{p}_1 - \mathbf{A}_1\theta \ , \tag{27}$$

$$\mathbf{B} = \mathrm{diag}\{2d_1, 2d_2, ..., 2d_N\} \ . \tag{28}$$

Since the elements of $\mathbf{B}$ are unknown distances, an approximate solution is obtained by using the measurements $\hat{d}_i$ instead of actual distances d_i in $\mathbf{B}$ for obtaining an initial solution. Then, a more accurate solution is obtained using this initial solution to re-calculate $\mathbf{B}$, and few iterations are shown to be sufficient for convergence.

D. Approximate ML Algorithm

Another ML based technique that approaches to the CRLB is proposed in [26]. It was shown in Section IV-A that the solution of the MT location using the ML method requires the knowledge of true distances d_i. With the assumption that $\sigma_i^2 = \sigma^2 \ \forall i$, [26] proposes an approximate ML (AML) solution that achieves the CRLB in most scenarios. They use the identity

$$d_i - \hat{d}_i = \frac{d_i^2 - \hat{d}_i^2}{d_i + \hat{d}_i} \ , \tag{29}$$

in solving (14) and (15) and obtain the following matrix equation [26]

$$2\begin{bmatrix} \sum g_i x_i & \sum g_i y_i \\ \sum h_i x_i & \sum h_i y_i \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \sum g_i(s + k_i - \hat{d}_i^2) \\ \sum h_i(s + k_i - \hat{d}_i^2) \end{bmatrix} \ , \tag{30}$$

where s and k_i are as in (9) and all the summations are from 1 to N, with

$$g_i = \frac{x - x_i}{d_i(d_i + \hat{d}_i)}, \quad h_i = \frac{y - y_i}{d_i(d_i + \hat{d}_i)} \ . \tag{31}$$

Note that other than s, the weights g_i and h_i also depend on $\mathbf{x}$ in the above relations, which is (obviously) unknown. Hence, the AML technique first obtains a rough initial estimate (e.g., using the estimator in (37)) of $\mathbf{x}$ to compute g_i and h_i. Then, (30) is solved using a LS algorithm and $\mathbf{x}$ is obtained in terms of s, which results in a quadratic expression. A root selection routine is then used to select the appropriate solution for $\mathbf{x}$. The new solution is re-used to compute new g_i and h_i, and few iterations of the algorithm can achieve results close to the CRLB.

V. LOS Scenarios: Least-Squares Techniques

In this section, first, we will review non-linear least squares (NLS) techniques for estimating the position of an MT. Then, some linearization techniques will be briefly discussed.

Authorized licensed use limited to: Xian Jiaotong University. Downloaded on October 19,2021 at 12:08:56 UTC from IEEE Xplore. Restrictions apply.

A. Non-Linear Least Squares

The NLS is a well known technique for the estimation of an unknown parameter when its probability distribution is not known. It is also one of the common techniques for the estimation of an MT's location, which is given by [30]

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \left\{ R_{es}(\mathbf{x}) \right\} \tag{32}$$

$$= \arg\min_{\mathbf{x}} \left\{ \sum_{i=1}^{N} \beta_i \Big(\hat{d}_i - ||\mathbf{x} - \mathbf{x}_i|| \Big)^2 \right\} , \tag{33}$$

where $R_{es}(\mathbf{x})$ is the residual error corresponding to MT location $\mathbf{x}$. Some weights β_i can be used to characterize the reliability of each link, which yields a weighted least squares (WLS) solution. If no reliability information is available, $\beta_i = 1$ for all i. Minimizing the non-linear expression in (33) requires numerical search methods such as the steepest descent [31] or the Gauss-Newton techniques [2], which may be computationally costly and require good initialization in order to avoid converging to the local minima of the loss function [2].

B. Matrix Representation of the Non-Linear Model

We may represent the non-linear expressions in (8) in matrix form. After some manipulation, we may write them as [32]

$$\mathbf{A}_1 \boldsymbol{\theta} = \frac{1}{2} \mathbf{p}_1 , \tag{34}$$

where

$$\mathbf{A}_1 = \begin{bmatrix} x_1 & y_1 & -0.5 \\ x_2 & y_2 & -0.5 \\ \vdots & \vdots & \\ x_N & y_N & -0.5 \end{bmatrix} ,$$

$$\boldsymbol{\theta} = \begin{bmatrix} x \\ y \\ s \end{bmatrix} , \mathbf{p}_1 = \begin{bmatrix} k_1 - \hat{d}_1^2 \\ k_2 - \hat{d}_2^2 \\ \vdots \\ k_n - \hat{d}_N^2 \end{bmatrix} . \tag{35}$$

with $s = x^2 + y^2$ being a part of the vector of unknown variables. In Section V-E, we will show how it can be used as a constraint to solve for $\mathbf{x}$.

After some further mathematical manipulation of (34) and (35), we may obtain an alternative LS solution as follows

$$\mathbf{A}_2 \mathbf{x} = \frac{1}{2} \mathbf{p}_2 , \ \hat{\mathbf{x}} = \frac{1}{2} (\mathbf{A}_2^T \mathbf{A}_2)^{-1} \mathbf{A}_2^T \mathbf{p}_2 , \tag{36}$$

where

$$\mathbf{A}_2 = \begin{bmatrix} x_1 & x_2 & ... & x_N \\ y_1 & y_2 & ... & y_N \end{bmatrix}^T , \ \mathbf{p}_2 = \begin{bmatrix} s + k_1 - \hat{d}_1^2 \\ s + k_2 - \hat{d}_2^2 \\ \vdots \\ s + k_n - \hat{d}_N^2 \end{bmatrix} . \tag{37}$$

Note that the above LS solution obtains $\mathbf{x}$ in terms of s, which results in a quadratic expression. Hence, a root-selection method can be employed to find $\mathbf{x}$ [26]. However, due to the inconsistency of equations, (36) is inaccurate. Nevertheless, it may be used as an initial location estimate to enhance the localization performance of more accurate (yet more complex) algorithms as will be discussed in the later sections.

C. Linearization of NLS Solution Through Taylor's Series Expansion

The non-linear function $\mathbf{d}(\mathbf{x})$ in (3) can be linearized around a reference point $\mathbf{x}_0$ using Taylor series expansion. If the higher order terms are neglected, we have [33]

$$\mathbf{d}(\mathbf{x}) \approx \mathbf{d}(\mathbf{x}_0) + \mathbf{H}_0 (\mathbf{x} - \mathbf{x}_0) , \tag{38}$$

where the Jacobian matrix of $\mathbf{d}(\mathbf{x})$ around $\mathbf{x}_0$ is given by

$$\mathbf{H}_0 = \begin{bmatrix} \frac{\partial d_1}{\partial x} & \frac{\partial d_2}{\partial x} & \cdots & \frac{\partial d_N}{\partial x} \\ \frac{\partial d_1}{\partial y} & \frac{\partial d_2}{\partial y} & \cdots & \frac{\partial d_N}{\partial y} \end{bmatrix}^T_{\mathbf{x}=\mathbf{x}_0} . \tag{39}$$

Note that the reference point $\mathbf{x}_0$ should be chosen sufficiently close to the true location in order for (38) to be valid. By substituting (38) into (33), we have a linear system which can be written in a matrix form and solved using a linear LS estimator. A more accurate iterative technique may use this LS estimate as an intermediate estimate, plug it into (38) to re-linearize the system around it, and iterate until convergence [31].

D. An Alternative Linear Least Squares Solution

The non-linear model discussed in previous sections contains the parameter s which is quadratic in x and y. In order to obtain a linear model, an alternative technique was proposed in [34] for canceling out these non-linear terms. By fixing expressions for the rth FT in (8), subtracting it from the rest of the equations for $i = 1, 2, ..., N$ $(i \neq r)$, and re-arranging the terms, we have the following linear model

$$\mathbf{A}_3 \mathbf{x} = \frac{1}{2} \mathbf{p}_3 , \tag{40}$$

with

$$\mathbf{A}_3 = \begin{bmatrix} x_1 - x_r & y_1 - y_r \\ x_2 - x_r & y_2 - y_r \\ \vdots & \vdots \\ x_N - x_r & y_N - y_r \end{bmatrix} , \mathbf{p}_3 = \begin{bmatrix} \hat{d}_r^2 - \hat{d}_1^2 - k_{r,1} \\ \hat{d}_r^2 - \hat{d}_2^2 - k_{r,2} \\ \vdots \\ \hat{d}_r^2 - \hat{d}_N^2 - k_{r,N} \end{bmatrix} , \tag{41}$$

where $k_{r,i} = k_r - k_i$, and r is the reference FT that is used to obtain the linear model. Note that the non-linear terms x^2 and y^2 in $\mathbf{p}_2$ of (36) are canceled out in $\mathbf{p}_3$. From the above expressions, the LS solution for $\mathbf{x}$ can be written as (call it *LLS-1*)

$$\mathbf{x} = \frac{1}{2} (\mathbf{A}_3^T \mathbf{A}_3)^{-1} \mathbf{A}_3^T \mathbf{p}_3 . \tag{42}$$

Note that while (8) defines a circle around each FT, the x^2 and y^2 terms are canceled in (40), resulting in linear expressions

Authorized licensed use limited to: Xian Jiaotong University. Downloaded on October 19,2021 at 12:08:56 UTC from IEEE Xplore. Restrictions apply.

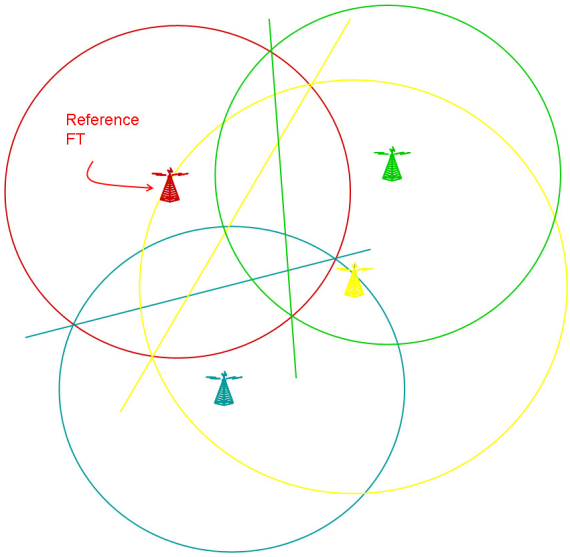


Fig. 4. The illustration of circles and lines for the non-linear and linear models, respectively.

that can be seen as the lines connecting the intersection points (if any) of the pairs of circles. An illustration of the geometric representations of the non-linear and linear models is given in Fig. 4.

In order to get more insight about the accuracy of *LLS-1* estimator, it is worth to analyze the perturbation in the vector $\mathbf{p}_3$. By replacing (1) in $\mathbf{p}_3$ and assuming that bias terms are zero, we have

$$\mathbf{p}_3 = \mathbf{p}_3^{(c)} + \mathbf{p}_3^{(n)} \ , \tag{43}$$

where the constant and noisy components are given by

$$\mathbf{p}_3^{(c)} = \begin{bmatrix} d_r^2 - d_1^2 - k_r + k_1 \\ d_r^2 - d_2^2 - k_r + k_2 \\ \vdots \\ d_r^2 - d_N^2 - k_r + k_N \end{bmatrix} , \tag{44}$$

$$\mathbf{p}_3^{(n)} = \begin{bmatrix} 2d_r n_r - 2d_1 n_1 + n_r^2 - n_1^2 \\ 2d_r n_r - 2d_2 n_2 + n_r^2 - n_2^2 \\ \vdots \\ 2d_r n_r - 2d_N n_N + n_r^2 - n_N^2 \end{bmatrix} , \tag{45}$$

Note that while we got rid of the quadratic s term, the number of noisy terms in $\mathbf{p}_3^{(n)}$ increased. In particular, we may claim that the accuracy of the above algorithm will degrade as the MT moves away from rth FT due to the distance-dependent noise terms at each link. If $\mathbf{p}_3^{(n)} \to 0$, all the lines in Fig. 4 will intersect at a single point. For theoretical derivation of the MSE for *LLS-1*, the reader is referred to [35].

1) Averaging Techniques: The *LLS-1* estimator for (40) utilizes the measurements $\hat{d}_i$, $i = 1, \ldots, N$, only through the terms $\hat{d}_r^2 - \hat{d}_i^2$, for $i = 1, \ldots, N$ and $i \neq r$. Therefore, the measurement set for *LLS-1* effectively becomes

$$\tilde{d}_i = \hat{d}_r^2 - \hat{d}_i^2, \quad i = 1, \ldots, N, \quad i \neq r \ . \tag{46}$$

Note that since the effective measurements in (46) become different than the measurements in (1), as also implied by Fig. 4, the corresponding CRLB will be different, which are derived in [36].

In another LLS approach proposed in [20] (call it *LLS-2*), $N \times (N-1)/2$ linear equations are obtained by subtracting each individual equation from all of the other equations. In other words, in the *LLS-2* technique, the following observations are employed for position estimation:

$$\check{d}_{ij} = \hat{d}_i^2 - \hat{d}_j^2 \ , \quad i, j = 1, 2, \ldots, N, \ i < j \ . \tag{47}$$

Similar to the *LLS-1*, the linear LS solution as in (42) is used in order to obtain the position of the target node in the *LLS-2* technique.

In a third LLS technique proposed in [37] (call it *LLS-3*), instead of obtaining the difference of the equations directly as in the *LLS-1* and *LLS-2* approaches, the average of the measurements is obtained first, and this average is subtracted from all the equations resulting in N linear relations. Then, the linear LS solution as in (42) is obtained for the position of the target node. The observation set employed in the *LLS-3* technique can be expressed as

$$\bar{d}_i = \hat{d}_i^2 - \frac{1}{N} \sum_{j=1}^{N} \hat{d}_j^2 \ , \quad i = 1, 2, \ldots, N \ . \tag{48}$$

2) Reference FT Selection: The *LLS-1* solution in (40) selects an arbitrary FT as the reference FT. However, observing the noisy terms in $\mathbf{p}_3^{(n)}$ given in (45), *all* the rows of the vector $\mathbf{p}_3^{(n)}$ depend on the true distance between the MT and the reference FT. If the FT is away from the MT location, this implies that all the elements of vector $\mathbf{p}_3$ will be more noisy, degrading the localization accuracy. Hence, how the reference FT is selected may considerably affect the estimator's mean square error (MSE). A simple method to select the reference FT for improved location accuracy in LOS scenarios is to choose it so that its measured distance is the smallest among all the distance measurements. The index of the reference FT that has the smallest measured distance is given by [38]

$$r = \arg\min_i \{\hat{d}_i\} \ , \quad i = 1, 2, \ldots, N \ . \tag{49}$$

Then, the matrix $\mathbf{A}_3$ and the vector $\mathbf{p}_3$ can be obtained using the selected reference FT (FT-r), and we refer the resulting estimator as LLS with reference selection (*LLS-RS*). For example, in Fig. 1, FT-1 is used to obtain the linear model from non-linear expressions since $\hat{d}_1$ is the minimum among all the measured distances.

3) Utilizing the Covariance Matrix: While the reference FT selection discussed above improves the location accuracy, it does not account for the correlation between the rows of the vector $\mathbf{p}_3^{(n)}$, which become correlated during the linearization process. As discussed in [39], the optimum estimator in the presence of correlated observations is given by an ML estimator. First, consider the following modification of the relationship in (43) for a LOS scenario

$$\mathbf{p}_3 = \mathbf{A}_3 \tilde{\mathbf{x}} + \mathbf{p}_3^{(n)} \ , \tag{50}$$

Authorized licensed use limited to: Xian Jiaotong University. Downloaded on October 19,2021 at 12:08:56 UTC from IEEE Xplore. Restrictions apply.

where $\tilde{\mathbf{x}}$ is the actual location of the MT, and hence $\mathbf{p}_3^{(c)} = \mathbf{A}_3\tilde{\mathbf{x}}$. Then, based on (50), the MLE[2] for this linear model can be written as [39]

$$\hat{\mathbf{x}} = (\mathbf{A}_3^T\mathbf{C}^{-1}\mathbf{A}_3)^{-1}\mathbf{A}_3^T\mathbf{C}^{-1}\mathbf{p}_3 \ , \tag{51}$$

where $\mathbf{C} = \mathrm{Cov}(\mathbf{p}_3^{(n)})$ is the covariance matrix of vector $\mathbf{p}_3^{(n)}$. When all the FTs are in LOS, the covariance matrix of vector $\mathbf{p}_n$ can be derived as

$$\mathbf{C} = 4d_r^2\sigma^2 + 2\sigma^4 + \mathrm{diag}\Big\{4\sigma^2 d_1^2 + 2\sigma^4, ..., 4\sigma^2 d_i^2 + 2\sigma^4, ..., 4\sigma^2 d_N^2 + 2\sigma^4\Big\} \ , \tag{52}$$

with $i \in \{1, 2, ..., N\}$, $i \neq r$, and where $\mathrm{diag}\{\lambda_1, ..., \lambda_N\}$ is a diagonal matrix obtained by placing λ_i on the diagonal of an $(N-1)\times(N-1)$ zero matrix $\forall i$. Note that since d_i are not available in practice, the noisy measurements $\hat{d}_i$ can be used to evaluate the covariance matrix.

4) Simulation Results: Monte-Carlo simulations are performed in order to compare the different LLS estimators. As in Fig. 2, four FTs are positioned on the corners of a square room. The simulation results in Fig. 5(a) show the 2-D MSE for *LLS-1* technique, where the FT-1 is used as a reference FT. We observe that the MSE tends to be smaller when the MT is closer to FT-1. For the simulation results in Fig. 5(b), the MT location $\tilde{\mathbf{x}}$ is changed with 10 meter intervals within $[-40, 40]$ m both in x and y directions, yielding a 9×9 grid of possible MT locations. The MSE of different techniques are simulated at each location on the grid, and then averaged over all the MT locations on the grid. The results show that the *LLS-1* performs worst compared to all the other techniques. The *LLS-2* and *LLS-3* techniques perform slightly better than *LLS-1*, and their MSEs are identical. However, they are both beaten by the *LLS-RS* technique. The MLE performs slightly better than that of *LLS-RS* and very close to the CRLB.

E. Constrained Weighted Least Squares

A constrained weighted least squares (CWLS) approach was presented in [32] which operates on (34) to find the MT location[3]. More specifically, it uses the relationship between s and $\mathbf{x}$ as a constraint on the LS solution, and develops a solution based on a Lagrange multiplier. The constraint optimization problem is formulated as [32]

$$\hat{\boldsymbol{\theta}}_{cw} = \arg\min_{\boldsymbol{\theta}} (\mathbf{A}_1\boldsymbol{\theta} - \mathbf{p}_1)^T\mathbf{W}(\mathbf{A}_1\boldsymbol{\theta} - \mathbf{p}_1) \ , \tag{53}$$

subject to the constraint $s = x^2 + y^2$, i.e.,

$$\mathbf{q}^T\boldsymbol{\theta} + \boldsymbol{\theta}^T\mathbf{P}\boldsymbol{\theta} = 0 \ , \tag{54}$$

where $\mathbf{A}_1$, $\boldsymbol{\theta}$, and $\mathbf{p}_1$ are as in (35), and

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \ , \text{ and } \mathbf{q} = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} \ . \tag{55}$$

[2]Note that in order to have the MLE as in (51), the elements of $\mathbf{p}_n$ should be zero-mean and Gaussian distributed random variables. While there are some non-Gaussian terms (i.e., the noise-square terms) in $\mathbf{p}_n$, they are assumed to be negligible, or fit closely to a Gaussian distribution to obtain the MLE.

[3]For a more detailed discussion on different constrained LS algorithms for AOA, RSS, TDOA, and hybrid techniques, the reader is referred to [40].

In order to determine the optimal weighting matrix $\mathbf{W}$, the authors examine the disturbance in $\mathbf{p}_1$. At high signal-to-noise ratios (SNRs), we have [32]

$$\hat{d}_i^2 = (d + n_i)^2 \approx d_i^2 + 2d_i n_i \text{ for } i = 1, 2, ..., N \ , \tag{56}$$

which implies a disturbance $\epsilon_i = \hat{d}_i^2 - d_i^2 = 2d_i n_i$, and can be represented in vector form as $\epsilon = [2d_1n_1, 2d_2n_2, ..., 2d_Nn_N]^T$. The covariance matrix of the disturbance is given by [32]

$$\boldsymbol{\Psi} = \mathrm{E}[\epsilon\epsilon^T] = \mathbf{BQB} \ , \tag{57}$$

with $\mathbf{B} = \mathrm{diag}(2d_1, 2d_2, ..., 2d_N)$, and the optimal weighting matrix becomes $\mathbf{W} = \boldsymbol{\Psi}^{-1}$. Note that it depends on the actual distances $\{d_i\}$ between the MT and the FTs which are unknown, and an approximate weighting matrix can be obtained using $\hat{\mathbf{B}} = \mathrm{diag}(2\hat{d}_1, 2\hat{d}_2, ..., 2\hat{d}_N)$, instead of $\mathbf{B}$.

The CWLS problem in (53) and (54) can then be solved by minimizing the Lagrangian as follows [32]

$$\mathcal{L}(\boldsymbol{\theta}, \lambda) = (\mathbf{A}_1\boldsymbol{\theta} - \mathbf{p}_1)^T\boldsymbol{\Psi}^{-1}(\mathbf{A}_1\boldsymbol{\theta} - \mathbf{p}_1) + \lambda(\mathbf{q}^T\boldsymbol{\theta} + \boldsymbol{\theta}^T\mathbf{P}\boldsymbol{\theta}) \ , \tag{58}$$

where λ is a Lagrange multiplier. It was shown in [32] that either a global or a local solution to (58) is given by

$$\hat{\boldsymbol{\theta}}_{cw} = (\mathbf{A}_1^T\boldsymbol{\Psi}^{-1}\mathbf{A}_1 + \lambda\mathbf{P})^{-1}\left(\mathbf{A}_1^T\boldsymbol{\Psi}^{-1}\mathbf{p}_1 - \frac{\lambda}{2}\mathbf{q}\right) \ , \tag{59}$$

with λ being determined from a five-root equation.

In another related work, the authors propose a covariance shaping LS (CSLS) technique for location estimation, which yields good performance compared to other LS estimators at low SNRs [41].

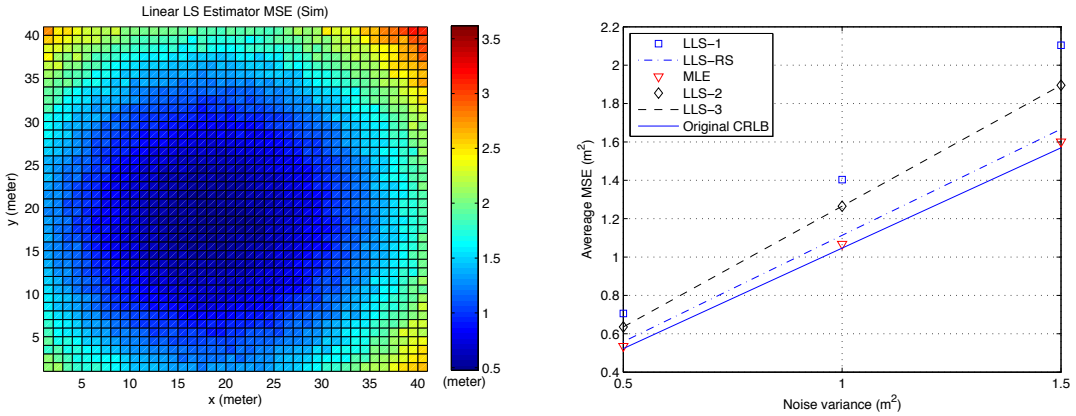
VI. NLOS Scenarios: Fundamental Limits and ML Solutions

In typical environments, especially in indoor scenarios, it may be possible that the LOS between the MT and some of the FTs may be obstructed (i.e., $b_i > 0$ for some i). These NLOS FTs may seriously degrade the localization accuracy. Simplest way of NLOS mitigation is achieved by identifying and discarding the NLOS FTs, and estimating the MT location by using one of the LOS techniques discussed in the previous section. However, there is always the possibility of false-alarms (identifying a LOS FT as NLOS) and missed-detections (identifying an NLOS FT as LOS) which degrade the localization accuracy. In this section, we review alternative NLOS mitigation techniques reported in the literature. Firstly, the ML based techniques and the CRLBs in NLOS scenarios will be discussed.

A. ML Based Algorithms

ML approaches for NLOS mitigation were discussed in [19], [23], which require prior knowledge regarding the distribution of NLOS bias. For example, [19] provides an ML solution for the position of the MT with the assumption that the NLOS bias b_i is exponentially distributed with parameter λ_i. Since in most cases NLOS bias is much larger than

Authorized licensed use limited to: Xian Jiaotong University. Downloaded on October 19,2021 at 12:08:56 UTC from IEEE Xplore. Restrictions apply.



(a) 2-D MSE of the LLS-1 in a LOS scenario.

(b) Simulation results for different linear LS estimators in LOS scenario.

Fig. 5. Comparison of the MSEs of different linear LS location estimation techniques averaged over a grid.

the Gaussian measurement error, the following simplifying assumption was made

$$\hat{d}_i = d_i + \begin{cases} b_i \ , & i = 1, 2, ..., N_{NL} \ , \\ n_i \ , & i = N_{NL}+1, ..., N, \end{cases} \tag{60}$$

where N_{NL} is the number of NLOS FTs, and $N_L = N - N_{NL}$ denotes the number of LOS FTs. Thus, a simplified ML solution is given as follows [19]

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \left\{ \sum_{i=1}^{N_{NL}} \lambda_i (\hat{d}_i - d_i) + \sum_{i=N_{NL}+1}^{N} \frac{(\hat{d}_i - d_i)^2}{2\sigma_i^2} \right\} . \tag{61}$$

It is also possible to obtain the exact decision rule by considering the summation of Gaussian and exponential random variables, which has the following probability density function

$$P(x) = \lambda \exp\Big(-\lambda\big(x - \lambda\sigma^2/2\big)\Big) Q\Big(\lambda\sigma - \frac{x}{\sigma}\Big) \ , \tag{62}$$

where $Q(\alpha) = \frac{1}{\sqrt{2\pi}} \int_{\alpha}^{\infty} \exp(-x^2/2) \mathrm{d}x$ denotes the Q-function, and the exact ML solution becomes [19]

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \left\{ \sum_{i=1}^{N_{NL}} \lambda_i \Big(\hat{d}_i - d_i - \lambda_i \sigma_i^2/2\Big) - \sum_{i=1}^{N_{NL}} \log\left[Q\Big(\lambda_i \sigma_i - \frac{\hat{d}_i - d_i}{\sigma_i}\Big)\right] + \sum_{i=N_{NL}+1}^{N} \frac{(\hat{d}_i - d_i)^2}{2\sigma_i^2} \right\} . \tag{63}$$

Another NLOS mitigation technique for TOA based systems based on the ML approach is introduced in [23]. The authors consider several hypothesis for different sets of FTs, and then, utilizing the ML principle, the best set (that is assumed to be composed of LOS FTs) is selected for location estimation. The hypothesis index estimate for the best FT set is derived as [23]

$$\hat{i} = \arg\min_{i} \left[\ln\gamma^{-1}(i) + \sum_{k \in S_i^{LOS}} \frac{N_{trn}}{2} \frac{\hat{\sigma}_k^2(i)}{\sigma_k^2} \right] , \tag{64}$$

where S_i^{LOS} denotes the ith set of MTs which are hypothesized to be LOS, and $\gamma(i)$ are assigned according to the *a-priori* probability of each hypothesis (equivalent to 1 if no information available). Also,

$$\hat{\sigma}_k^2(i) = \frac{1}{N_{\mathrm{trn}}} \sum_{m=1}^{N_{\mathrm{trn}}} (t_{k,m} - ||\hat{\mathbf{x}}(i) - \mathbf{x}_k||)^2 \ , \tag{65}$$

denotes the estimated variance of the N_{trn} TOA measurements tied with the kth FT under the ith hypothesis, and $t_{k,m}$ denotes the mth TOA measurement at the kth FT. Note that the above approach requires buffering of N_{trn} TOA measurements for the purpose of obtaining the noise statistics at a particular FT. Once the set of LOS FTs is selected using the ML principle, the MT location is estimated using only these FTs and the ML algorithm. Simulation results in [23] show that this yields better accuracy than residual weighting (Rwgh) technique introduced in [18] (to be discussed in Section VII-B), and slightly worse than when only the true LOS FTs are used in localization. Also, at low SNR and small NLOS bias values, simulation results imply it may be better not to employ any NLOS mitigation in order not to degrade the accuracy.

B. Cramer-Rao Lower Bound

In NLOS scenarios, the CRLB depends on if there is any prior information available about the NLOS bias. First consider that there is no prior information about the NLOS bias, except that only the NLOS FTs are assumed to be perfectly known. Then, an extended version of the FIM in (19) is given by [42]

$$\mathbf{I}(\mathbf{x}_b) = \tilde{\mathbf{A}} \mathbf{I}(\mathbf{d}) \tilde{\mathbf{A}}^T \ , \tag{66}$$

where

$$\mathbf{x}_b = [x, \ y, \ b_1, \ b_2, ...b_{N_{NL}}]^T \ , \tag{67}$$

Authorized licensed use limited to: Xian Jiaotong University. Downloaded on October 19,2021 at 12:08:56 UTC from IEEE Xplore. Restrictions apply.

is an $(N_{NL}+2)\times 1$ vector of unknown parameters (including the NLOS bias values) with

$$\tilde{\mathbf{A}} = \frac{\partial \mathbf{d}}{\partial \mathbf{x}_b} = \begin{bmatrix} \frac{\partial d_1}{\partial x} & \frac{\partial d_2}{\partial x} & \cdots & \frac{\partial d_{N_{NL}}}{\partial x} & \cdots & \frac{\partial d_N}{\partial x} \\ \frac{\partial d_1}{\partial y} & \frac{\partial d_2}{\partial y} & \cdots & \frac{\partial d_{N_{NL}}}{\partial y} & \cdots & \frac{\partial d_N}{\partial y} \\ \frac{\partial d_1}{\partial b_1} & \frac{\partial d_2}{\partial b_1} & \cdots & \frac{\partial d_{N_{NL}}}{\partial b_1} & \cdots & \frac{\partial d_N}{\partial b_1} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial d_1}{\partial b_{N_{NL}}} & \frac{\partial d_2}{\partial b_{N_{NL}}} & \cdots & \frac{\partial d_{N_{NL}}}{\partial b_{N_{NL}}} & \cdots & \frac{\partial d_N}{\partial b_{N_{NL}}} \end{bmatrix}, \tag{68}$$

and

$$\mathbf{I}(\mathbf{d}) = \mathrm{E}_{\mathbf{d}}\left[\frac{\partial \ln f(\hat{\mathbf{d}}|\mathbf{d})}{\partial \mathbf{d}}\left(\frac{\partial \ln f(\hat{\mathbf{d}}|\mathbf{d})}{\partial \mathbf{d}}\right)^T\right]. \tag{69}$$

As discussed in [42], $\tilde{\mathbf{A}}$ can be written in terms of its LOS and NLOS components as

$$\tilde{\mathbf{A}} = \begin{bmatrix} \tilde{\mathbf{A}}_{NL} & \tilde{\mathbf{A}}_L \\ \mathbf{I}_{N_{NL}} & \mathbf{0}_{N_{NL},N_L} \end{bmatrix}, \tag{70}$$

where $\mathbf{I}_{N_{NL}}$ and $\mathbf{0}_{N_{NL},N_L}$ are an identity matrix of size $N_{NL}\times N_{NL}$ and a zero matrix of size $N_{NL}\times N_L$, respectively, and

$$\tilde{\mathbf{A}}_{NL} = \begin{bmatrix} \cos\alpha_1 & \cos\alpha_2 & \ldots & \cos\alpha_{N_{NL}} \\ \sin\alpha_1 & \sin\alpha_2 & \ldots & \sin\alpha_{N_{NL}} \end{bmatrix}, \tag{71}$$

$$\tilde{\mathbf{A}}_L = \begin{bmatrix} \cos\alpha_{N_{NL}+1} & \cos\alpha_{N_{NL}+2} & \ldots & \cos\alpha_N \\ \sin\alpha_{N_{NL}+1} & \sin\alpha_{N_{NL}+2} & \ldots & \sin\alpha_N \end{bmatrix}. \tag{72}$$

Similarly, $\mathbf{I}(\mathbf{d})$ can be written in terms of its NLOS and LOS components as [42]

$$\mathbf{I}(\mathbf{d}) = \begin{bmatrix} \mathbf{\Lambda}_{NL} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Lambda}_L \end{bmatrix}, \tag{73}$$

where $\mathbf{\Lambda}_{NL} = \mathrm{diag}(\sigma_1^{-2},...,\sigma_{N_{NL}}^{-2})$ and $\mathbf{\Lambda}_L = \mathrm{diag}(\sigma_{N_{NL}+1}^{-2},...,\sigma_{N_N}^{-2})$. After some manipulation, $\mathbf{I}(\mathbf{x}_b)$ can be obtained as [42]

$$\mathbf{I}(\mathbf{x}_b) = \begin{bmatrix} \tilde{\mathbf{A}}_{NL}\mathbf{\Lambda}_{NL}\tilde{\mathbf{A}}_{NL}^T + \tilde{\mathbf{A}}_L\mathbf{\Lambda}_L\tilde{\mathbf{A}}_L^T & \tilde{\mathbf{A}}_{NL}\mathbf{\Lambda}_{NL} \\ \mathbf{\Lambda}_{NL}\tilde{\mathbf{A}}_{NL}^T & \mathbf{\Lambda}_{NL} \end{bmatrix}. \tag{74}$$

Note that (74) depends both on NLOS and LOS signals. However, it was further proven in [42] that the CRLB for the MT location is given by

$$E\left[(\hat{\mathbf{x}}_b-\mathbf{x}_b)(\hat{\mathbf{x}}_b-\mathbf{x}_b)^T\right] \geq \mathbf{I}^{-1}(\mathbf{x}_b) = \left(\tilde{\mathbf{A}}_L\mathbf{\Lambda}_L\tilde{\mathbf{A}}_L^T\right)^{-1}. \tag{75}$$

In other words, the CRLB exclusively depends on the LOS signals if the NLOS FTs can be accurately identified. Hence, the ML estimator that can achieve the CRLB in NLOS scenarios first identifies the NLOS FTs, discards these measurements, and then obtains the location estimate using the LOS FTs, as illustrated in Fig. 6(a).

If there is further side information related to the statistics of the NLOS bias vector $\mathbf{b}$, a better positioning accuracy can be obtained. Then, the generalized CRLB (G-CRLB) can be written as [42]

$$E\left[(\hat{\mathbf{x}}_b-\mathbf{x}_b)(\hat{\mathbf{x}}_b-\mathbf{x}_b)^T\right] \geq \left(\mathbf{I}(\mathbf{x}_b) + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Omega} \end{bmatrix}\right)^{-1} \tag{76}$$

$$= \left(\begin{bmatrix} \tilde{\mathbf{A}}_{NL}\mathbf{\Lambda}_{NL}\tilde{\mathbf{A}}_{NL}^T + \tilde{\mathbf{A}}_L\mathbf{\Lambda}_L\tilde{\mathbf{A}}_L^T & \tilde{\mathbf{A}}_{NL}\mathbf{\Lambda}_{NL} \\ \mathbf{\Lambda}_{NL}\tilde{\mathbf{A}}_{NL}^T & \mathbf{\Lambda}_{NL}+\mathbf{\Omega} \end{bmatrix}\right)^{-1}, \tag{77}$$

where $\mathbf{\Omega} = \mathrm{diag}(\tilde{\sigma}_1^2,...,\tilde{\sigma}_{N_{NL}}^2)$, and $\tilde{\sigma}_i^2$ can be interpreted as the variance[4] of b_i. As an upper bound on the G-CRLB, when the variances $\tilde{\sigma}_i^2$ are infinitely large, $\mathbf{\Omega}\rightarrow\mathbf{0}$, and G-CRLB is reduced to the CRLB (since there is practically no information available on b_i). The estimator that asymptotically achieves the G-CRLB is given by the maximum a-posteriori (MAP) estimator, and it employs the statistics of the NLOS biases as illustrated in Fig. 6(b).

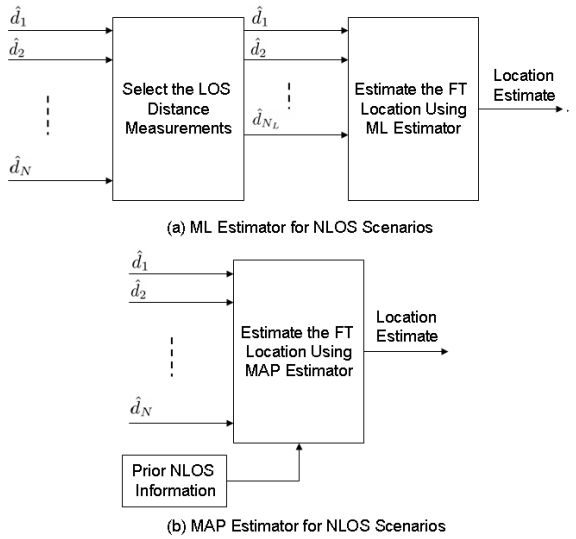


Fig. 6. Illustration of block diagrams for (a) ML estimator and (b) MAP estimator in NLOS scenarios. In part (a), without loss of generality, it is assumed that the first N_L measurements are the LOS measurements.

VII. NLOS Scenarios: Least Squares Techniques

The LS techniques for location estimation can be tuned to suppress the NLOS bias effects, e.g., through some appropriate weighting. In this section, weighted least squares approaches as well as the residual weighting algorithm will be briefly reviewed.

A. Weighted Least Squares

A simple way to mitigate the effects of NLOS FTs is to give less emphasis to corresponding NLOS terms in the LS solution. In [19], [30], with the assumption that the variances of the distance measurements are larger for NLOS FTs, the

[4]For Gaussian distributed NLOS bias, it is strictly the variance of the NLOS bias.

Authorized licensed use limited to: Xian Jiaotong University. Downloaded on October 19,2021 at 12:08:56 UTC from IEEE Xplore. Restrictions apply.

TABLE II
STEPS OF THE RESIDUAL WEIGHTING ALGORITHM IN [18].

1) Form $N_{cb} = \sum_{i=3}^{N} {}_N C_i$ range measurement combinations, where ${}_N C_i$ denotes the total number of combinations with i FTs selected from a total of N FTs. Also let $\{S_k | k = 1, 2, ..., N_{cb}\}$ denote the set of FTs for the kth combination.
2) For each set of combinations S_k, compute an intermediate LS location estimate as follows

$$\hat{\mathbf{x}}_k = \arg\min_{\mathbf{x}} \left\{ R_{es}(\mathbf{x}; S_k) \right\} , \quad (83)$$

where $R_{es}(\mathbf{x}; S_k)$ is the residual error when only the FTs in set S_k are used for calculating the MT location. Also define the normalized residual

$$\tilde{R}_{es}(\hat{\mathbf{x}}_k; S_k) = R_{es}(\hat{\mathbf{x}}_k; S_k)/|S_k| , \quad (84)$$

where $|S_k|$ denotes the size of S_k.
3) Find the final location estimate by weighting the intermediate location estimates with their corresponding normalized residual errors:

$$\hat{\mathbf{x}} = \frac{\sum_{k=1}^{N_{cb}} \hat{\mathbf{x}}_k \left[\tilde{R}_{es}(\hat{\mathbf{x}}_k; S_k) \right]^{-1}}{\sum_{k=1}^{N_{cb}} \left[\tilde{R}_{es}(\hat{\mathbf{x}}_k; S_k) \right]^{-1}} . \quad (85)$$

inverses of these variances are used as a reliability metric β_i in (33). This is actually derived from the ML solution as follows. For the ML algorithm, the location estimate is given by

$$\mathbf{x}_{ML} = \arg\max_{\mathbf{x}} \mathrm{p}(\hat{\mathbf{d}}|\mathbf{x}) , \quad (78)$$

where

$$\mathrm{p}(\hat{\mathbf{d}}|\mathbf{x}) = \mathrm{p_n}(\hat{\mathbf{d}} - \mathbf{d}|\mathbf{x}) . \quad (79)$$

If the noise is Gaussian distributed, we have

$$\mathrm{p_n}(n) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{n^2}{2\sigma_i^2} \right) . \quad (80)$$

Then, the joint probability function becomes

$$\mathrm{p}(\hat{\mathbf{d}}|\mathbf{x}) = \frac{1}{(2\pi)^{N/2} \prod_{i=1}^{N} \sigma_i} \exp\left(-\sum_{i=1}^{N} \frac{(\hat{d}_i - ||\mathbf{x} - \mathbf{x}_i||)^2}{2\sigma_i^2} \right) . \quad (81)$$

Upon further manipulation of (81), the ML solution becomes equivalent to

$$\hat{\mathbf{x}}_{ML} = \arg\min_{\mathbf{x}} \sum_{i=1}^{N} \frac{(\hat{d}_i - ||\mathbf{x} - \mathbf{x}_i||)^2}{\sigma_i^2} , \quad (82)$$

which is equivalent to the WLS solution for $\beta_i = 1/\sigma_i^2$.

However, for a static MT, the variance of TOA measurements may not be significantly different for LOS and NLOS FTs. Still, the bias in NLOS distance measurements may degrade the localization accuracy. Hence, in [43], [44] an alternative weighting technique is proposed, which uses certain statistics of the multipath components of the received signals. In particular, kurtosis, mean excess delay, and root-mean-square (RMS) delay spread of the received signal are used to evaluate the likelihood value of the received signal to be LOS. The likelihood values are then used to evaluate the weighting parameters β_i.

B. Residual Weighting Algorithm

The Rwgh algorithm proposed in [18] is based on the observation that the residual error $R_{es}(\mathbf{x})$ is typically larger if NLOS FTs are used when estimating the MT location. By assuming that there are more than three FTs available, the Rwgh estimates the MT location as detailed in Table II.

It was shown in [18] through simulations that Rwgh performs better than choosing the location estimate with the minimum residual error (MRE). A sub-optimal version of Rwgh algorithm that has lower computational complexity was proposed in [45]. In that paper, instead of considering all the combinations of the FTs (which may be very large if N is large), first, all the combinations with $(N-1)$ FTs are considered in order to calculate the intermediate location estimates and the corresponding residuals. Then, among N different combinations, the FT which is not employed in the best estimator (i.e., corresponding to the combination with the smallest residual error) is discarded. The process iterates until a certain pre-determined stopping rule is reached (such as when a minimum number of FTs is reached, or, if the change in the residual error is small).

VIII. NLOS SCENARIOS: CONSTRAINED LOCALIZATION TECHNIQUES

In this section, a different class of NLOS mitigation algorithms which utilize some constraints associated with the NLOS measurements will be briefly reviewed.

A. Constrained LS Algorithm and Quadratic Programming

The two-step ML algorithm discussed in Section IV-C is not robust to NLOS effects. In [46], a quadratic programming (QP) technique for NLOS environments is developed. The mathematical programming is formulated as follows

$$\hat{\boldsymbol{\theta}}_{cw} = \arg\min_{\boldsymbol{\theta}} (\mathbf{A}_1\boldsymbol{\theta} - \mathbf{p}_1)^T \boldsymbol{\Psi}^{-1} (\mathbf{A}_1\boldsymbol{\theta} - \mathbf{p}_1) , \quad (86)$$

$$\text{s.t.} \quad \mathbf{A}_1\boldsymbol{\theta} \leq \mathbf{p}_1 , \quad (87)$$

where $\mathbf{A}_1$, $\mathbf{p}_1$, and $\boldsymbol{\theta}$ are as in (35). Note that (86) and (87) constitute a constrained LS (CLS) algorithm that can be solved using quadratic programming techniques[5]. The intuitive explanation of the CLS is that (86) finds a WLS solution to the MT location, while the constraint (87) relaxes the equality (which holds in LOS scenarios) into an inequality for the NLOS scenarios. In [46], a further refining stage is also introduced to incorporate the dependency between s and $\mathbf{x}$.

B. Linear Programming

In [20], [47], a linear programming approach was introduced which assumes perfect *a-priori* identification of LOS and NLOS FTs. As opposed to the identify&discard (IAD) type algorithms (to be discussed in Section X), it does not discard NLOS FTs, but uses them to construct a linear feasible region for the MT location. The location estimate is obtained using a linear programming technique that employs only the

[5]E.g., using the *quadprog* function in Matlab.

Authorized licensed use limited to: Xian Jiaotong University. Downloaded on October 19,2021 at 12:08:56 UTC from IEEE Xplore. Restrictions apply.

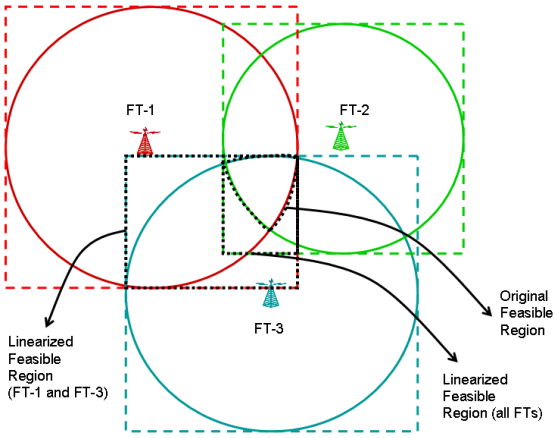


Fig. 7. Illustration of the constrained linear LS technique. The NLOS FTs are used to determine the feasible region. For original (non-linear) model, the feasible region is obtained from the intersections of the circles. For the linear model, the feasible region is obtained from the intersections of squares.

LOS FTs, with the constraint that it has to be within the feasible region.

First, [20] re-expresses the LS location estimator in (42) using linear programming with all the LOS FTs. The goal is to minimize a linear objective function with respect to a certain constraint, and it is shown that the performance is very similar to that of (42).

If the ith FT is identified to be NLOS, we have the non-linear constraint

$$||\mathbf{x} - \mathbf{x}_i|| \leq \hat{d}_i, \tag{88}$$

which were used in [46] to formulate a quadratic programming technique for two-step ML algorithm (without the assumption that the NLOS FTs are accurately identified). In order to linearize the constraints in (88), we can relax them as [48]

$$x - x_i \leq \hat{d}_i \ , \quad -x + x_i \leq \hat{d}_i \ , \tag{89}$$

$$y - y_i \leq \hat{d}_i \ , \quad -y + y_i \leq \hat{d}_i \ , \tag{90}$$

where $i = 1, 2, ..., N$. In essence, the above is equivalent to relaxing the circular constraints into rectangular (i.e. square) constraints for the purpose of linearizing the equations, as illustrated in Fig. 7. By defining some slack variables[6], (89) and (90) can be converted to equalities that define the feasible region. Then, the MT location is estimated by using only the LOS FTs for minimizing the objective function with the constraint that, the location estimate should be within the feasible region obtained using both the NLOS and LOS FTs.

C. Geometry-Constrained Location Estimation

In [49], a geometry-constrained location estimation (GLE) was proposed, which uses the two-step ML technique in Section IV-C with some additional parameters to incorporate the geometry of the FTs (only a scenario with three FTs was considered). Let the intersection points of the three circles in (8) be defined as $\mathbf{x}_A = [x_A, y_A]^T$, $\mathbf{x}_B = [x_B, y_B]^T$, $\mathbf{x}_C = [x_C, y_C]^T$. Then, a constrained cost function, which is referred as the *virtual distance* is defined as [49]

$$\gamma = \sqrt{\frac{1}{3}\left[||\mathbf{x} - \mathbf{x}_A||^2 + ||\mathbf{x} - \mathbf{x}_B||^2 + ||\mathbf{x} - \mathbf{x}_C||^2\right]} \ . \tag{91}$$

For an expected position $\mathbf{x}_e$ of the MT, we can calculate the *expected virtual distance* using (91) as $\gamma_e = \gamma + n_e$, with n_e denoting the noise in the expected virtual distance. The coordinates for $\mathbf{x}_e$ are chosen as [49]

$$\mathbf{x}_e = \omega_1 \mathbf{x}_A + \omega_2 \mathbf{x}_B + \omega_3 \mathbf{x}_C \ , \tag{92}$$

where the weights are obtained as

$$\omega_i = \frac{\sigma_i^2}{\sigma_1^2 + \sigma_2^2 + \sigma_3^2} \ , \text{with } i = 1, 2, 3 \ . \tag{93}$$

Basically, it is assumed that for NLOS FTs, the measurement variance will be larger. Hence, the weights defined in (93) will move $\mathbf{x}_e$ towards the center of the NLOS FT circle.

These geometric constraints are then incorporated into the two-step ML algorithm in Section IV-C by updating $\mathbf{A}_1$ and $\mathbf{p}_1$ in (35) as follows [49]

$$\tilde{\mathbf{A}}_1 = \begin{bmatrix} & \mathbf{A}_1 & \\ \gamma_x & \gamma_y & -0.5 \end{bmatrix} \ , \text{ and } \tilde{\mathbf{p}}_1 = \begin{bmatrix} \mathbf{p}_1 \\ \gamma_k - \gamma_e^2 \end{bmatrix} \ , \tag{94}$$

where

$$\gamma_x = \frac{1}{3}(x_A + x_B + x_C) \ , \tag{95}$$

$$\gamma_y = \frac{1}{3}(y_A + y_B + y_C) \ , \tag{96}$$

$$\gamma_k = \frac{1}{3}(x_A^2 + x_B^2 + x_C^2 + y_A^2 + y_B^2 + y_C^2) \ . \tag{97}$$

The geometric constraints are also incorporated into other variables as

$$\tilde{\mathbf{B}} = \text{diag}(d_1, d_2, d_3, \gamma) \ , \tag{98}$$

$$\tilde{\mathbf{n}} = [n_1, n_2, n_3, n_\gamma]^T \ . \tag{99}$$

Then, the two-step ML algorithm in [29] is employed to solve for the MT location using the variables updated with the geometric constraints.

D. Interior Point Optimization

In [33], an interior point optimization (IPO) method was proposed to find the optimum location estimate in the presence of NLOS bias. Using the linearization of the system using Taylor's series approximation as discussed in Section V-C, a linearized measurement vector is defined as follows [33]

$$\mathbf{y} = \mathbf{H}_0 \mathbf{x} + \mathbf{b} + \mathbf{n} \ , \tag{100}$$

where $\mathbf{H}_0$ is as defined in (39). If NLOS bias is neglected, the bias-free position estimate is given by [33]

$$\tilde{\mathbf{x}} = (\mathbf{H}_0^T \mathbf{Q}^{-1} \mathbf{H}_0)^{-1} \mathbf{H}_0^T \mathbf{Q}^{-1} \mathbf{y} \ . \tag{101}$$

[6]Slack variable is a variable which is used to turn an inequality into an equality (e.g., $x < 5 \Rightarrow x + s = 5$).

Authorized licensed use limited to: Xian Jiaotong University. Downloaded on October 19,2021 at 12:08:56 UTC from IEEE Xplore. Restrictions apply.

If the bias vector $\mathbf{b}$ is known, a more accurate *bias-free* location estimate is given by [33]

$$\hat{\mathbf{x}} = \tilde{\mathbf{x}} + \mathbf{V}\mathbf{b} \ , \tag{102}$$

where

$$\mathbf{V} = -(\mathbf{H}_0^T\mathbf{Q}^{-1}\mathbf{H}_0)^{-1}\mathbf{H}_0^T\mathbf{Q}^{-1} \ , \tag{103}$$

is a bias correction matrix. However, in reality, $\mathbf{b}$ is unknown and has to be estimated. In order to estimate $\mathbf{b}$ from (100), the observed bias metric is defined as [33]

$$\mathbf{z} = \mathbf{y} - \mathbf{H}_0\tilde{\mathbf{x}} \ , \tag{104}$$

which can be simplified to $\mathbf{z} = \mathbf{S}\mathbf{b} + \mathbf{w}$, where $\mathbf{S} = \mathbf{I} + \mathbf{H}_0\mathbf{V}$, and the *bias noise* is given by

$$\mathbf{w} = \mathbf{H}_0(\mathbf{x} - \hat{\mathbf{x}}) - \mathbf{n} \ . \tag{105}$$

Then, the following constrained optimization problem is defined to estimate the NLOS bias errors [33]

$$\hat{\mathbf{b}} = \arg\min_{\mathbf{b}} (\mathbf{z} - \mathbf{S}\mathbf{b})^T\mathbf{Q}_w^{-1}(\mathbf{z} - \mathbf{S}\mathbf{b}) \ , \tag{106}$$

$$\text{s.t.} \quad b_i \in B_i \ , \quad i = 1, 2, ..., N \ , \tag{107}$$

where $B_i = [l_i, u_i]$ are the *a-priori* information for the range of b_i lower-bounded by $l_i \geq 0$ and upper-bounded by u_i, and $\mathbf{Q}_w$ is the covariance matrix of $\mathbf{w}$.

In order to solve the constrained optimization problem in (106) and (107), an IPO technique was used in [33]. In particular, (106) and (107) are modified as

$$\hat{\mathbf{b}} = \arg\min_{\mathbf{b}} (\mathbf{z} - \mathbf{S}\mathbf{b})^T\mathbf{Q}_w^{-1}(\mathbf{z} - \mathbf{S}\mathbf{b}) \ , \tag{108}$$

$$\text{s.t.} \quad g_i(b_i) - s_i = 0, \text{ and } s_i > 0 \ , \ i = 1, ..., N \ , \tag{109}$$

where s_i is a slack variable, and $g_i(b_i)$ is a barrier function that satisfies $g_i(b_i) > 0 \ \forall b_i \in [l_i, u_i]$. A generally used smooth second order function that satisfies the requirement is $g_i(b_i) = (u_i - b_i)/(b_i - l_i)$. Then, (108) and (109) are solved by minimizing the following Lagrangian [33]

$$\mathcal{L}(\mathbf{b}, \lambda, \mathbf{s}) = (\mathbf{z} - \mathbf{S}\mathbf{b})^T\mathbf{Q}_w^{-1}(\mathbf{z} - \mathbf{S}\mathbf{b}) - \mu\sum_{i=1}^{N} \ln s_i - \lambda^T(\mathbf{g}(\mathbf{b}) - \mathbf{s}) \ , \tag{110}$$

where $\mathbf{g}(\mathbf{b})$ and $\mathbf{s}$ are obtained upon stacking $g_i(b_i)$ and s_i, respectively, into $N \times 1$ vectors. Note that the logarithmic barrier function

$$\mu\sum_{i=1}^{N} \ln s_i \ , \tag{111}$$

ensures that $s_i = g_i(b_i) > 0$ and bias error is always within $[l_i, u_i]$.

The solution to (110) can be obtained by differentiating (110) with respect to $\mathbf{b}$, λ, and $\mathbf{s}$, and solving them together to obtain $\mathbf{b}$. Once an estimate of the bias vector $\mathbf{b}$ is obtained, the authors employ the bias correction matrix in (103) to calculate the bias-free location using (102). The simulation results reported in [33] show that better accuracies can be obtained through IPO compared to Rwgh and iterative LS algorithms in NLOS scenarios.

IX. NLOS Scenarios: Robust Estimators for Localization

Robust estimators are commonly used to suppress the impact of outliers in a given data, and different classes of robust estimators have already been used in the literature for NLOS mitigation purposes. In below, few of the popular robust estimators considered for NLOS mitigation are briefly reviewed.

A. Huber M-Estimator

The *M*-estimators, which are "ML type" of estimators, are a class of robust estimators that have been considered for NLOS mitigation purposes. As discussed in the previous sections, the ML algorithm tries to maximize a function of the form

$$\prod_{i=1}^{N} f(x_i) \ , \tag{112}$$

which is equivalent to minimizing $\sum_{i=1}^{N} -\log \ f(x_i)$. In the presence of outliers, ML algorithm fails to yield accurate results. A generalized form of the ML algorithm is referred as the M-estimator, which was introduced by Huber in 1964, and aims to minimize

$$\sum_{i=1}^{N} \rho(x_i) \ , \tag{113}$$

where $\rho(.)$ is a convex function. For the Huber M-estimator, the $\rho(.)$ is defined as [50]

$$\rho(\nu) = \begin{cases} \nu^2/2 & |\nu| \leq \xi \ , \\ \xi|\nu| - \xi^2/2 & |\nu| > \xi \ , \end{cases} \tag{114}$$

which is not strictly convex, and therefore, minimization of the objective function yields multiple solutions which are close to each other [50].

In [51], M-estimator was used to estimate the MT location, which yields

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \left\{ \sum_{i=1}^{N} \rho\Big(\big(\hat{d}_i - ||\mathbf{x} - \mathbf{x}_i||\big)/\sigma_i\Big) \right\} . \tag{115}$$

Simulation results in [51] show that M-estimator outperforms the conventional LS estimator, especially for large NLOS bias errors. If bootstrapping technique is used in conjunction with M-estimator, the accuracy can be improved further [51].

B. Least Median Squares

In [37], a least median squares (LMS) technique was proposed for NLOS mitigation, which is one of the most commonly used robust fitting algorithms. It can tolerate as high as 50% of outliers in the absence of noise. The location estimate of the MT using the LMS solution is given by [37]

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \left\{ \text{med}_i\Big(\hat{d}_i - ||\mathbf{x} - \mathbf{x}_i||\Big)^2 \right\} \ , \tag{116}$$

where $\text{med}_i(\theta(i))$ is the median of $\theta(i)$ over all possible values of i. Since calculation of (116) is computationally intensive, [37] proposes a lower-complexity implementation

Authorized licensed use limited to: Xian Jiaotong University. Downloaded on October 19,2021 at 12:08:56 UTC from IEEE Xplore. Restrictions apply.

that uses random subsets of $\mathbf{x}_i$ to obtain several candidate $\hat{\mathbf{x}}$, and the one with the least median of residue is selected as the solution. It should be noted that [37] uses LMS algorithm in the context of security, where some of the measurements may be outliers in the presence of certain attacks to the localization system. A parallel approach that uses LMS algorithm for NLOS mitigation purposes is reported in [52].

C. Other Robust Estimation Options

Besides the M-estimation and LMS techniques discussed above, there are some other robust estimation techniques [53] which, to our best knowledge, have not been considered in detail for NLOS mitigation purposes. For example, least trimmed squares (LTS) aims to minimize the sum of squares for the smallest n of the residuals. S-estimators, on the other hand, is quiet robust to outliers; however, it is not as efficient as the M-estimator. It attempts to find a line minimizing a robust estimate of the scale of the residuals. Finally, MM-estimator can be considered as an M-estimator, which starts at the coefficients given by the S-estimator. With more computational complexity compared to the other two techniques, MM-estimator has both good robustness and good efficiency.

There are also some other parametric approaches for robust regression, where, the normal distribution is replaced with a heavy tailed distribution in order to model the outliers. For example, t-distribution can be a good distribution for modeling the outliers in a Bayesian robust regression algorithm [54]. Alternatively, a mixture of zero-mean Gaussian distributions (i.e., a contaminated normal distribution) with different variances can be considered to model the noise in the presence of NLOS bias, e.g.

$$n_i \sim (1-\epsilon)\mathcal{N}(0,\sigma^2) + \epsilon\mathcal{N}(0, c\sigma^2) \ , \tag{117}$$

where the second Gaussian distribution is intended to capture the outliers, and $\epsilon < 1$ is a small number (typically smaller than 0.1) that characterizes the impact of the outliers together with the constant term c.

X. NLOS Scenarios: Identify and Discard Based Techniques

As discussed in the beginning of Section VI, one of the simplest techniques to mitigate the NLOS effects is to identify the NLOS FTs and discard them during localization (i.e., find the MT location using only the LOS FTs). In fact, the ML estimator of [42] illustrated in Fig. 6(a) and the ML estimator of [23] discussed in Section VI-A are also IAD type of estimators. Moreover, the WLS technique discussed in Section VII-A also becomes an IAD type of technique if the weights are set to 0 for NLOS FTs, and 1 for LOS FTs. A common problem in all these approaches is accurate identification of the NLOS FTs. In this section, we will review another IAD based technique which uses a residual test algorithm to identify the NLOS FTs.

A. Residual Test Algorithm

Residual test (RT) algorithm proposed in [28] falls into the group of algorithms where the MT is localized using only the LOS FTs. Hence, the NLOS FTs have to be correctly identified and discarded, which is achieved as follows. Firstly, using the AML algorithm discussed in Section IV-D, by employing different combinations of FTs,[7]

$$S_0 = \sum_{i=3}^{N} {}_N C_i \ , \tag{118}$$

different location estimates $\hat{\mathbf{x}}_k$ are computed ($k \in \{1, 2, ..., S_0\}$).

For each k, the squares of the normalized residuals[8] are computed as

$$\chi_x^2(k) = \frac{[\hat{x}_k - \hat{x}_{S_0}]^2}{I_x(k)} \ , \chi_y^2(k) = \frac{[\hat{y}_k - \hat{y}_{S_0}]^2}{I_y(k)} \ , \tag{119}$$

where $I_x(k)$ and $I_y(k)$ are obtained from (17), and are the CRLBs[9] for x and y dimensions for the kth hypothesis. Without loss of generality, S_0 is the case when all the FTs are used in localization.

If all the FTs in the kth hypothesis are LOS, from (119), we have $\chi_x(k) \sim \mathcal{N}(0,1)$ and $\chi_y(k) \sim \mathcal{N}(0,1)$. Hence, $\chi_x^2(k)$ and $\chi_y^2(k)$ have centralized Chi-square distribution with one degree of freedom. On the other hand, if there is at least one NLOS FT in the kth hypothesis, both random variables have non-centralized Chi-square distributions with non-centrality parameter depending on the NLOS bias.

This observation suggests that if the PDF of $\chi_x^2(k)$ and $\chi_y^2(k)$ (which should ideally be the same) can be identified correctly, this allows for determining if all the FTs are LOS or not. This can simply be achieved by a threshold test. For example, for seven FTs case as in [28], an appropriate threshold to characterize the Chi-square PDF for the LOS case is determined to be 2.71. If the area under the PDF on the right of this threshold is larger than 0.1, the random variable is identified to be non-centralized Chi-square distributed, and otherwise, it is centralized Chi-square distributed.

In case the PDF is determined to be non-centralized Chi-square, this implies that there is at least one NLOS FT. Then, the algorithm forms ${}_N C_{N-1} = N$ sets of FTs, with $(N-1)$ FTs in each set. For each set,

$$\sum_{i=1}^{N-1} {}_{N-1} C_i \ , \tag{120}$$

estimates of $\mathbf{x}_k$ are obtained. If any of these sets are found to be distributed according to centralized Chi-square distribution using the threshold test, then the number of LOS FTs is $(N-1)$, and the FTs within that particular set are used to estimate the MT location. Otherwise, the algorithm iterates until there are at most three LOS FTs. Since three FTs do not provide

[7]The summation in (118) starts from 3 since at least 3 FTs are required for location estimation in 2-D.

[8]Note that the definition of the *residual* in RT algorithm is different than the *residual* in Rwgh algorithm.

[9]The computation of CRLB requires the actual MT location $\mathbf{x}$. Since it is not available, $\mathbf{x}_{S_0}$ is used as an approximation to $\mathbf{x}$ to compute an approximate CRLB.

Authorized licensed use limited to: Xian Jiaotong University. Downloaded on October 19,2021 at 12:08:56 UTC from IEEE Xplore. Restrictions apply.

sufficient number of realizations for a reliable RT, delta test procedure is proposed, which takes two FTs first, and then combines them with one of the rest of the FTs to check if all three FTs are LOS. The simulation results show that the proposed technique outperforms the Rwgh [18] and CLS [46] algorithms, and can achieve the CRLB if the number of LOS FTs is larger than half of total number of FTs.

XI. Impact of FT Distribution on the Localization Accuracy

Before concluding the survey, one last important issue to be discussed relates to the impact of FT distribution[10] on the localization accuracy. Impact of the node locations on the accuracy has been analyzed in terms of achievable fundamental lower bounds in [21], [56]. In [56], the authors conclude that, for anchor-free localization[11], if the nodes are distributed within a rectangular region of $L_1 \times L_2$, the achievable accuracy improves as $L_1 \rightarrow L_2$ (i.e., when the region converges to a square). Nevertheless, for large number of nodes, the impact of the network shape on the achievable accuracy becomes insignificant.

In [21], an iterative algorithm called RELOCATE is proposed for optimally placing the reference nodes. For a fixed position of the target node, it optimally places the reference nodes so as to minimize the Cramer-Rao bound. Extension of the algorithm for multiple locations of the target node (such as a walking path within a building) is also presented.

Practical aspects of three dimensional placement of the FTs are evaluated in [57] using well known optimal solutions. An example scenario for placing four FTs within a cubic room is considered. Placing all the FTs on a planar surface (e.g., four different corners of the room's ceiling) yields a relatively low horizontal dilution of precision (HDOP) but a large vertical dilution of precision (VDOP)[12]. On the other hand, if the target nodes are placed in an "as good as possible" tetrahedron configuration, the HDOP is relatively smaller while the VDOP is significantly smaller compared to the planar configuration.

Optimum geometries of the FTs for different number of FTs are derived in [58]. In general, the FTs are placed on a geometry whose corners are "equally" distributed on a unit spherical surface. The five solutions to this problem for $N = 4, 6, 8, 12, 20$ correspond to a tetrahedron, octahedron, cube, icosahedron, and dodecahedron, respectively, which are also referred as Platonic solids. Also, any superposition of centered Platonic solids yields another optimum geometry [58].

In [59], [60], the authors analyze the relation between the localization probability and node distribution. First, the nodes are classified as L-nodes and NL-nodes. L-nodes are assumed to know their location, and NL-nodes are assumed unaware of their location (and need to localize themselves). The distributions of the L-nodes and NL-nodes in a two dimensional domain $\mathcal{S}_{dom} \subseteq \mathcal{R}^2$ are modeled through Poisson point processes ρ_L and ρ_{NL}, respectively. Then, [59] derives the probability that a randomly chosen NL-node over $\mathcal{S}_{dom}$ gets localized, as well as the probability of whole network of NL-nodes being localized for a log-normal shadow fading scenario.

The NL-node localization failure probability over a circular domain of radius R_{cir}, with per-node radio coverage radius $d_{rad} < R_{cir}$, total number of NL-nodes k_{NL}, and total number of nodes N is shown to be tightly bounded as follows in [60]

$$P_F \geq \left[1-(1-a)b^2\right]^{N-3}\left[1+b^2(1-a)(N-3) + b^4(1-a)^2\frac{(N-1)(N-2)}{2}\right], \quad (121)$$

where $a = 1 - k_{NL}/N$ is the fraction of the NL-nodes to total number of nodes and $b = d_{rad}/R_{cir}$. Extensions to log-normal shadowing and analysis of transition thresholds are also provided.

XII. Conclusion

In this paper, an extensive survey of different TOA based localization and NLOS mitigation techniques is presented. While some algorithms can perform close to the CRLB, they may require high computational complexities and availability of different prior information. For example in NLOS situations, prior information regarding the NLOS bias may not be available in many scenarios. In Table III, we provide a brief summary of different techniques, as well as their complexities and requirements. Practical and efficient localization techniques in the presence of NLOS bias still requires further research. The authors believe that this survey will serve as a valuable resource for evaluating the merits and trade-offs of the different available techniques towards developing more efficient and practical NLOS mitigation algorithms.

Acknowledgement

The authors would like to thank Dr. Fujio Watanabe from DOCOMO USA Labs and Dr. Sinan Gezici from Bilkent University for fruitful discussions, to Mr. Hiroshi Inamura from NTT DOCOMO Japan for his continuous support, and to anonymous reviewers and the editor Dr. Nelson Fonseca for their insightful comments.

References

[1] A. H. Sayed, A. Tarighat, and N. Khajehnouri, "Network-based wireless location," *IEEE Signal Processing Mag.*, vol. 22, no. 4, pp. 24–40, July 2005.

[2] F. Gustafsson and F. Gunnarsson, "Mobile positioning using wireless networks: Possibilites and fundamental limitations based on available wireless network measurements," *IEEE Signal Processing Mag.*, vol. 22, no. 4, pp. 41–53, July 2005.

[3] S. Gezici, "A survey on wireless position estimation," *Springer Wireless Personal Communications*, vol. 44, no. 3, pp. 263–282, Feb. 2008.

[4] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of wireless indoor positioning techniques and systems," *IEEE Trans. Syst., Man, Cybern. C: Applications and Reviews*, vol. 37, no. 6, pp. 1067–1080, Nov. 2007.

[5] N. Patwari, J. N. Ash, S. Kyperountas, A. O. H. III, R. L. Moses, and N. S. Correal, "Locating the nodes: cooperative localization in wireless sensor networks," *IEEE Signal Processing Mag.*, vol. 22, no. 4, pp. 54–69, Jul. 2005.

[10] Other than the FT distribution, other nuisance parameters may also have a considerable impact on the localization accuracy. For example, the reader is referred to [55] for a detailed discussion on the effects of FT height and path loss exponent on the achievable localization accuracies in a log-normal fading channel.

[11] No node knows its own location, but only the inter-node distance measurements are known.

[12] HDOP and VDOP are expressions for GDOP in horizontal and vertical domains, respectively.

Authorized licensed use limited to: Xian Jiaotong University. Downloaded on October 19,2021 at 12:08:56 UTC from IEEE Xplore. Restrictions apply.

TABLE III
OVERVIEW OF TOA BASED LOCALIZATION ALGORITHMS IN LOS AND NLOS SCENARIOS.

	Algorithm Name	References	Summary	Complexity, A-priori Knowledge etc.
ML Type Algorithms (LOS)	ML Algorithm	[19], [23]	The $\mathbf{x}$ that maximizes the joint probability of the distance measurements is taken as the location estimate.	Requires a comprehensive search over possible MT locations. Requires the knowledge of PDFs for distance measurements.
	Two-Step ML	[29]	The location estimate can be expressed in closed-form as in (24).	Requires iteration: first, $\mathbf{B}$ is evaluated based on the measured distances, an initial location estimate is obtained, and this is used to refine $\mathbf{B}$.
	Approximate ML	[26]	Uses the relation in (30) to obtain $\mathbf{x}$ in terms of s. The resulting quadratic relation is solved by employing a root selection routine.	Needs an initial location estimate (with a simpler estimator) to evaluate (31). May also need to iteratively update
LS Algorithms (LOS)	Non-Linear LS	[2], [30], [31]	Finds the $\mathbf{x}$ that minimizes the residual error as in (33).	Requires a search over possible MT locations. May employ techniques such as Gauss-Newton or Steepest Descent for faster convergence.
	Linear LS through Taylor's Series Expansion	[34]	Employs the Jacobian matrix in (39) in order to obtain the linear model in (38). Then solves it through a simple linear LS estimator.	Requires an accurate initial estimate $\mathbf{x}_0$ for linearization. May need to iterate for improved accuracy.
	LLS-1, LLS-2, LLS-3	[34] [37] [20]	Cancels out the non-linear x^2 and y^2 terms in (8) by simple subtraction operations to obtain the linear model. Then employs linear LS estimator.	LLS-1 does not appropriately selects the reference FT for linearization. Averaging techniques LLS-2 and LLS-3 have same accuracy, but may still use undesired FTs in linearization.
	LLS-RS	[38]	Selects the FT with smallest measured distance as a reference for linearization in LLS-1.	May not work well in NLOS scenarios.
	LLS-MLE	[38]	Utilizes the covariance matrix of observations in the linear model to obtain the MLE.	Requires the noise variance information, which is assumed identical at all the FTs.
	Constrained Weighted Least Squares	[32]	Uses the constraints in (54) to solve for the WLS formulation in (53).	Need to solve for the Langrange multiplier λ from the 5-root expression in (59).
ML Type Algorithms (NLOS)	ML Algorithm Utilizing NLOS Statistics	[19] [42]	In [19], $\mathbf{x}$ that maximizes the joint probability density function of the observations in NLOS scenarios is selected. MAP estimator utilizing the NLOS bias statistics is introduced in [42].	The probability density function of the NLOS bias and the distance measurements are assumed known, and requires a search over possible MT locations.
	IAD based ML Algorithm	[23] [42]	Uses the ML principle to discard the NLOS FTs. Then, only the LOS FTs are used in location estimation.	Need to collect N_{trn} TOA measurements at each FT to capture the noise statistics [23]. There may always be a possibility of mis-identification of the LOS FTs.
LS Algorithms (NLOS)	Weighted LS	[43], [44]	Uses some appropriate weights (e.g., using the variance of the distance measurements, or the statistics of the multipath components) to assign less reliability to NLOS FTs.	For a static MT, variance information may not be very different for LOS and NLOS FTs.
	Residual Weighting Algorithm	[18]	Different possible combinations of FT locations are considered. Then, each of the corresponding location estimates are weighted with the inverses of the residual errors to obtain the final location estimate.	Needs to solve for $N_{cb} = \sum_{i=3}^{N} {}_N C_i$ location estimates for different hypothesis before weighting them.
Constrained Localization Techniques (NLOS)	Constrained LS with QP	[46]	Two-step ML technique is used to obtain an estimate of the MT, with a quadratic constraint given as in (87).	May have high computational complexity.
	Constrained LS with LP	[20] [47]	The NLOS FTs are used to obtain a feasible region composed of squares. Then, the LOS FTs are used to solve for the MT location via LLS-1 technique so that the solution is within the feasible region.	Linear constraints yield a less complex (yet a coarser) solution compared to the quadratic constraints.
	Geometry Constrained Localization	[49]	A constraint related to the intersection points of the circles is incorporated into the two-step ML algorithm.	Slightly more complex than the two-step ML algorithm.
	Interior Point Optimization	[33]	First estimate NLOS bias values with IPO. Then use the NLOS bias estimates in a WLS solution (linearized using Taylor's series approximation).	Bias estimation through IPO may be computationally complex.
Robust Estimators (NLOS)	M-estimators	[51]	Employs a convex function $\rho(\nu)$ to capture the effects of NLOS bias values in an "ML-type" of estimator.	Need to tune $\rho(\nu)$ appropriately. Better alternatives such as the S-estimators (more robust) and MM-estimators (both robust and efficient) are available.
	Least Median of Squares	[37]	The location that minimizes the LMS of the residual is selected as a location estimate.	Robust up to %50 of outliers. More computationally complex than the NLS.
Identify and Discard Techniques (NLOS)	Residual Test Algorithm	[28]	Identifies and discards the NLOS FTs. The residual errors are normalized by the CRLBs, and resulting variables are checked to find if they are centralized or non-centralized Chi-square distributed.	Computationally complex due to testing numerous hypothesis, Delta-test etc.

Authorized licensed use limited to: Xian Jiaotong University. Downloaded on October 19,2021 at 12:08:56 UTC from IEEE Xplore. Restrictions apply.

[6] L. Cong and W. Zhuang, "Non-line-of-sight error mitigation in mobile location," in *Proc. IEEE Conf. Computer Commun. (INFOCOM)*, Hong Kong, Mar. 2004, pp. 650–659.

[7] B. Li, A. Dempster, C.Rizos, and H. K. Lee, "A database method to mitigate NLOS error in mobilephone positioning," in *Proc. IEEE Position, Location, and Navigation Symp. (PLANS)*, San Diego, CA, Apr. 2006, pp. 173–178.

[8] H. Miao, K. Yu, and M. J. Juntti, "Positioning for NLOS propagation: Algorithm derivations and CramerRao bounds," *IEEE Trans. Veh. Technol.*, vol. 56, no. 5, pp. 2568–2580, Sep. 2007.

[9] J.-Y. Lee and R. A. Scholtz, "Ranging in a dense multipath environment using an UWB radio link," *IEEE J. Select. Areas Commun.*, vol. 20, no. 9, pp. 1677–1683, Dec. 2002.

[10] K. Yu and I. Oppermann, "Performance of UWB position estimation based on time-of-arrival measurements," in *Proc. IEEE Conf. Ultrawideband Syst. Technol. (UWBST)*, Kyoto, Japan, May 2004, pp. 400–404.

[11] P. Cheong, A. Rabbachin, J. Montillet, K. Yu, and I. Oppermann, "Synchronization, TOA and position estimation for low-complexity LDR UWB devices," in *Proc. IEEE Int. Conf. UWB (ICU)*, Zurich, Switzerland, Sep 2005, pp. 480–484.

[12] A. Rabbachin, I. Oppermann, and B. Denis, "ML time-of-arrival estimation based on low complexity UWB energy detection," in *Proc. IEEE Int. Conf. Ultrawideband (ICUWB)*, Waltham, MA, Sept. 2006.

[13] D. Dardari, C.-C. Chong, and M. Z. Win, "Threshold-based time-of-arrival estimators in UWB dense multipath channels," *IEEE Trans. Commun.*, vol. 56, no. 8, Aug. 2008.

[14] I. Guvenc, Z. Sahinoglu, and P. Orlik, "TOA estimation for IR-UWB systems with different transceiver types," *IEEE Trans. Microwave Theory Tech. (Special Issue on Ultrawideband)*, vol. 54, no. 4, pp. 1876–1886, Apr. 2006.

[15] I. Guvenc and Z. Sahinoglu, "Threshold selection for UWB TOA estimation based on kurtosis analysis," *IEEE Commun. Lett.*, vol. 9, no. 12, pp. 1025–1027, Dec. 2005.

[16] Z. Sahinoglu and I. Guvenc, "Multiuser interference mitigation in noncoherent UWB ranging via nonlinear filtering," *EURASIP J. Wireless Commun. and Networking*, vol. 2, pp. 1–10, Apr. 2006.

[17] S. Gezici, Z. Tian, G. B. Giannakis, H. Kobayashi, A. F. Molisch, H. V. Poor, and Z. Sahinoglu, "Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks," *IEEE Signal Processing Mag.*, vol. 22, no. 4, pp. 70–84, July 2005.

[18] P. C. Chen, "A non-line-of-sight error mitigation algorithm in location estimation," in *Proc. IEEE Int. Conf. Wireless Commun. Networking (WCNC)*, vol. 1, New Orleans, LA, Sept. 1999, pp. 316–320.

[19] S. Gezici and Z. Sahinoglu, "UWB geolocation techniques for IEEE 802.15.4a personal area networks," MERL Technical report, Cambridge, MA, Aug. 2004.

[20] S. Venkatesh and R. M. Buehrer, "A linear programming approach to NLOS error mitigation in sensor networks," in *Proc. IEEE Int. Symp. Information Processing in Sensor Networks (IPSN)*, Nashville, Tennessee, Apr. 2006, pp. 301–308.

[21] D. B. Jourdan and N. Roy, "Optimal sensor placement for agent localization," in *Proc. IEEE Position, Location, and Navigation Symposium (PLANS)*, San Diego, CA, Apr. 2006, pp. 128–139.

[22] V. Dizdarevic and K. Witrisal, "On impact of topology and cost function on LSE position determination in wireless networks," in *Proc. Workshop on Positioning, Navigation, and Commun. (WPNC)*, Hannover, Germany, Mar. 2006, pp. 129–138.

[23] J. Riba and A. Urruela, "A non-line-of-sight mitigation technique based on ML-detection," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 2, Quebec, Canada, May 2004, pp. 153–156.

[24] B. Denis, J. B. Pierrot, and C. Abou-Rjeily, "Joint distributed synchronization and positioning in UWB ad hoc networks using TOA," *IEEE Trans. Microwave Theory Tech.*, vol. 54, no. 4, pp. 1896–1911, Apr. 2006.

[25] D. B. Jourdan, D. Dardari, and M. Z. Win, "Position error bound for UWB localization in dense cluttered environments," in *Proc. IEEE Int. Conf. Commun. (ICC)*, vol. 8, Istanbul, Turkey, June 2006, pp. 3705–3710.

[26] Y. T. Chan, H. Y. C. Hang, and P. C. Ching, "Exact and approximate maximum likelihood localization algorithms," *IEEE Trans. Veh. Technol.*, vol. 55, no. 1, pp. 10–16, Jan. 2006.

[27] C. Cheng and A. Sahai, "Estimation bounds for localization," in *Proc. IEEE Int. Conf. Sensor and Ad-Hoc Communications and Networks (SECON)*, Santa Clara, CA, Oct. 2004, pp. 415–424.

[28] Y. T. Chan, W. Y. Tsui, H. C. So, and P. C. Ching, "Time of arrival based localization under NLOS conditions," *IEEE Trans. Veh. Technol.*, vol. 55, no. 1, pp. 17–24, Jan. 2006.

[29] Y. T. Chan and K. C. Ho, "A simple and efficient estimator for hyperbolic location," *IEEE Trans. Signal Processing*, vol. 42, no. 8, pp. 1905–1915, Aug. 1994.

[30] J. J. Caffery and G. L. Stuber, "Overview of radiolocation in CDMA cellular systems," *IEEE Commun. Mag.*, vol. 36, no. 4, pp. 38–45, Apr. 1998.

[31] ——, "Subscriber location in CDMA cellular networks," *IEEE Trans. Veh. Technol.*, vol. 47, no. 2, pp. 406–416, May 1998.

[32] K. W. Cheung, H. C. So, W. K. Ma, and Y. T. Chan, "Least square algorithms for time-of-arrival-based mobile location," *IEEE Trans. Signal Processing*, vol. 52, no. 4, pp. 1121–1128, Apr. 2004.

[33] W. Kim, J. G. Lee, and G. I. Jee, "The interior-point method for an optimal treatment of bias in trilateration location," *IEEE Trans. Veh. Technol.*, vol. 55, no. 4, pp. 1291–1301, July 2006.

[34] J. J. Caffery, "A new approach to the geometry of TOA location," in *Proc. IEEE Veh. Technol. Conf. (VTC)*, vol. 4, Boston, MA, Sep. 2000, pp. 1943–1949.

[35] I. Guvenc, C. C. Chong, and F. Watanabe, "Analysis of a linear least-squares localization technique in LOS and NLOS environments," in *Proc. IEEE Veh. Technol. Conf. (VTC)*, Dublin, Ireland, Apr. 2007, pp. 1886–1890.

[36] S. Gezici, I. Guvenc, and Z. Sahinoglu, "On the performance of linear least-squares estimation in wireless positioning systems," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Beijing, China, May 2008.

[37] Z. Li, W. Trappe, Y. Zhang, and B. Nath, "Robust statistical methods for securing wireless localization in sensor networks," in *Proc. IEEE Int. Symp. Information Processing in Sensor Networks (IPSN)*, Los Angeles, CA, Apr. 2005, pp. 91–98.

[38] I. Guvenc, S. Gezici, F. Watanabe, and H. Inamura, "Enhancements to linear least squares localization through reference selection and ML estimation," in *Proc. IEEE Wireless Commun. Networking Conf. (WCNC)*, Las Vegas, NV, Apr. 2008, pp. 284–289.

[39] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*. Upper Saddle River, NJ: Prentice Hall, Inc., 1993.

[40] K. W. Cheung, H. C. So, W.-K. Ma, and Y. T. Chan, "A constrained least squares approach to mobile positioning: algorithms and optimality," *EURASIP J. Applied Sig. Processing*, vol. 2006, pp. 1–23, 2006.

[41] A.-C. Chang and C.-M. Chung, "Covariance shaping least-squares location estimation using TOA measurements," *IEICE Trans. Fundamentals of Electron., Commun., Computer Sciences*, vol. E90-A, no. 3, pp. 691–693, Mar. 2007.

[42] Y. Qi, H. Kobayashi, and H. Suda, "Analysis of wireless geolocation in a non-line-of-sight environment," *IEEE Trans. Wireless Commun.*, vol. 5, no. 3, pp. 672–681, Mar. 2006.

[43] I. Guvenc, C. C. Chong, and F. Watanabe, "NLOS identification and mitigation for UWB localization systems," in *Proc. IEEE Int. Conf. Wireless Commun. Networking (WCNC)*, Hong Kong, Mar. 2007, pp. 1571–1576.

[44] I. Guvenc, C. C. Chong, F. Watanabe, and H. Inamura, "NLOS identification and weighted least squares localization for UWB systems using multipath channel statistics," *EURASIP J. Advances in Signal Processing (Special Issue on Signal Processing for Location Estimation and Tracking in Wireless Environments)*, vol. 2008, no. 1, pp. 1–14, Jan. 2008.

[45] X. Li, "An iterative NLOS mitigation algorithm for location estimation in sensor networks," in *Proc. IST Mobile and Wireless Commun. Summit*, Myconos, Greece, June 2006.

[46] X. Wang, Z. Wang, and B. O. Dea, "A TOA based location algorithm reducing the errors due to non-line-of-sight (NLOS) propagation," *IEEE Trans. Veh. Technol.*, vol. 52, no. 1, pp. 112–116, Jan. 2003.

[47] S. Venkatesh and R. M. Buehrer, "NLOS mitigation using linear programming in ultrawideband location-aware networks," *IEEE Trans. Veh. Technol.*, vol. 56, no. 5, pp. 3182–3198, Sep. 2007.

[48] E. G. Larsson, "Cramer-Rao bound analysis of distributed positioning in sensor networks," *IEEE Signal Processing Lett.*, vol. 11, no. 3, pp. 334–337, Mar. 2004.

[49] C. L. Chen and K. T. Feng, "An efficient geometry-constrained location estimation algorithm for NLOS environments," in *Proc. IEEE Int. Conf. Wireless Networks, Commun., Mobile Computing*, Hawaii, USA, June 2005, pp. 244–249.

[50] P. Petrus, "Robust Huber adaptive filter," *IEEE Trans. Signal Processing*, vol. 47, no. 4, pp. 1129–1133, Apr. 1999.

[51] G. L. Sun and W. Guo, "Bootstrapping M-estimators for reducing errors due to non-line-of-sight (nlos) propagation," *IEEE Commun. Lett.*, vol. 8, no. 8, pp. 509–510, Aug. 2004.

[52] R. Casas, A. Marco, J. J. Guerrero, and J. Falco, "Robust estimator for non-line-of-sight error mitigation in indoor localization," *Eurasip J. Applied Sig. Processing*, vol. 2006, no. 1, pp. 1–8, Jan. 2006.

Authorized licensed use limited to: Xian Jiaotong University. Downloaded on October 19,2021 at 12:08:56 UTC from IEEE Xplore. Restrictions apply.

[53] B. D. Ripley, "Robust statistics," 2004. [Online]. Available: http://www.stats.ox.ac.uk/pub/StatMeth/

[54] J. Taylor, "Robust modelling using the t-distribution," in *Proc. Australasian Genstat Conf.*, Busselton, Australia, Dec. 2002.

[55] R. A. Malaney, "Nuisance parameters and location accuracy in log-normal fading models," *IEEE Trans. Wireless Commun.*, vol. 6, no. 3, pp. 937–947, Mar. 2007.

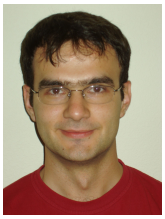
[56] C. Chang and A. Sahai, "Cramer-rao-type bounds for localization," *EURASIP J. Applied Sig. Processing*, vol. 2006, pp. 1–13, 2006.

[57] J. Schroeder, S. Galler, K. Kyamakya, and K. Jobmann, "Practical considerations for optimal three dimensional localization," in *Proc. IEEE Int. Conf. Multisensor Fusion and Integration for Intelligent Syst. (MFI)*, Heidelberg, Germany, Sep. 2006, pp. 439–443.

[58] B. Yang and J. Scheuing, "Cramer-Rao bound and optimum sensor array for source localization from time differences of arrival," in *Proc. IEEE Int. Conf. Acoust., Speech, Sig. Processing (ICASSP)*, vol. 4, Philadelphia, PN, Mar. 2005, pp. 961–964.

[59] F. Daneshgaran, M. Laddomada, and M. Mondin, "The problem of localization in networks of randomly deployed nodes: Asymptotic and finite analysis, and thresholds," *submitted to IEEE Trans. Inform. Theory*, Oct. 2007, available on ArXiv.org arXiv:0710.0531.

[60] ——, "Connection between system parameters and localization probability in network of randomly distributed nodes," *IEEE Trans. Wireless Commun.*, vol. 6, no. 12, pp. 4383–4389, Dec. 2007.



Ismail Guvenc received the B.S. degree in electrical and electronics engineering from Bilkent University, Turkey, in 2001, the M.S. degree in electrical and computer engineering from University of New Mexico in 2002, and the Ph.D. degree in electrical engineering from University of South Florida in 2006. He was with Mitsubishi Electric Research Labs between January and August, 2005, where he worked on UWB ranging and positioning. Since June 2006, he has been with the DOCOMO USA Communications Laboratories, Palo Alto, CA. His research interests are related to UWB communications, UWB ranging/localization, and next generation wireless systems. He has published more than 30 international conference and journal papers, several standardization contributions, and a book chapter. He has several pending US patent applications and has co-authored a book on UWB ranging and localization. He is a member of the IEEE.

Chia-Chin Chong received the B.Eng (Hons) and Ph.D. degrees from The University of Manchester, Manchester, UK and The University of Edinburgh, Edinburgh, UK, in 2000 and 2003, respectively, and currently a senior researcher at the DOCOMO USA Labs. Her research interests include channel measurements and modeling, UWB systems, ranging and positioning techniques, 4G cellular systems, and relaying and cooperative communications. She has published more than 80 international journals, conference papers, and standard contributions. Dr. Chong has received numerous awards including the IEEE International Conference on Ultra-Wideband Best Paper Award, DOCOMO USA Labs President Award and The Outstanding Young Malaysian Award, all in 2006, and URSI Young Scientist Award in 2008. She currently serves as an Editor for the IEEE Transactions on Wireless Communications and has served on the Technical Program Committee (TPC) of various international conferences including the TPC Co-Chair for the Wireless Communications Symposium of the IEEE International Conference on Communications (ICC) 2008. She is also the Chair for the DG-EVAL Channel Model standardization group within the ITU-R WP5D for IMT systems. She is a senior member of the IEEE.

Authorized licensed use limited to: Xian Jiaotong University. Downloaded on October 19,2021 at 12:08:56 UTC from IEEE Xplore. Restrictions apply.

附录 B　论文翻译

基于时间到达量测和非线性缓解的定位技术

Ismail G ¨uvenc¸, Chia-Chin Chong,
Member, IEEE, Senior Member, IEEE

摘要　关于利用不同基站使用时间到达量测的无线设备定位已经被深入研究。同时报道了很多带有不同准确度，计算复杂度，先验知识要求以及不同层次的对抗非线性误差健壮性的算法。然而就我们所知，目前仍缺少一个详细统一的关于不同定位算法的综述。本篇文章希望给出这些定位算法的综合的回顾以及他们的技术挑战，并且指出可能的研究方向。首先，我们总结了非线性场景下的下限以及一些接近这些下限的实用算法。然后在给出下限之后，我们分类并总结了不同的缓解非线性误差的技巧我们同样提供了仿真结果来比较不同算法的表现。最后，我们给出了一个表格来归纳这些算法的关键特点。

关键字　克拉美罗下限　位置预测　非线性场景　时间到达量测　定位

一．引言

近来，在诸如蜂窝移动网络，无线局域网，无线传感器网络的无线系统中，定位意识获得了极大的关注，因为他能提供大范围的附加应用。为了提升未来的生活质量，基于定位的服务变得越来越重要，如基于定位的社交，广告以及 911 服务等。例如，基于定位的服务允许使用者有策略的向那些住在附近且已被影响的顾客传达信息。例如，只有当用户进入商城时才会收到优惠券和电子销售项目。另一方面，基于位置的社交网络可能会进一步增强基于互联网的社交网络服通过允许用户按照偏好和兴趣组成社区，如 facebook,friendsters,myspace,等。对于 911 紧急服务，使用者能够拨打紧急服务使得当地政府精准跟踪并定位使用者的地位，无论在室内还是室外。上文提到的由定位支持的服务使得普遍的定位服务网络成为可能，而这需要无线设备的精准定位。

尽管过去几十年已经深入研究了定位问题，现在仍有许多问题待解决。一个关键的挑战是在密集分布的非线性场景中定位的精准度和效率。非线性场景是指发射器和接收器之间被阻断，而这无论是在室内还是室外都很常见。在这种场景下，全球定位系统即使不是不可能也变得不实用。

之前的一些文章提供了基于角度到达，时间到达，时间差到达以及到达信号强度量测技巧的定位问题的广泛观点。然而，这些工作并没有为了提高由路径阻断造成的衰减表现而研究非线性缓解的技巧。在这篇文章中，我们提供了适用于线性以及非线性场景的基于时间到达量测的综合性研究。对于其他的技巧，感兴趣的读者可以阅读参考文献。

这篇文章的目的有两个。首先，提供一个统一的关于不同时间到达量测技巧的概述以及相关的非线性缓解的方法。其次，研究这些技巧的优劣及内在关联。请注意，需要特定的技术的基于第一个到达路径的 TOA 不在本文范围之内。感兴趣的读者可以阅读参考文献。

本文组织如下。第二节简要回顾了不同的位置估计技术，即 TOA，TDOA、AOA、RSS 和基于模式匹配的方法。在第 III 部分中，概述了基于 TOA 的本地化方案并提出了系统模型以及问题定义。对于其余各节，第四节和第五节专用于 LOS 场景，而第六节专用于 NLOS 方案。第四节提供了 LOS 系统的基本下限并总结一些基于最大似 x 然 ML 的关键技术从而接近这些基本界限。第五节提供了基于 TOA 系统的不同最小二乘法 （LS）估计技术的统一讨论。第六

节提供了基本下限及一些用于 NLOS 的 ML 技术，第七节是关于在 NLOS 场景中的 LS 技术，第 九部分是关于鲁棒估计器的 NLOS 场景，第十部分是对 NLOS 缓解技术识别和丢弃类型的回顾。第十一部分研究 FT 分布对本地化的影响准确性。最后，第十二节总结了可用的技术并提供一些结论性意见。

二．不同定位技术概述

总的来说，高精度的定位分为两步。1.首先要精准地确定范围（测量两点之间距离的一个动作）2.然后定位（通过三个或以上已知点的交叉确定未知点的位置）。在本部分，讨论了五种常见定位方式的概述。在表格一中展现。

1. 对于 TOA 方法，距离信息由发送点和接收点之间的传播延迟获得。这个方法可以更具体地分为单路 TOA 和双路 TOA。单路 TOA 要求发送点和接收点之间地完美同步，而双路 TOA 并不要求同步。这个方法经常用于蜂窝移动网络，因为在该网络中接收点和基站同步。
2. 对于 TDOA 方法，和 TOA 的不同在于该方法用了多个接收点来重建发送点地位置。该方法既可以基于来自单个点地单信号到达多个点的时间差，也可以基于来自单个点地多个信号到达单个点地时间差。该方法要求接收点之间地完美同步但并不要求发送点和接收点之间的同步。TDOA 通常应用与无线设备传感器网络中。
3. 对于 AOA，节点之间的距离通常是由他们之间的角度确定。该方法主要的缺陷在于它要求一个新的硬件设备-天线阵列及各元素之间最小距离的应用通常会导致额外的花费和更大的节点。另外，这个方法对多路，非线性，矩阵精准度高度敏感。
4. 对于 RSS，距离由发送点和接收点之间传播信号的衰减确定。因为距离与衰减之间的关系依赖于通道，可靠的距离预测需要一个准确的传播模型。该方法的优点在于它的花费低而且大部分接收点足以预测。然而，移动点的移动性以及通道不可预测的误差通常会导致距离分析中大的误差。该方法对噪声和干扰高度敏感。因此，RSS 并不是准确的方法，它的应用通常限制在那些粗糙定位的情况。
5. 最后对于模式匹配，它利用了不同几何位置点的测量信号的指纹信息来进行位置预测。这种信息通常是位置敏感并且可以在训练阶段收集的。在实际情况下，指纹信息可以用来定位移动节点。指纹信息库通常由来自不同参考点以及不同移动节点的接收信号的强度组成。然而它也具有更细节的指纹信息，如平均过度延时，均方根延时传播，最大过度延时，总接收强度，多路设备数量等。该系统的一个挑战是由通道和环境状况变化引起的，会导致数据库信息不可靠并且需要频繁更新数据库。

三．基于 TOA 定位的系统模型

考虑一个有 N 个固定终端的无线网络，**x** 是移动终端的预测位置，**x_i** 是第 i 个固定终端的位置，**d_i** 移动终端和第 i 个终端之间的距离，通常为如下

$$\hat{d}_i = d_i + b_i + n_i = ct_i, i = 1, 2, ..., N \tag{1}$$

其中，ti 是第 i 个固定终端信号的 TOA，c 是光速，di 是移动终端和第 i 个固定终端之间的实际距离，ni 是方差为 σ 的高斯白噪声，bi 是正距离误差，由路径阻挡造成。

$$b_i = \begin{cases} 0, & \text{if } i \text{ th FT is LOS} \\ \psi_i, & \text{if } i \text{ th FT is NLOS} \end{cases} \tag{2}$$

对于 NLOS 的固定终端，偏差通常被建模为指数分布，均匀分布，高斯分布,时间窗口的常数或基于实验测量的模型。通常，模型取决于无线传播路径和特殊的技术（蜂窝移动网络，无线传感器设备网等）。
让

$$\mathbf{d}=\mathbf{d}(\mathbf{x})=\left[d_1,d_2,\ldots,d_N\right]^T \tag{3}$$

为移动终端和固定终端之间的实际距离

$$\mathrm{d}=\left[\hat{d}_1,\hat{d}_2,\ldots,\hat{d}_N\right]^T \tag{4}$$

为测量距离

$$\mathbf{b}=\left[b_1,b_2,\ldots,b_N\right]^T \tag{5}$$

为偏差向量

$$\mathbf{Q}=\mathrm{E}\left[\mathbf{n}\mathbf{n}^T\right]=\mathrm{diag}\left[\sigma_1^2,\sigma_2^2,\ldots,\sigma_N^2\right]^T \tag{6}$$

为噪声向量的协方差，噪声为独立零均值高斯随机变量。
在没有噪声和 NLOS 误差干扰下，移动终端和第 i 个固定终端的实际距离 di 定义了一个以 FTi 为圆心的圆

$$\left(x-x_i\right)^2+\left(y-y_i\right)^2=d_i^2, i=1,2,\ldots,N \tag{7}$$

所有的圆交在一点上，解表达式即可求出 MT 的准确位置。然而，由于测量噪声和不同 FT 的 NLOS 误差的影响，这些也并不交在同一点上，导致了方程式的不一致。

$$\left(x-x_i\right)^2+\left(y-y_i\right)^2=\hat{d}_i^2, i=1,2,\ldots,N \tag{8}$$

为了有更紧凑的表达式，我们定义了

$$s=x^2+y^2, k_i=x_i^2+y_i^2 \tag{9}$$

基于 TOA 的定位预测问题可以被定义为基于 FT 位置 xi 的带测量噪声的 MT 位置 x 的预测。换种说法，给出一系列等式，提出不同的定位方法来预测 MT 的位置。在接下来的部分，我们会概述这些方法并讨论他们的优劣。

四．LOS 场景：基本限制和最大似然解决方法

在本部分，我们会概述算法在 LOS 场景的下界和最大似然类型。首先，我们定义了最大化测量距离 d 的条件概率，之后会引出克拉美罗下限并且讨论两个近似克拉美罗下限的次最优解最大似然类型的算法。

A. 最大概率算法

在没有 NLOS 偏差的情况下，测量距离 d^的条件概率密度可以用 x 表示为

$$P(d|\ x)=\prod_{i=1}^{N}\frac{1}{\sqrt{2\pi\sigma_i^2}}\exp\left\{-\frac{\left(\hat{d}_i-d_i\right)^2}{2\sigma_i^2}\right\} \tag{10}$$

$$=\frac{1}{\sqrt{(2\pi)^N\det(\mathbf{Q})}}\exp\left\{-\frac{\mathrm{J}}{2}\right\} \tag{11}$$

其中，

$$\mathrm{J}=[\hat{d}-d(x)]^T Q^{-1}[\hat{d}-d(x)] \tag{12}$$

最大似然解决方案的 x 是最大化 P(d|x)的 x

$$\hat{x}=\arg\max_x P(\hat{d}|\ x) \tag{13}$$

注意到 x 需要在所有可能的移动终端中寻找，而这需要大量的计算。
对于 σ i^2= σ 的特殊情况，ML 的解等价于最小化 J。为了找到 J 的最小值，J 关于 x 的微分应该等于 0，产生如下公式，

$$\sum_{i=1}^{N}\frac{\left(d_i-\hat{d}_i\right)\left(x-x_i\right)}{d_i}=0 \tag{14}$$

$$\sum_{i=1}^{N}\cdot\frac{\left(d_i-\hat{d}_i\right)\left(y-y_i\right)}{d_i}=0 \tag{15}$$

而这是非线性等式，因此 x 不能用线性最小化二乘法以封闭式求出。同样，（14）和（15）依赖于未知的 di。尽管闭合式的解不能求出，近似和迭代的方法可以推导出来，他们可以求得 CRLB 的渐进解，这会在第八部分讨论。

B. 克拉美罗下限

基于 d^的条件概率密度，我们可以推导出基于 TOA 的位置预测的 CRLB。总的来说，CRLB 是一个无差参数的误差估计的方差的理论下限。基于 TOA 的 CRLB 通常由下述条件决定。

1. FT 的位置 2.MT 的实际位置和 3.测量噪声的方差

CRLB 由 fisher 信息矩阵求出，它的元素定义为

$$[\mathbf{I}(\mathbf{x})]_{ij}=-\mathrm{E}\left[\frac{\partial^2\ln P(\hat{d}|\ \mathbf{x})}{\partial\mathbf{x}_i\partial\mathbf{x}_j}\right] \tag{16}$$

然后，使用 PDF，FIM 可以被计算出来如下

$$\mathbf{I}(\mathbf{x})=\begin{bmatrix}\sum_{i=1}^{N}\cdot\frac{\left(x-x_i\right)^2}{\sigma_i^2d_i^2} & \sum_{i=1}^{N}\cdot\frac{\left(x-x_i\right)\left(y-y_i\right)}{\sigma_i^2d_i^2}\\ \sum_{i=1}^{N}\cdot\frac{\left(x-x_i\right)\left(y-y_i\right)}{\sigma_i^2d_i^2} & \sum_{i=1}^{N}\cdot\frac{\left(y-y_i\right)^2}{\sigma_i^2d_i^2}\end{bmatrix} \tag{17}$$

$$=\begin{bmatrix} \sum_{i=1}^{N}\cdot\frac{\cos^2(\alpha_i)}{\sigma_i^2} & \sum_{i=1}^{N}\cdot\frac{\cos(\alpha_i)\sin(\alpha_i)}{\sigma_i^2} \\ \sum_{i=1}^{N}\cdot\frac{\cos(\alpha_i)\sin(\alpha_i)}{\sigma_i^2} & \sum_{i=1}^{N}\cdot\frac{\sin^2(\alpha_i)}{\sigma_i^2} \end{bmatrix} \tag{18}$$

其中，α 为 MT 和 FT 之间的夹角，CRLB 由 I-1(x)给出。因此，对于任意无偏估计器得到的 MT 位置 x 的预测，我们有

$$\mathrm{E}_{\mathbf{x}}\left[(\mathbf{x}-\mathbf{x})(\mathbf{x}-\mathbf{x})^T\right]\geq\mathbf{I}^{-1}(\mathbf{x}) \tag{19}$$

CRLB 同样和另一个被定义为几何放大因子（GDOP）的测量矩阵有关。对于方差一致的噪声，GDOP 可以定义为

$$\mathrm{GDOP}=\frac{\mathrm{RMSE}_{\mathrm{loc}}}{\mathrm{RMSE}_{\mathrm{range}}}=\frac{\sigma_{\mathrm{loc}}}{\sigma} \tag{20}$$

其中 RMSE 和 RMSE 是位置估计和范围估计的均方根误差，σ loc 是位置估计的标准偏差。GDOP 高度依赖于 FT 和 MT 的位置。GDOP 小于三时是比较好的，大于六时往往意味着一个糟糕的 FT 的集合布局。如果采用的位置估计器可以到达 CRLB，GDOP 由下式给出

$$\mathrm{GDOP}=\frac{\sqrt{\mathrm{trace}\left[\mathbf{I}^{-1}(\mathbf{x})\right]}}{\sigma} \tag{21}$$

$$=\sqrt{\mathrm{trace}\left[\mathbf{I}^{-1}(\mathbf{x})\right]} \tag{22}$$

其中

$$\tilde{\mathbf{I}}(\mathbf{x})=\begin{bmatrix} \sum_{i=1}^{N}\cdot\cos^2(\alpha_i) & \sum_{i=1}^{N}\cdot\cos(\alpha_i)\sin(\alpha_i) \\ \sum_{i=1}^{N}\cdot\cos(\alpha_i)\sin(\alpha_i) & \sum_{i=1}^{N}\cdot\sin^2(\alpha_i) \end{bmatrix} \tag{23}$$

可实现的位置准确度和 MT 与 FT 之间的几何布局之间的关系由（22）式可以明显看出。

（1）仿真结果：不同的 MT 位置在一个简单的无线位置场景中的 CRLB 由图可知。四个 FT 放在[0,0]m,[0,20]m,[20,0]m,[20,20]m 并且所有 FT 的 σ 都一样。当 MT 靠近房间中心时，CRLB 变小。同样，在四个特殊点，FIM 矩阵变为奇异矩阵，无法求逆，解释了图 2 中的白点。

为了看到在无线场景中 GDOP 的典型值，图三模拟了一些典型的拓扑节点。MT 的三个位置[5,5]m,[25,-25]m,[-50,50]m 被纳入考虑。FT1 被固定在[20√2,0]m，新的 FT 被逆时针放在图示圆中。我们考虑三个拓扑节点 T1,T2,T3。对每个拓扑，一个新的 FT 被按 $2\pi/\mathrm{N}_{\max}$，

$\pi/\mathrm{N}_{\max}$，

$\pi/2\mathrm{N}_{\max}$ 增量放置。$\mathrm{N}_{\max}=10$ 表示最大 FT 个数。当 MT 在[5,5]位置时，GDOP 对所有拓扑都小于 2 而且当更多的 FT 被放置时，GDOP 变得更好。当 MT 放置在[50,-50]时，GDOP 是最差的。对于 T3 来说可能大于 10。结果显示，当我们提升 FT 的数量时，GDOP 可能会小于 1，这意味着当测量标准差变小时，位置预测的标准差也在变小。

C. **两步最大似然算法**

尽管 CRLB 给出了可到达准确度的下限，实际情况中往往很难到达。在特定情况下，早期的基于 TOA 的位置预测能到达 CRLB 的方法在[29]中介绍。这是两步最大似然算法,ML 的解可以通过下式获得。

$$\boldsymbol{\theta}=\frac{1}{2}\left(\mathbf{A}_1^T\boldsymbol{\Psi}^{-1}\mathbf{A}_1\right)^{-1}\mathbf{A}_1^T\boldsymbol{\Psi}^{-1}\mathbf{p}_1 \tag{24}$$

其中，A1 和 p1 被定义如下

$$\boldsymbol{\theta}=[x,y,s]^T \tag{25}$$

$$\Psi=E\left[\psi\psi^T\right]=BQB \tag{26}$$

$$\psi=\mathbf{p}_1-\mathbf{A}_1\theta \tag{27}$$

$$\mathbf{B}=\mathrm{diag}\left\{2d_1,2d_2,\ldots,2d_N\right\} \tag{28}$$

因为 B 的元素未知，一个近似的解决方案是用测量距离 d 来代替实际距离作为初始解。之后通过利用初始解重新计算 B 获得一个更加准确的解，几次迭代就足够收敛了。

D. **近似最大似然算法**

另一个可到达 CRLB 的基于最大似然的算法在[26]中介绍。4-A 部分显示最大似然算法需要实际距离 d 的情况来求得 MT 位置的解。假设所有的$\sigma_i=\sigma$，[26]提出了在大多数情况下可到达 CRLB 的 AML 算法，他们使用等式

$$d_i-\hat{d}_i=\frac{d_i^2-\hat{d}_i^2}{d_i+\hat{d}_i} \tag{29}$$

来解（14）和（15）并获得了如下等式

$$2\begin{bmatrix}\sum g_i x_i & \sum g_i y_i\\ \sum h_i x_i & \sum h_i y_i\end{bmatrix}\begin{bmatrix}x\\ y\end{bmatrix}=\begin{bmatrix}\sum g_i\left(s+k_i-\hat{d}_i^2\right)\\ \sum h_i\left(s+k_i-\hat{d}_i^2\right)\end{bmatrix} \tag{30}$$

其中 s 和 k 如（9）中定义并且全部是从 1 到 N，且

$$g_i=\frac{x-x_i}{d_i\left(d_i+d_i\right)},h_i=\frac{y-y_i}{d_i\left(d_i+d_i\right)} \tag{31}$$

注意到，除了 s，权重 g 和 h 同样依赖于 **x**,而这明显是未知的。因此 AML 技巧首先要获得一个 **x 的**粗略的初始预测来计算 g 和 h（如使用（37）的预测器）。之后利用 LS 算法解（30）得到用 s 表示的 x，产生了一个二次方程。之后用根轨迹选择获得一个 x 的合适解。使用新的解重新计算 g 和 h，几次迭代后即可实现 CRLB。

五．**LOS 场景：最小二乘法**

在这一部分，我们首先回顾用来预测 MT 位置的非线性最小二乘法，之后简要讨论一些线性化技巧。

A. **非线性最小二乘法**

NLS 是一种著名的用来估计概率分布未知的未知参数的技术，它同样是一种用来估计 ML 位置的技术，由[30]给出，

$$\mathbf{x} = \arg\min_{\mathbf{x}} \left\{ R_{es}(\mathbf{x}) \right\} \tag{32}$$

$$= \arg\min_{\mathbf{x}} \left\{ \sum_{i=1}^{N} \beta_i \left(\hat{d}_i - \left\| \mathbf{x} - \mathbf{x}_i \right\| \right)^2 \right\} \tag{33}$$

其中 Res 是 MT 位置 x 的残差权重β用来代表每一个关系的可信度，这产生了带权最小二乘估计法。如果没有可信度的信息，则所有β=1。最小化非线性表达式需要数字的搜索方式，如最速下降法，高斯牛顿迭代法，而这计算成本高而且需要很好的初始值来避免收敛到随时函数的局部最优点。

B. **非线性模型的矩阵表示**

我们可以把（8）中的表达式用矩阵形式替代，在经过一些操作后，我们得到如下[32]，

$$\mathbf{A}_1 \theta = \frac{1}{2} \mathbf{p}_1 \tag{34}$$

其中，

$$\mathbf{A}_1 = \begin{bmatrix} x_1 & y_1 & -0.5 \\ x_2 & y_2 & -0.5 \\ \vdots & \vdots & \\ x_N & y_N & -0.5 \end{bmatrix} \boldsymbol{\theta} = \begin{bmatrix} x \\ y \\ s \end{bmatrix}, \mathbf{p}_1 = \begin{bmatrix} k_1 - \hat{d}_1^2 \\ k_2 - \hat{d}_2^2 \\ \vdots \\ k_n - \hat{d}_N^2 \end{bmatrix} \tag{35}$$

s 作为向量的一部分是未知的。在 5-E 部分我们会展示它如何作为限制条件求出 x。在经过进一步的数学操作之后，我们得到替代的 LS 的解如下，

$$\mathbf{A}_2 \mathbf{x} = \frac{1}{2} \mathbf{p}_2, \mathbf{x} = \frac{1}{2} \left(\mathbf{A}_2^T \mathbf{A}_2 \right)^{-1} \mathbf{A}_2^T \mathbf{p}_2 \tag{36}$$

其中，

$$\mathbf{A}_2 = \begin{bmatrix} x_1 & x_2 & \cdots & x_N \\ y_1 & y_2 & \cdots & y_N \end{bmatrix}^T, \mathbf{p}_2 = \begin{bmatrix} s + k_1 - \hat{d}_1^2 \\ s + k_2 - \hat{d}_2^2 \\ \vdots \\ s + k_n - \hat{d}_N^2 \end{bmatrix} \tag{37}$$

注意到上式中 x 用 s 的形式表达，产生了一个二次表达式。因此需要用根选择方法找到 x[26]。然而因为因为等式的不一致性，（36）是不准确的。尽管如此，它可以作为初始位置的预测来提高更好的算法定位的表现，我们将在下一部分谈到。

C. 通过泰勒展开式线性化 NLS 解

非线性表达式 d(x)可以用泰勒展开式在参考点 x0 处展开.如果高阶忽略，我们可以得到如下[33]，

$$\mathrm{d}(\mathrm{x}) \approx \mathrm{d}\left(\mathrm{x}_0\right)+\mathrm{H}_0\left(\mathrm{x}-\mathrm{x}_0\right) \tag{38}$$

其中 d(x)在 x0 点的雅可比矩阵如下，

$$\mathbf{H}_0=\left[\begin{array}{cccc} \frac{\partial d_1}{\partial x} & \frac{\partial d_2}{\partial x} & \cdots & \frac{\partial d_N}{\partial x} \\ \frac{\partial d_1}{\partial y} & \frac{\partial d_2}{\partial y} & \cdots & \frac{\partial d_N}{\partial y} \end{array}\right]^T_{\mathbf{x}=\mathbf{x}_0} \tag{39}$$

为了让(38)有效，参考点 x0 应该足够接近真实点。把（38）代入（33）我们有了一个可以用矩阵形式表示的线性系统，并且可以用线性 LS 求解。一个更准确的迭代技术是把这个 LS 预测作为中间预测，插入（38）并重新线性化这个系统，直到收敛。

D. 一个替代的线性最小二乘解

在前一部分讨论的非线性模型包含了用 x 和 y 的二次形式表示的 s,为了获得线性的模型，[34]提出了一种技巧用来取消这些非线性项。通过固定（8）中第 r 给 FT 的表达式并从剩余表达式中减去，再重新组织这些项，我们可以得到如下的线性模型。

$$\mathbf{A}_3\mathbf{x}=\frac{1}{2}\mathbf{p}_3 \tag{40}$$

其中

$$\mathbf{A}_3=\begin{bmatrix} x_1-x_r & y_1-y_r \\ x_2-x_r & y_2-y_r \\ \vdots & \vdots \\ x_N-x_r & y_N-y_r \end{bmatrix}, \mathbf{p}_3=\begin{bmatrix} \hat{d}_r^2-\hat{d}_1^2-k_{r,1} \\ \hat{d}_r^2-\hat{d}_2^2-k_{r,2} \\ \vdots \\ \hat{d}_r^2-\hat{d}_N^2-k_{r,N} \end{bmatrix} \tag{41}$$

其中 Kr,i=Kr-Ki,r 是用来获得线性模型的参考 FT。注意到 p2 中的非线性项，x，y 的二次型在 p3 中取消了。从上述的表达式中，LS 的解可以被写为，

$$\mathbf{x}=\frac{1}{2}\left(\mathbf{A}_3^T\mathbf{A}_3\right)^{-1}\mathbf{A}_3^T\mathbf{p}_3 \tag{42}$$

注意到（8）式定义了每个 FT 周围的一个圆，而二次项在（40）中被取消，产生了一个线性表达，可以被视为连接圆的交点的直线，4 图是几何表示。

为了获得 LLS 预测器准确度的更深入的理解，值得分析 p3 中的扰动。通过把（1）待人并假设偏差项为 0，我们有

$$\mathbf{p}_3=\mathbf{p}_3^{(c)}+\mathbf{p}_3^{(n)} \tag{43}$$

其中，常数和噪音项由下式给出。

$$\mathbf{p}_3^{(c)} = \begin{bmatrix} d_r^2 - d_1^2 - k_r + k_1 \\ d_r^2 - d_2^2 - k_r + k_2 \\ \vdots \\ d_r^2 - d_N^2 - k_r + k_N \end{bmatrix} \tag{44}$$

$$\mathbf{p}_3^{(n)} = \begin{bmatrix} 2d_r n_r - 2d_1 n_1 + n_r^2 - n_1^2 \\ 2d_r n_r - 2d_2 n_2 + n_r^2 - n_2^2 \\ \vdots \\ 2d_r n_r - 2d_N n_N + n_r^2 - n_N^2 \end{bmatrix} \tag{45}$$

注意到当我们摆脱了二次项 s 时，p3 中的噪声项却变多了。具体来说，我们可以生成随着 MT 远离 FT，上式的准确度在下降，因为有一个与距离有关的噪声项。如果 p3->0，所有的线会交于一点。对于 LLS-1 的 MSE 的理论偏差，可以参考[35]。

1）平均技巧：（40）的 LLS-1 预测器仅仅通过$\hat{d}_r^2 - \hat{d}_i^2$利用了测量量 d 因此 LLS-1 的测量集有效地变成了如下，

$$\tilde{d}_i = \hat{d}_r^2 - \hat{d}_i^2, \quad i = 1, \ldots, N, \quad i \neq r \tag{46}$$

注意到因为（46）地有效测量与（1）中的有效测量不同，同样意味着图四中对应的 CRLB 也会不同，这在[36]中推出。

在[20]提出的另一个 LLS 方式中，通过把每一个独立等式代入所有其余等式中，获得了 N*（N-1）/2 个线性等式。换句话说，在 LLS-2 技巧中，使用了下述观察来进行位置预测：

$$\check{d}_{ij} = \hat{d}_i^2 - \hat{d}_j^2, \quad i, j = 1, 2, \ldots, N, i < j \tag{47}$$

和 LLS-1 相似，在 LLS-2 中，（42）中的线性 LS 解被用来获得目标节点的位置。

在（37）提出的第三个 LLS 技巧中，并没有像 LLS-1 和 LLS-2 那样直接获得等式的不同，首先得到了测量距离 d 的均值，然后从所有其他等式中减去这个平均值，之后像（42）那样获得目标节点的线性 LS 的解。LLS-3 用的观察集可以表示如下，

$$\bar{d}_i = \hat{d}_i^2 - \frac{1}{N}\sum_{j=1}^{N} \hat{d}_j^2, \quad i = 1, 2, \ldots, N \tag{48}$$

参考 FT 选择：LLS-1 中任意选择了一个 FT，然而观察 p 中的噪声项，p3 中的所有行都与 MT 和 FT 之间的距离有关。如果 FT 远离 MT，这意味着 p3 中的该项噪声更大，降低了预测的准确度。因此，如何选择 FT 可能会很大程度上影响预测器的均方差。一个简单的技巧选取 LOS 场景下的 FT 以改善预测准确度是选择所有距离测量中最小的。参考 FT 的下标由[38]给出

$$r = \arg\min_i \left\{ \hat{d}_i \right\}, \quad i = 1, 2, \ldots, N \tag{49}$$

然后，矩阵 A3 和向量 p3 可以通过使用选择的参考 FT 获得，我们把该预测器称为 LLS-RS。举个例子，图一中，FT-1 被用来从非线性表达式中获得线性模型因为 d1 是所有测量距离中最小的。

2）利用协方差矩阵：尽管参考 FT 提升了位置预测的准确度，它并没有解释 ps 各行之间的关系，而这在线性化过程中是相关的。[39]中讨论过，ML 预测器是在有相关观测时的最优预测器。首先，考虑对一个 LOS 场景的（43）中的关系的以下修改，

$$\mathbf{p}_3 = \mathbf{A}_3\mathbf{X} + \mathbf{p}_3^{(n)} \tag{50}$$

其中，x 是 MT 的实际位置，因此$p_3^{(c)} = A_3\tilde{x}$。之后基于（50），这个线性模型的 MLE^2 可以写为[39],

$$\mathbf{x} = \left(\mathbf{A}_3^T\mathbf{C}^{-1}\mathbf{A}_3\right)^{-1}\mathbf{A}_3^T\mathbf{C}^{-1}\mathbf{p}_3 \tag{51}$$

其中 C=Cov（p_3^n），是p_3^n的协方差矩阵。当所有的 FT 都是 LOS 场景时，$\mathbf{p}_n$的协方差矩阵可以写为

$$\mathbf{C} = 4d_r^2\sigma^2 + 2\sigma^4 + \mathrm{diag}\left\{4\sigma^2 d_1^2 + 2\sigma^4, \ldots 4\sigma^2 d_i^2 + 2\sigma^4, \ldots, 4\sigma^2 d_N^2 + 2\sigma^4\right\} \tag{52}$$

其中 diag{$\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots., \boldsymbol{\lambda}_N$}是对角矩阵。注意到，实际中 di 并不可用，可以使用有噪声的测量距离分析协方差矩阵。

3）仿真结果：使用了蒙特卡洛仿真来比较不同的 LLS 预测器。如图二，4 个 FT 固定在方形房间的角落。图五（ａ）显示了 LLS-1 的二维 MSE，当 FT1 作为参考点时。我们观察到当 MT 靠近 FT 时，MSE 在变小。图五（ｂ）中的仿真结果是 MT 的位置以１０ｍ为间隔在［－４０，４０］之间变化，产生了一个９＊９的 MT 可能位置的方格。不同方法在方格中所有位置 MSE 都模拟出来然后做平均处理。结果显示，相比于其他的方法，LLS－１表现最差，LLS-２和 LLS-３比 LLS-１表现稍好一些，并且他们的 MSE 一致。然而，他们都没有 LLS－RS 好。MLE 比 LLS-RS 稍好并且很接近 CRLB。

E. 受约束的加权最小二乘法

在[32]提出的受约束最小二乘法应用在（34）中来寻找 MT 的位置。更具体地说，它使用了 s 和 x 之间的关系作为约束条件，并基于拉格朗日多项式发展出了一种方法。受约束的最优化问题被公式化为，

$$\boldsymbol{\theta}_{cw} = \arg\min_{\boldsymbol{\theta}}\left(\mathbf{A}_1\boldsymbol{\theta} - \mathbf{p}_1\right)^T\mathbf{W}\left(\mathbf{A}_1\boldsymbol{\theta} - \mathbf{p}_1\right) \tag{53}$$

受限于$\mathrm{s} = x^2 + y^2$，如

$$\mathbf{q}^T\boldsymbol{\theta} + \boldsymbol{\theta}^T\mathbf{p}\boldsymbol{\theta} = 0 \tag{54}$$

其中 A1，θ和 p1 同（35）相同，且

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \text{ and } \mathbf{q} = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} \tag{55}$$

为了确定最优的权重矩阵 W，作者检查了 p1 的干扰。当高信噪比时，我们有，

$$\hat{d}_i^2 = \left(d + n_i\right)^2 \approx d_i^2 + 2d_i n_i \text{ for } i = 1, 2, \ldots, N \tag{56}$$

这表示干扰$\epsilon_i = \hat{d}_i^{\,2} - d_i^2 = 2d_i n_i$，而且可以表示为向量形式$\epsilon =$ $[2d_1n_1, 2d_2n_2, \ldots\ldots, 2d_Nn_N]^T$。干扰的协方差由下式给出，

$$\Psi = \mathrm{E}\left[\grave{o}\grave{o}^T\right] = \mathbf{BQB} \tag{57}$$

其中 B=diag(d1,d2,....,dn)，最有权重矩阵$W = \varphi^{-1}$,注意到它依赖于 MT 和 FT 之间的实际距离 d，而这时未知的，一个近似的权重矩阵可以通过$\hat{B} = diag(2\widehat{d_1}, 2\widehat{d_2}, \ldots., 2\widehat{d_N})$获得，而不是 B。

CWLS 问题可以通过最小化拉格朗日解决，如下，

$$\mathrm{L}(\boldsymbol{\theta}, \lambda) = \left(\mathbf{A}_1\boldsymbol{\theta} - \mathbf{p}_1\right)^T \boldsymbol{\Psi}^{-1}\left(\mathbf{A}_1\boldsymbol{\theta} - \mathbf{p}_1\right) + \lambda\left(\mathbf{q}^T\boldsymbol{\theta} + \boldsymbol{\theta}^T\mathbf{P}\boldsymbol{\theta}\right) \tag{58}$$

其中 λ 时拉格朗日乘子。[32]中显示局部或全局最优解由下式给出，

$$\boldsymbol{\theta}_{cw} = \left(\mathbf{A}_1^T\boldsymbol{\Psi}^{-1}\mathbf{A}_1 + \lambda\mathbf{P}\right)^{-1}\left(\mathbf{A}_1^T\boldsymbol{\Psi}^{-1}\mathbf{p}_1 - \frac{\lambda}{2}\mathbf{q}\right) \tag{59}$$

其中 λ 由五根等式决定。

在另一个相关工作中，作者提出了协方差塑性（CSLS）来预测位置，当处于低信噪比时，取得的效果要好于其他 LS 方法。

六．NLOS 场景：基本限制和最大似然解

在通常情况下，尤其是室内，一些 FT 和 MT 之间的直线通道往往会被阻挡。这些 NLOS 的 FT 会严重降低预测精准度。最简单的方式来减轻 NLOS 的影响是识别并舍弃 NLOS 的 FT，然后使用之前讨论的 LOS 的方法中的一个进行预测。然而，总是有可能误警或漏报，而这回降低准确度。在本部分，我们会回顾替代的减轻 NLOS 影响的方法。首先会讨论基于 ML 的方法和 NLOS 场景下的 CRLB。

A. 基于 ML 的算法

在[19]和[23]讨论了减轻 NLOS 影响的 ML 方法，这需要 NLOS 误差分布的先验知识。例如,[19]提出了一个 MT 可能位置的解，假设 NLOS 的偏差为指数分布，参数为 λ 。因为大多数情况下 NLOS 偏差比高斯误差大，下式简化假设

$$\hat{d}_i = d_i + \begin{cases} b_i, & i = 1, 2, \ldots, N_{NL} \\ n_i, & i = N_{NL} + 1, \ldots, N \end{cases} \tag{60}$$

其中N_{NL}是 NLOS 的 FT 的数量，$N_L = N - N_{NL}$表示 LOS 的 FT 的数量。因此，一个简化的 ML 解如下，

$$\mathbf{x} = \arg\min_{\mathbf{x}}\left\{\sum_{i=1}^{N_{NL}} \lambda_i\left(\hat{d}_i - d_i\right) + \sum_{i=N_{NL}+1}^{N} \frac{\left(\hat{d}_i - d_i\right)^2}{2\sigma_i^2}\right\} \tag{61}$$

同样有可能获得精确的决定规则考虑高斯和指数分布变量的峰值，有如下的概率密度函数，

$$P(x) = \lambda\exp\left(-\lambda\left(x - \lambda\sigma^2/2\right)\right)Q\left(\lambda\sigma - \frac{x}{\sigma}\right) \tag{62}$$

其中$Q(\alpha)=\frac{1}{\sqrt{2\pi}}\int_{\alpha}^{\infty}\exp(-x^2/2)\mathrm{d}x$表示 Q 函数，准确的 ML 解变成，

$$\mathrm{x}=\arg\min_{\mathbf{x}}\left\{\sum_{i=1}^{N_{NL}}\lambda_i\left(\hat{d}_i-d_i-\lambda_i\sigma_i^2/2\right)\sum_{i=1}^{N_{NL}}\log\left[Q\left(\lambda_i\sigma_i-\frac{\hat{d}_i-d_i}{\sigma_i}\right)\right]+\sum_{i=N_{NL}+1}^{N}\frac{\left(\hat{d}_i-d_i\right)^2}{2\sigma_i^2}\right\} \tag{63}$$

另一个基于 TOA 系统利用 ML 方法的减轻 NLOS 影响的方法在［2 3］中介绍。作者为不同的 FT 集合考虑了一些假设，之后利用 ML 原则，选择最好的集合用来预测位置，用来预测位置的最好 FT 的集合下标由下式给出，

$$\hat{i}=\arg\min_i\left[\ln\gamma^{-1}(i)+\sum_{k\in S^{LOS}}\frac{N_{trn}}{2}\frac{\hat{\sigma}_k^2(i)}{\sigma_k^2}\right] \tag{64}$$

其中S_i^{LOS}表示被假设为 LOS 的 MT 的第 i 个集合，并且根据先验概率分配 γ（i）。同样，

$$\hat{\sigma}_k^2(i)=\frac{1}{N_{\mathrm{trn}}}\sum_{m=1}^{N_{\mathrm{trn}}}\left(t_{k,m}-\left\|\mathbf{x}(i)-\mathbf{x}_k\right\|\right)^2 \tag{65}$$

表示第 i 个假设下N_{trn}的 TOA 测量值与第 k 个 FT 相关的估计方差，t_k，m 表示第 k 个 FT 处的第 m 个 TOA 测量值。注意到，上述的方法要求N_{trn} TOA 测量的缓存以便于获得特定 FT 的噪声数据。一旦利用 ML 原则选出了 LOS 的 FT 的集合，就只使用 LOS 的 FT 和 ML 算法来预测 MT 位置。[23]的仿真结果显示这产生了更好的准确度相比于[18]中的残差权重方法，而且只比使用真正的 LOS 的 FT 要差一些。同样，在低信噪比和小 NLOS 偏差的情况下，仿真结果显示，不使用任何减轻 NLOS 的方法可能会好一些，为了不损害准确度。

B. **克拉美罗下限**

在 NLOS 场景下，克拉美罗下限依赖于是否有 NLOS 偏差的先验信息。首先考虑没有 NLOS 偏差先验信息的情况，只直到 NLOS 的 FT。之后，FIM 由下式给出，

$$\mathbf{I}\left(\mathbf{x}_b\right)=\mathbf{AI}(\mathbf{d})\mathbf{A}^T \tag{66}$$

其中，

$$\mathbf{x}_b=\left[x,y,\quad b_1,\quad b_2,\ldots b_{N_{NL}}\right]^T \tag{67}$$

是一个$(N_{NL}+2)\times 1$有未知参数的向量，其中，

$$\mathbf{A}=\frac{\partial \mathbf{d}}{\partial \mathbf{x}_b}=\begin{bmatrix} \frac{\partial d_1}{\partial x} & \frac{\partial d_2}{\partial x} & \cdots & \frac{\partial d_{N_{NL}}}{\partial x} & \cdots & \frac{\partial d_N}{\partial x} \\ \frac{\partial d_1}{\partial y} & \frac{\partial d_2}{\partial y} & \cdots & \frac{\partial d_{NLL}}{\partial y} & \cdots & \frac{\partial d_N}{\partial y} \\ \frac{\partial d_1}{\partial b_1} & \frac{\partial d_2}{\partial b_1} & \cdots & \frac{\partial d_{N_{NL}}}{\partial b_1} & \cdots & \frac{\partial d_N}{\partial b_1} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial d_1}{\partial b_{N_{NL}}} & \frac{\partial d_2}{\partial b_{N_{NL}}} & \cdots & \frac{\partial d_{N_{NL}}}{\partial b_{N_{NL}}} & \cdots & \frac{\partial d_N}{\partial b_{NNL}} \end{bmatrix} \tag{68}$$

且，

$$\mathbf{I}(\mathbf{d})=\mathrm{E}_{\mathbf{d}}\left[\frac{\partial \ln f(\mathbf{d}|\ \mathbf{d})}{\partial \mathbf{d}}\left(\frac{\partial \ln f(\mathbf{d}|\ \mathbf{d})}{\partial \mathbf{d}}\right)^T\right] \tag{69}$$

正如在[42]中讨论的，$\tilde{A}$可以被写为 LOS 和 NLOS 项的形式，

$$\mathbf{A}=\begin{bmatrix} \mathbf{A}_{NL} & \mathbf{A}_L \\ \mathbf{I}_{N_{NL}} & \mathbf{0}_{N_{NL},N_L} \end{bmatrix} \tag{70}$$

其中$I_{N_{NL}}$和 $0_{N_{NL}}$ 是单位矩阵和零矩阵。并且，

$$\mathbf{A}_{NL}=\begin{bmatrix} \cos\alpha_1 & \cos\alpha_2 & \ldots & \cos\alpha_{N_{NL}} \\ \sin\alpha_1 & \sin\alpha_2 & \ldots & \sin\alpha_{N_{NL}} \end{bmatrix} \tag{71}$$

$$\mathbf{A}_L=\begin{bmatrix} \cos\alpha_{N_{NL}+1} & \cos\alpha_{N_{NL}+2} & \ldots & \cos\alpha_N \\ \sin\alpha_{N_{NL}+1} & \sin\alpha_{N_{NL}+2} & \ldots & \sin\alpha_N \end{bmatrix} \tag{72}$$

相似的,**I(d)**可以被写为 NLOS 和 LOS 项的形式，

$$\mathbf{I}(\mathrm{d})=\begin{bmatrix} \mathbf{\Lambda}_{NL} & 0 \\ 0 & \Lambda_L \end{bmatrix} \tag{73}$$

其中$\mathbf{\Lambda}_{NL}=\mathrm{diag}\left(\sigma_1^{-2},\ldots,\sigma_{N_{NL}}^{-2}\right)$和$\mathbf{\Lambda}_L=\mathrm{diag}\left(\sigma_{N_{NL+1}}^{-2},\ldots,\sigma_{N_N}^{-2}\right)$。在一些变换之后。I($x_b$)可以由下式得到，

$$\mathbf{I}\left(\mathbf{x}_b\right)=\begin{bmatrix} \mathbf{A}_{NL}\mathbf{\Lambda}_{NL}\mathbf{A}_{NL}^T+\mathbf{A}_L\mathbf{\Lambda}_L\mathbf{A}_L^T & \mathbf{A}_{NL}\mathbf{\Lambda}_{NL} \\ \mathbf{\Lambda}_{NL}\mathbf{A}_{NL}^T & \mathbf{\Lambda}_{NL} \end{bmatrix} \tag{74}$$

注意到（74）既依赖于 NLOS 又依赖于 LOS 信号。然而[42]中进一步证明 MT 位置的 CRLB 由下式给出，

$$E\left[\left(\mathbf{x}_b-\mathbf{x}_b\right)\left(\mathbf{x}_b-\mathbf{x}_b\right)^T\right]\geq\mathbf{I}^{-1}\left(\mathbf{x}_b\right)=\left(\mathbf{A}_L\mathbf{\Lambda}_L\mathbf{A}_L^T\right)^{-1} \tag{75}$$

换句话说，CRLB 特别的依赖于 LOS 信号，如果 NLOS 的 FT 可以被准确识别。因此，ML 预测器可以在 NLOS 场景下实现 CRLB 首先要识别并抛弃 NLOS 的 FT，然后用 LOS 的 FT 预测位置，如图六（ a ）中所示。

如果有其他的关于 NLOS 偏差 b 的相关信息，可以获得更好的定位准确度。然后，泛化的 CRLB 可以表达为，

$$E\left[\left(\mathrm{x}_b-\mathrm{x}_b\right)\left(\mathrm{x}_b-\mathrm{x}_b\right)^T\right]\geq\left(\mathbf{I}\left(\mathrm{x}_b\right)+\begin{bmatrix}0 & 0\\ 0 & \Omega\end{bmatrix}\right)^{-1} \tag{76}$$

$$=\left(\begin{bmatrix}\mathbf{A}_{NL}\mathbf{\Lambda}_{NL}\mathbf{A}_{NL}^T+\mathbf{A}_L\mathbf{\Lambda}_L\mathbf{A}_L^T & \mathbf{A}_{NL}\mathbf{\Lambda}_{NL}\\ \mathbf{\Lambda}_{NL}\mathbf{A}_{NL}^T & \mathbf{\Lambda}_{NL}+\mathbf{\Omega}\end{bmatrix}\right)^{-1} \tag{77}$$

其中$\mathbf{\Omega}=\mathrm{diag}\left(\tilde{\sigma}_1^2,\ldots,\tilde{\sigma}_{N_{NL}}^2\right)$，且 σ 可以看作 b i 的方差。作为 G-CRLB 的上限，当方差无限大时，Ω－＞0 时，G-CRLB 下降至 CRLB（因为实际上没有 b i 的信息），MAP 预测器可以渐进的到达 G-CRLB，它使用了 NLOS 偏差的数据，如图六（b）。

七．NLOS 场景：最小二乘法

用来预测位置的 LS 方法可以修改后一直 NLOS 偏差的影响，如分配权重。在本部分，会简要讨论带权最小二乘法和残差权重法。

A. 带权最小二乘法

一个用来减小 NLOS 影响的简单方式是给 NLOS 的 FT 分配更小的权重。假设 NLOS 的 FT 测量距离的方差更大，这些方差的逆用来作为（3 3）中的可信矩阵。这事实上由如下 ML 解推出。对于 ML 算法，位置预测由下式给出，

$$\mathbf{x}_{ML}=\arg\max_{\mathbf{x}}\mathrm{p}(\mathbf{d}|\ \mathbf{x}) \tag{78}$$

其中，

$$\mathrm{p}(\mathbf{d}|\ \mathrm{x})=\mathrm{p}_n(\mathbf{d}-\mathrm{d}|\ \mathrm{x}) \tag{79}$$

如果噪声是高斯分布，我们有，

$$\mathrm{pn}_n(n)=\frac{1}{\sqrt{2\pi}\sigma_i}\exp\left(-\frac{n^2}{2\sigma_i^2}\right) \tag{80}$$

然后，联合概率函数变成了，

$$\mathrm{p}(\mathbf{d}|\ \mathrm{x})=\frac{1}{(2\pi)^{N/2}\prod_{i=1}^{N}\sigma_i}\exp\left(-\sum_{i=1}^{N}\frac{\left(\hat{d}_i-\|\mathbf{x}-\mathbf{x}_i\|\right)^2}{2\sigma_i^2}\right) \tag{81}$$

在进一步变换（81）后，ML 解变得等价于，

$$\mathbf{x}_{ML}=\arg\min_{\mathbf{x}}\sum_{i=1}^{N}\frac{\left(\hat{d}_i-\|\mathbf{x}-\mathbf{x}_i\|\right)^2}{\sigma_i^2} \tag{82}$$

它同样等价于当$\beta_{i=}1/\sigma_i{}^2$时 WLS 的解。

然而，对于静止的 MT，TOA 测量量的方差对 LOS 和 NLOS 来说差别并不是很大。当然，NLOS 距离测量偏差可能降低定位准确度。因此，［4 3］和［4 4］提出了一个替代的带

权方法，使用接受信号多路部分的特定数据。具体来说，接收信号的峰度，平均过度延迟和均方根延迟被用来分析接收信号为 LOS 的似然值似然值然后被用来分析权重β。

B. 残差权重算法

[18]提出的残差权重算法是基于这样的观察，当预测 MT 位置时用到了 NLOS 的 FT，残差会变大。通过假设有超过三个可用 FT，RWGH 预测 MT 位置，详述见表 2。
[18]通过仿真展示了 RWGH 比用 MRE 选择位置表现的要好。[45]提出了一个 RWGH 次优版本，它计算花费小。在那篇文章中，并没有考虑所有 FT 的组合（如果 N 很大的话会是一个极大的数）。首先，考虑所有（N-１）FT 的组合来计算位置预测的中间值和相应的残差。之后，在 N 个不同的组合中，表现最好的预测器没有用到的 FT 被舍弃（对应的组合有最小的残差）。一直迭代该过程直到先定的停止规则（比如到达最小 FT 个数或残差的改变很小）

八. NLOS 场景：有约束的定位方法

在本部分，会简要讨论不同类型 NLOS 缓解算法利用一些和 NLOS 测量相关的限制条件。

A. 约束 LS 算法与二次规划

在４－C 部分讨论的二步 ML 算法对 NLOS 的影响鲁棒性差。［４６］提出了用于 NLOS 场景的二次规划法。数学规划公式如下，

$$\boldsymbol{\theta}_{cw} = \arg\min_{\boldsymbol{\theta}} \left(\mathbf{A}_1\boldsymbol{\theta} - \mathbf{p}_1\right)^T \Psi^{-1} \left(\mathbf{A}_1\boldsymbol{\theta} - \mathbf{p}_1\right) \tag{86}$$

$$\text{s.t. } \mathbf{A}_1\boldsymbol{\theta} \le \mathbf{p}_1 \tag{87}$$

其中，A１，ｐ１和θ和（３５）一样。注意到（８６）和（８７）构成了有约束的 LS 算法，可以用二次规划法解决。CLS 的直观解释是（86）找到了 MT 位置的 WLS 解，而（87）的约束条件将 LOS 情况下的等式放宽到了 NLOS 的不等式。在[46]，引入了进一步修改阶段来包含 s 和 x 之间的依赖关系。

B. 线性规划

在[20]和[47]中引入了线性规划法，它假设有充足的先验知识来确定 LOS 和 NLOS 的 FT。和确定抛弃算法不同，它并没有抛弃 NLOS 点，而是用他们搭建了一个 MT 位置的线性区域。而 MT 的位置是由线性规划法仅仅利用 LOS 的 FT 获得，但是位置必须要在可行区域内。
首先，［２０］用仅利用 LOSFT 的线性规划法重新表达了（４２）中的 LS 预测器，目标是在特定约束下最小化线性目标函数，结果显示表现和（４２）近似。
如果第ｉ个 FT 被确定为 NLOS，我们有非线性约束，

$$\left\|\mathrm{x} - \mathbf{x}_i\right\| \le \hat{d}_i \tag{88}$$

这在[46]中被用来公式化二步 ML 算法的二次规划法（没有假设 NLOS 的 FT 被正确确定），为了线性化（88）的约束条件，我们把他们放松为，

$$x - x_i \le \hat{d}_i, \quad -x + x_i \le \hat{d}_i \tag{89}$$

$$y - y_i \le \hat{d}_i, \quad -y + y_i \le \hat{d}_i \tag{90}$$

其中，i=1,2,....N。本质上来说，上式是为了线性化等式而把圆约束放松为矩形约束，如图七所示。通过定义一些松弛变量，可以把（89）和（90）转化为定义可行域的等式。然后，仅仅用 LOS 的 FT 最小化有约束（限制在由 LOS 和 NLOS 获得的可行区域内）的目标函数求出 MT 的位置。

C. **几何约束位置预测**

在[49]提出了几何约束预测，它使用了 4-C 部分的两步 ML 算法外加一些参数来包含 FT 的几何关系（仅仅考虑有三个 FT 的场景）。将三个圆的交点定义为X_a, X_b, X_c。然后，一个被定义为虚拟距离的有约束的损失函数如下，

$$\gamma = \sqrt{\frac{1}{3}\left[\left\|\mathrm{x} - \mathrm{x}_A\right\|^2 + \left\|\mathrm{x} - \mathrm{x}_B\right\|^2 + \left\|\mathrm{x} - \mathrm{x}_C\right\|^2\right]} \tag{91}$$

对于 MT 的期望位置X_e，我们可以计算出（91）的期望距离$\gamma_e = \gamma + n_e$，其中 Ne 表示为期望虚拟距离的噪声。X_e的坐标被选为，

$$\mathbf{x}_e = \omega_1 \mathbf{x}_A + \omega_2 \mathbf{x}_B + \omega_3 \mathbf{x}_C \tag{92}$$

其中，权重由下式获得，

$$\omega_i = \frac{\sigma_i^2}{\sigma_1^2 + \sigma_2^2 + \sigma_3^2}, \text{ with } i = 1, 2, 3 \tag{93}$$

基本上，它假设对于 NLOS 的 FT 来说偏差更大。因此（93）定义的权重会把$\mathbf{x}_e$移向 NLOS 的 FT 圆的中心。

通过更新 A1 和 p1，这些几何约束条件被包含在 4-C 中的二步 ML 算法中。

$$\mathbf{A}_1 = \begin{bmatrix} \mathbf{A}_1 & \\ \gamma_x \quad \gamma_y & -0.5 \end{bmatrix}, \text{ and } \mathbf{p}_1 = \begin{bmatrix} \mathbf{p}_1 \\ \gamma_k - \gamma_e^2 \end{bmatrix} \tag{94}$$

其中，

$$\gamma_x = \frac{1}{3}\left(x_A + x_B + x_C\right) \tag{95}$$

$$\gamma_y = \frac{1}{3}\left(y_A + y_B + y_C\right) \tag{96}$$

$$\gamma_k = \frac{1}{3}\left(x_A^2 + x_B^2 + x_C^2 + y_A^2 + y_B^2 + y_C^2\right) \tag{97}$$

这些几何约束条件同样被包含在别的变量中，

$$\mathbf{B} = \mathrm{diag}\left(d_1, d_2, d_3, \gamma\right) \tag{98}$$

$$\mathbf{n}=\left[n_1,n_2,n_3,n_\gamma\right]^T \tag{99}$$

然后，[29]中的二步 ML 算法使用更新后的带几何约束的变量来求解 MT 的位置。

D. 内点最优化

在[32]提出了一个内点最优化法在有 NLOS 偏差的情况下找出最优位置预测。利用泰勒近似展开的线性系统，一个线性测量向量被定义如下，

$$\mathbf{y}=\mathbf{H}_0\mathrm{x}+\mathbf{b}+\mathbf{n} \tag{100}$$

其中 H0 在（39）中定义。如果忽略 NLOS 的偏差，无偏差位置预测由[33]给出，

$$\mathbf{x}=\left(\mathbf{H}_0^T\mathbf{Q}^{-1}\mathbf{H}_0\right)^{-1}\mathbf{H}_0^T\mathbf{Q}^{-1}\mathbf{y} \tag{101}$$

如果偏差向量 b 已知，一个更准确的无偏差位置预测由下式给出，

$$\mathbf{x}=\mathrm{x}+\mathbf{Vb} \tag{102}$$

其中，

$$\mathbf{V}=-\left(\mathbf{H}_0^T\mathbf{Q}^{-1}\mathbf{H}_0\right)^{-1}\mathbf{H}_0^T\mathbf{Q}^{-1} \tag{103}$$

是一个偏移修正值矩阵。然而，实际上 b 是未知的而且需要预测。为了从（100）中预测 b,观测偏差矩阵定义如下，

$$\mathrm{z}=\mathbf{y}-\mathbf{H}_0\mathbf{x} \tag{104}$$

这可以被简化为$\mathrm{z}=\mathrm{Sb}+\mathrm{w}$,其中$\mathrm{S}=\mathrm{I}+H_0V$，而且偏移噪声由下式给出，

$$\mathbf{w}=\mathbf{H}_0(\mathbf{x}-\mathbf{x})-\mathbf{n} \tag{105}$$

然后，有约束的最优化问题被定义为预测 NLOS 偏差的误差，

$$\mathbf{b}=\arg\min_{\mathbf{b}}(\mathbf{z}-\mathbf{Sb})^T\mathbf{Q}_w^{-1}(\mathbf{z}-\mathbf{Sb}) \tag{106}$$

$$\text{s.t.}\quad b_i\in B_i,\quad i=1,2,\ldots,N \tag{107}$$

其中，B=[l,u]是范围 bi 的先验知识，下限 l>=0 上限由 ui 定义，且 Qw 是 w 的协方差矩阵。为了解决（106）和（107）的带约束最优化问题，[33]使用了 IPO 法。具体来说，（106）和（107）被修改为，

$$\mathbf{b}=\arg\min_{\mathbf{b}}(\mathbf{z}-\mathbf{Sb})^T\mathbf{Q}_w^{-1}(\mathbf{z}-\mathbf{Sb}) \tag{108}$$

$$\text{s.t.}\quad g_i\left(b_i\right)-s_i=0,\ \text{and}\ s_i>0, i=1,\ldots,N \tag{109}$$

其中s_i是松弛变量，$g_i(b_i)$是满足$g_i(b_i)>0$的障碍函数。一个常用的满足要求的平滑二阶函数是$g_i(b_i)=(u_i-b_i)/(b_i-l_i)$ 。然后，（108）和（109）通过最小化拉格朗日函求解，

$$\mathrm{L}(\mathbf{b},\lambda,\mathbf{s})\ =(\mathbf{z}-\mathbf{Sb})^T\mathbf{Q}_w^{-1}(\mathbf{z}-\mathbf{Sb})-\mu\sum_{i=1}^{N}\ln s_i-\lambda^T(\mathbf{g}(\mathbf{b})-\mathbf{s}) \tag{110}$$

其中$\mathbf{g}(\mathbf{b})$和$\mathbf{s}$通过堆叠$g_i(b_i)$和s_i后获得，变成 N*1 向量。注意到对数障碍函数，

$$\mu\sum_{i=1}^{N}\ln s_i \tag{111}$$

确保$s_i = g_i(b_i) > 0$且偏差误差一直在$[l_i,\ u_i]$。
（110）的解可以通过对 b，λ，s 做微分并将他们一起求解得到 b。一旦得到偏差向量 b 的预测值，作者将偏差修正矩阵代入（103）利用（102）去计算无偏位置。[33]的仿真结果显示通过 IPO 得到的结果相比于 RWGH 和迭代 LS 算法在 NLOS 的结果有更好的准确度。

九．NLOS 场景：位置预测的强健预测器

强健预测器通常被用来抑制数据中的极端值，而且不同类型的强健预测器已经被用来缓解 NLOS 的影响了。下面简要论述了几个常见的用于 NLOS 场景的强健预测器。

A. Huber M-estimator

M-预测器是 ML 类型的预测器，是一类用来缓解 NLOS 的健壮性预测器。如前所述，ML 算法尝试最大化函数，

$$\prod_{i=1}^{N} f\left(x_i\right) \tag{112}$$

等价于最小化$\sum_{i=1}^{N} -\log f(x_i)$。当有极端值时，ML 算法不能产生准确的结果。通用的被称作 M-预测器的 ML 算法的形式是由 HUBER 于 1964 引入且是为了最小化

$$\sum_{i=1}^{N}\rho\left(x_i\right) \tag{113}$$

其中$\rho(\cdot)$是凸函数。对于 HUBER　M-预测器来说，$\rho(\cdot)$被定义为，

$$\rho(\nu)=\begin{cases}\nu^2/2 & |\nu|\le\xi\\ \xi|\nu|-\xi^2/2 & |\nu|>\xi\end{cases} \tag{114}$$

不是严格凸函数，因此最小化目标函数导致了彼此靠近的多个解。
在［５１］中，M-预测器被用来预测 MT 的位置，产生了，

$$\mathbf{x}=\arg\min_{\mathbf{x}}\left\{\sum_{i=1}^{N}\rho\left(\left(\hat{d}_i-\left\|\mathbf{x}-\mathbf{x}_i\right\|\right)/\sigma_i\right)\right\} \tag{115}$$

仿真结果显示，M-预测器比传统的 LS 预测器表现要好，尤其是对大 NLOS 偏差误差来说。如果 M-预测器与ｂｏｏｔｓｔｒａｐ结合，准确度可以进一步提高。

B. 最小中位数平方

[37]提出了最小中位数平方来缓解 NLOS 影响，这是最常用的健壮性适应算法之一。它可以容忍高达５０％的极端值。用 LMS 预测的 MT 的位置由下式表示，

$$\mathbf{x} = \arg\min_{\mathbf{x}} \left\{ \mathrm{med}_i \left(\hat{d}_i - \left\| \mathbf{x} - \mathbf{x}_i \right\| \right)^2 \right\} \tag{116}$$

其中$med_i(\theta(i))$是所有可能$\theta(i)$的中间值。因为（１１６）的计算量过大，［３７］提出了一个低计算复杂度的方案，即使用 x 的随机子集去获得一些候选的 x，并选择残差中值最小的一个作为解。需要注意的是，[37]在安全方面使用了 LMS 算法，在对定位系统存在某些攻击时，某些度量可能是异常值。在[52]中报告了一种使用 LMS 算法进行 NLOS 缓解目的的并行方法。

C.其他健壮性预测选择

除了上述讨论的 M 预测和 LS 预测外，[53]还有其他的健壮性预测。就我们所知，并没有详细讨论缓解 NLOS 影响。最小修剪方差（LTS）打算最小化最小残差 n 的平方和。另一方面，S 预测器对异常稳健但它并不像 M 预测器那样高效。它希望找到一条最小化残差范围健壮性预测的线。最后，MM 预测器可以被视为 M 预测器，由 S 预测器给出的协方差开始。和其他两个预测器相比，MM 预测器有更大的计算复杂度，但同时更健壮也更高效。还有一些其他的参数化的鲁棒回归方法，用重尾分布代替正态分布，以模拟离群值。例如，在贝叶斯鲁棒回归算法中，t 分布可以是一个很好的分布。或者，可以考虑具有不同方差的零均值高斯分布（即污染正态分布）的混合来建模存在 NLOS 偏差时的噪声，例如

$$n_i \sim (1-\grave{o})\mathrm{N}\left(0, \sigma^2\right) + \grave{o}\mathrm{N}\left(0, c\sigma^2\right) \tag{117}$$

其中，第二个高斯分布旨在捕获异常值，而$\epsilon < 1$是一个小数(通常小于 0.1)，这描述了异常值和常数项 c 的影响。

十．NLOS 场景：基于识别和丢弃的技术

正如第六节开始讨论的，减轻 NLOS 影响的最简单的技术之一是识别 NLOS 的 FT 并在定位过程中丢弃它们(即，仅使用 LOS 的 FT 找到 MT 位置)。事实上，图 6(a)中所示的[42]的 ML 估计量和第 VI-A 节中讨论的[23]的 ML 估计量也是 IAD 类型的估计量。此外，如果 NLOS 的 FT 的权重设置为 0，LOS 的 FT 设置为 1，则第七节中讨论的 WLS 技术也成为 IAD 类型的技术。

所有这些方法中的一个共同问题是准确的 NLOS 的 FT 的识别。在本节中，我们将回顾另一种基于 IAD 的技术，该技术使用残差测试算法来识别 NLOS 的 FT。

A. **残差测试算法**

在[28]中提出的残差检验(RT)算法属于仅使用 LOS 的 FT 进行 MT 定位的算法组。因此，必须正确识别和丢弃 NLOS 的 FT，如下所示。首先，使用第 IV-D 节中讨论的 AML 算法，采用不同的 FT 组合，

$$S_0 = \sum_{i=3}^{N} NC_i \tag{118}$$

计算了不同的位置估计值$\hat{x}_k$ (k ∈ {1, 2, ..., S0}).对于每个 k，归一化残差的平方计算为，

$$\chi_x^2(k)=\frac{\left[\hat{x}_k-\hat{x}_{S_0}\right]^2}{I_x(k)},\chi_y^2(k)=\frac{\left[\hat{y}_k-\hat{y}_{S_0}\right]^2}{I_y(k)} \tag{119}$$

其中，Ix(k)和 Iy(k)从（17）中得到，为第 k 假设的 x 和 y 维数的 CRLB。不失一般性，s0 是所有 FT 都用于定位的情况。如果第 k 个假设中的所有 FT 都是 LOS，来自（119），我们有 ϰ x(k)~N（0,1）和 ϰ y(k)~N（0,1）。因此，ϰ 2x(k)和 ϰ 2y(k)具有一个自由度的集中式卡方分布。另一方面，如果在第 k 个假设中至少有一个 NLOSFT，则两个随机变量都有非集中的卡方分布，其非中心性参数取决于 NLOS 偏差。这一观察结果表明，如果 ϰ 2x(k)和 ϰ 2y(k)（理想情况下应该是相同的）的 PDF 可以被正确地识别出来，这就可以确定是否所有的 FT 都是 LOS。这可以简单地通过一个阈值测试来实现。例如，对于如[28]中所述的 7 种 FT 病例，描述 LOS 的卡方 PDF 的适当阈值被确定为 2.71。如果该阈值右侧 PDF 下的面积大于 0.1，则确定该随机变量为非集中卡方分布，否则则为集中卡方分布。如果 PDF 被确定为非集中式的，这意味着至少有一个 NLOS 的 FT。然后，算法形成$NC_{N-1}=N$集，每个集中有 (N−1)FT。对于每组，

$$\sum_{i=1}^{N-1}N-1C_i \tag{120}$$

得到了x_k的估计数。如果使用阈值检验发现这些集合中的任何一个按照集中的卡方分布进行分布，则 LOS 的 FT 的数量为(N−1)，并使用该特定集合内的 FT 来估计 MT 位置。否则，算法迭代直到最多有三个 LOS 的 FT。因为三个 FT 并不提供足够数量实现可靠的 RT，提出了增量测试程序。它首先需要两个 FT，然后将它们与其余的一个 FT 结合起来，以检查这三个 FT 是否都是 LOS。仿真结果表明，该技术优于 Rwgh[18]和 CLS[46]算法，当 LOS 的 CRLB 的数量大于 CRLB 总数的一半时，可以实现 CRLB。

十一．FT 分布对定位精度的影响

在结束调查之前，最后一个需要讨论的重要问题是关于 FT 分布对定位精度的影响。根据[21]，[56]中可实现的基本下界，分析了节点位置对精度的影响。在[56]中，作者得出结论，对于无锚定位，如果节点分布在 L1×L2 的矩形区域内，可实现的精度随着 L1→L2（即当区域收敛于一个正方形时）而提高。然而，对于大量的节点，网络形状对可实现精度的影响变得不显著。

在[21]中，提出了一种被称为重新定位的迭代算法来最优地放置参考节点。对于目标节点的固定位置，它会最优地放置参考节点，从而使 Cramer-Rao 边界最小化。并对该算法在目标节点的多个位置（如建筑物内的行走路径）进行了扩展。在[57]中，使用已知的最优解来评估 FTs 的三维放置的实际方面。考虑了一个在一个立方房间内放置四个 FT 的示例场景。将所有的 FTs 放置在一个平面上（例如，房间天花板的四个不同角落）可以产生相对较低的水平稀释的精度(HDOP)，但一个较大的垂直稀释的精度(VDOP)12。另一方面，如果目标节点被放置在一个“尽可能好”的四面体配置中，HDOP 相对较小，而 VDOP 明显小于平面配置。

在[58]中导出了不同数量 FT 的最优几何形状。一般来说，FTs 被放置在一个边角“平均”分布在一个单位球面上的几何图形上。对于 N=4、6、8、12、20，这个问题的五个解分别对应

于一个四面体、八面体、立方体、二十面体和十二面体，它们也被称为柏拉图固体。此外，任何中心柏拉图固体的叠加产生另一个最优几何[58]。

在[59]，[60]中，作者分析了定位概率与节点分布之间的关系。首先，将这些节点分为 I-节点和 NL 节点。假设 I 节点知道它们的位置，而假设 NL 节点不知道它们的位置（需要定位自己）。

I 节点和 NL 节点在两个节点中的分布维域S_{dom}r2 分别通过泊松点过程ρL和ρNL进行建模。然后，[59]推导出一个随机选择的、覆盖在S_{dom}上的 NL 节点被局部化的概率，以及在对数正态阴影衰落场景下，整个 NL 节点的网络被局部化的概率。

在半径为R_{cir}的圆形域上，d_{rad}<R_{cir}节点的定位失败概率，每个节点的无线电覆盖半径为R_{cir}，节点的k_{NL}的总数和节点总数 N 的紧密边界如下[60]所示

$$P_F \geq \left[1-(1-a)b^2\right]^{N-3}\left[1+b^2(1-a)(N-3)+b^4(1-a)^2\frac{(N-1)(N-2)}{2}\right] \quad (121)$$

其中=1–k_{NL}/N 是=节点占节点总数的比例，b=d_{rad}/R_{cir}。还提供了对对数正态阴影的扩展和过渡阈值的分析。

十二．总结

本文对不同的基于 TOA 的定位和 NLOS 缓解技术进行了广泛的调查。虽然一些算法可以执行接近 CRLB 的操作，但它们可能需要较高的计算复杂度和不同的先验信息的可用性。例如，在 NLOS 的情况下，关于 NLOS 偏倚的先验信息在许多情况下可能都不可用。在表三中，我们简要总结了不同的技术，以及它们的复杂性和需求。在存在 NLOS 偏倚的情况下，实用和有效的定位技术仍需进一步的研究。作者相信，这项调查将作为一个宝贵的资源，以评估优点和权衡的不同现有技术的发展并开发更有效实用的 NLOS 缓解算法的有效技术。

附录 C　源代码

附录 C.1　仿真 UI

```
Simulation
import tkinter.messagebox as messagebox
import matplotlib.pyplot as plt
from tkinter import ttk
from tkinter import *
import numpy as np
import threading
import analysis
import tkinter
import random
import time
import math
import NLOS
import PIL
import LOS

class simulation:

    def __init__(self, master=None):
        self.root = master
        self.root.geometry('1000x800')
        self.root.title('SIMULATION')
        self.para = {}
        self.para['MT_x'], self.para['MT_y'] = 500, 350 #init value
for convenience
        self.para['BT_Los_x'], self.para['BT_Los_y'] = [100, 800, 100,
800], [100, 100, 600, 600]
        self.para['BT_Nlos_x'], self.para['BT_Nlos_y'] = [600, 500,
400], [200, 150, 500]
        self.para['ave'], self.para['sigma'], self.para['Number_Los']
= 0, 5, 4
        self.para['lamda'], self.para['Number_Nlos'] = 5,3
        self.para['Number']=7
        self.para['t0']=random.random()*50
        self.para['map'] = 'White.png'
        self.para['BT_NLOS'], self.para['BT_LOS'], self.para['BT'] =
[], [], []
        self.Home()

    def get_BT(self, e1, e2, e3, e4):
        BT_x, BT_y, BT_NLOS,BT_LOS = [], [], [],[]
        try:
            for i in range(len(e1)):
                x = float(e1[i].get())
```

```
                BT_x.append(x)
                BT_y.append(y)
                d = ((self.para['MT_x'] - x) ** 2 + (self.para['MT_y']
- y) ** 2) ** 0.5
                n = random.gauss(self.para['ave'], self.para['sigma'])
                BT_LOS.append([x, y, d + n])

            self.para['BT_LOS']=BT_LOS
            self.para['BT_Los_x'] = BT_x
            self.para['BT_Los_y'] = BT_y

            BT_x, BT_y = [], []
            for i in range(len(e3)):
                x = float(e3[i].get())
                y = float(e4[i].get())
                BT_x.append(x)
                BT_y.append(y)
                d = ((self.para['MT_x'] - x) ** 2 + (self.para['MT_y']
- y) ** 2) ** 0.5
                n =
(self.para['lamda']*np.random.exponential(self.para['lamda'])+random.
gauss(self.para['ave'], self.para['sigma']))

            self.para['BT_Nlos_x'] = BT_x
            self.para['BT_Nlos_y'] = BT_y
            self.para['BT_NLOS']=BT_NLOS
            self.para['BT']=self.para['BT_LOS']+self.para['BT_NLOS']
            self.Home()
            self.drawbase()
        except:
            messagebox.showwarning("warning!",'input can not be none!')

    def Gauss(self,e1,e2):
        try:
            self.para['ave'] = float(e1.get())
            self.para['sigma'] = float(e2.get())
            #print(self.para)
        except:
            messagebox.showwarning("warning!",'input can not be none!')

    def expoen(self,e1):
        try:
            self.para['lamda'] = float(e1.get())
            #print(self.para)
```

```
        except:
            messagebox.showwarning("warning!",'input can not be none!')

    def MT_pos(self,e1,e2):
        try:
            self.para['MT_x'] = float(e1.get())
            self.para['MT_y'] = float(e2.get())
        except:
            messagebox.showwarning("warning!",'input can not be none!')

    def GetNum_Los(self,e1):
        try:
            if int(e1.get()) < 3:
                messagebox.showwarning('Warning!', "larger than 3!")
            else:
                self.para['Number_Los'] = int(e1.get())

self.para['Number']=self.para['Number_Los']+self.para['Number_Nlos']
        except:
            messagebox.showwarning("warning!",'input can not be none!')

    def GetNum_Nlos(self,e1):
        try:
            self.para['Number_Nlos'] = int(e1.get())
            self.para['Number'] = self.para['Number_Los'] +
self.para['Number_Nlos']
        except:
            messagebox.showwarning("warning!",'input can not be none!')

    def drawbase(self):
        self.frame1.img_bg =
PIL.ImageTk.PhotoImage(PIL.Image.open(self.para['map']).resize((1000,
680)))
        mt_png = self.frame1.create_image((0, 0), anchor='nw')
        self.frame1.itemconfig(mt_png, image=self.frame1.img_bg)

        self.frame1.img_mt =
PIL.ImageTk.PhotoImage(PIL.Image.open('estimate.png').resize((30,
30)))
        mt_png = self.frame1.create_image((self.para['MT_x'],
self.para['MT_y']), anchor='nw')
        self.frame1.itemconfig(mt_png, image=self.frame1.img_mt)  # 内
存保护机制，自动销毁图片内存，需要将图片绑定到 canavas 本身
```

```
        self.frame1.img_bt_los =
PIL.ImageTk.PhotoImage(PIL.Image.open('BT.png').resize((30, 30)))
        for x, y in zip(self.para['BT_Los_x'], self.para['BT_Los_y']):
            bt_png = self.frame1.create_image((x, y), anchor='nw')
            self.frame1.itemconfig(bt_png,
image=self.frame1.img_bt_los)

        self.frame1.img_bt_Nlos =
PIL.ImageTk.PhotoImage(PIL.Image.open('BT_N.png').resize((30, 30)))
        for x, y in zip(self.para['BT_Nlos_x'],
self.para['BT_Nlos_y']):
            bt_png = self.frame1.create_image((x, y), anchor='nw')
            self.frame1.itemconfig(bt_png,
image=self.frame1.img_bt_Nlos)

    def draw_espos(self,es_pos):
        self.frame1.img_es_aml =
PIL.ImageTk.PhotoImage(PIL.Image.open('MT.png').resize((30, 30)))
        mt_png = self.frame1.create_image((es_pos[0], es_pos[1]),
anchor='nw')
        self.frame1.itemconfig(mt_png, image=self.frame1.img_es_aml)
        self.frame1.create_text((es_pos[0] + 10, es_pos[1] - 10),
anchor='nw',
                                text='[' + str(round(es_pos[0],1)) + ',' +
str(round(es_pos[1],1)) + ']', fill='#f20242')

    def Set_Para(self):
        self.para_frame = Frame(self.root)
        self.para_frame.config(bg='#e8c974', height=280, width=300)
        Label(self.para_frame, text='para').place(in_=self.para_frame,
anchor=NW)
        self.para_frame.place(x=400, y=200)
        Number_Los = Entry(self.para_frame)
        Number_Los.place(in_=self.para_frame, anchor=NW, x=60, y=40,
width=85)
        Button(self.para_frame, text='Lnum', command=lambda:
self.GetNum_Los(Number_Los)).place(in_=self.para_frame, anchor=NW,
x=160,y=40,width=50)
        Number_Nlos = Entry(self.para_frame)
        Number_Nlos.place(in_=self.para_frame, anchor=NW, x=60, y=80,
width=85)
        Button(self.para_frame, text='NLnum', command=lambda:
self.GetNum_Nlos(Number_Nlos)).place(in_=self.para_frame,anchor=NW,
x=160, y=80,width=50)
```

```
        Ave = Entry(self.para_frame)
        Ave.place(in_=self.para_frame, anchor=NW, x=60, y=120,
width=40)
        Sigma = Entry(self.para_frame)
        Sigma.place(in_=self.para_frame, anchor=NW, x=105, y=120,
width=40)
        Button(self.para_frame, text='Gauss', command=lambda:
self.Gauss(Ave, Sigma)).place(in_=self.para_frame,
anchor=NW,x=160,y=120, width=50)

        Lamda = Entry(self.para_frame)
        Lamda.place(in_=self.para_frame, anchor=NW, x=60, y=160,
width=85)
        Button(self.para_frame, text='Exp', command=lambda:
self.expoen(Lamda)).place(in_=self.para_frame,anchor=NW, x=160,
y=160,width=50)

        MT_x = Entry(self.para_frame)
        MT_x.place(in_=self.para_frame, anchor=NW, x=60, y=200,
width=40)
        MT_y = Entry(self.para_frame)
        MT_y.place(in_=self.para_frame, anchor=NW, x=105, y=200,
width=40)
        Button(self.para_frame, text='MT pos', command=lambda:
self.MT_pos(MT_x, MT_y)).place(in_=self.para_frame, anchor=NW,x=160,
y=200, width=50)
        Button(self.para_frame, bg='#f5ce42', text='Home',
command=self.Home).place(in_=self.para_frame, anchor=NW, x=120,
y=240, width=80)

    def input(self):
        number_Los = int(self.para['Number_Los'])
        number_Nlos = int(self.para['Number_Nlos'])

        self.frame2_1 = Frame(self.root)
        self.frame2_1.config(bg='white', height=800, width=1000)
        Label(self.frame2_1, text='frame1').place(in_=self.frame2_1,
anchor=NW)
        self.frame2_1.place(x=0, y=0)
        E1_x, E1_y = [], []
        E2_x, E2_y = [], []
        skip = int(400 / number_Los)
```

```
        Label(self.frame2_1, text='LOS').place(in_=self.frame2_1,
anchor=NW,x=250,y=100)
        for i in range(number_Los):
            e_t1 = Entry(self.frame2_1, bg='#f5ce42')
            e_t1.place(in_=self.frame2_1, anchor=NW, x=250, y=160 + i *
skip, width=50)
            e_t2 = Entry(self.frame2_1, bg='#f5ce42')
            e_t2.place(in_=self.frame2_1, anchor=NW, x=350, y=160 + i *
skip, width=50)
            E1_x.append(e_t1)
            E1_y.append(e_t2)

        Label(self.frame2_1, text='NLOS').place(in_=self.frame2_1,
anchor=NW, x=550, y=100)
        for i in range(number_Nlos):
            e_t1 = Entry(self.frame2_1, bg='#f5ce42')
            e_t1.place(in_=self.frame2_1, anchor=NW, x=550, y=160 + i *
skip, width=50)
            e_t2 = Entry(self.frame2_1, bg='#f5ce42')
            e_t2.place(in_=self.frame2_1, anchor=NW, x=650, y=160 + i *
skip, width=50)
            E1_x.append(e_t1)
            E1_y.append(e_t2)

        Button(self.frame2_1, bg='#f5ce42', text='Confirm',
command=lambda: self.get_BT(E1_x, E1_y,E2_x,E2_y)).place(
            in_=self.frame2_1, anchor=NW, x=800, y=400, width=100)
        Button(self.frame2_1, bg='#f5ce42', text='Home',
command=self.Home).place(
            in_=self.frame2_1, anchor=NW, x=800, y=300, width=100)

    def Circle(self):
        self.frame1.delete(tkinter.ALL)
        BT_NLOS,BT_LOS=[],[]
        BT_Los_x, BT_Los_y = [], []
        BT_Nlos_x, BT_Nlos_y = [], []
        rand_list = [x for x in range(self.para['Number'])]
        random.shuffle(rand_list)
        '''
        for i in rand_list[0:-3]:
            x = self.para['MT_x'] + math.sin(2 * math.pi /
self.para['Number'] * i) * min(self.para['MT_x'],980 -
self.para['MT_x']) * 0.75
            y = self.para['MT_y'] + math.cos(2 * math.pi /
```

```
self.para['Number'] * i) * min(self.para['MT_y'],650 -
self.para['MT_y']) * 0.75
            d = ((self.para['MT_x'] - x) ** 2 + (self.para['MT_y'] - y)
** 2) ** 0.5
            tag=random.randint(0,2)
            if tag==0:
                BT_Los_x.append(x)
                BT_Los_y.append(y)
                n = random.gauss(self.para['ave'], self.para['sigma'])
                BT_LOS.append([x, y, d + n])
            else:
                BT_Nlos_x.append(x)
                BT_Nlos_y.append(y)
                n = math.log(d) *
(self.para['lamda']*np.random.exponential(self.para['lamda'])+random.
gauss(self.para['ave'], self.para['sigma']))
                BT_NLOS.append([x, y, d + n])
        '''
        for i in rand_list:
            x = 400 + math.sin(2 * math.pi / self.para['Number'] * i) *
290
            y = 300 + math.cos(2 * math.pi / self.para['Number'] * i) *
290
            d = ((self.para['MT_x'] - x) ** 2 + (self.para['MT_y'] - y)
** 2) ** 0.5
            BT_Los_x.append(x)
            BT_Los_y.append(y)
            n = random.gauss(self.para['ave'], self.para['sigma'])
            BT_LOS.append([x, y, d + n])

        self.para['BT_Los_x'],self.para['BT_Nlos_x']  =
BT_Los_x,BT_Nlos_x
        self.para['BT_Los_y'],self.para['BT_Nlos_y'] =
BT_Los_y,BT_Nlos_y
        self.para['BT_NLOS']=BT_NLOS
        self.para['Number_Los']=len(BT_Los_x)
        self.para['Number_Nlos'] = len(BT_Nlos_x)
        self.para['BT_LOS']=BT_LOS
        self.para['BT'] = self.para['BT_LOS'] + self.para['BT_NLOS']
        self.drawbase()

    def Randd(self):
        self.frame1.delete(tkinter.ALL)
        BT_NLOS,BT_LOS=[],[]
```

```
        BT_Los_x, BT_Los_y= [], []
        BT_Nlos_x, BT_Nlos_y= [], []
        for i in range(self.para['Number']-3):
            x = random.random() * 900+50
            y = random.random() * 600+50
            d = ((self.para['MT_x'] - x) ** 2 + (self.para['MT_y'] - y)
** 2) ** 0.5
            tag = random.randint(0, 2)
            if tag==0:
                BT_Los_x.append(x)
                BT_Los_y.append(y)
                n = random.gauss(self.para['ave'], self.para['sigma'])
                BT_LOS.append([x,y,d+n])
            else:
                BT_Nlos_x.append(x)
                BT_Nlos_y.append(y)
                n =
(self.para['lamda']*np.random.exponential(self.para['lamda'])+random.
gauss(self.para['ave'], self.para['sigma']))
                BT_NLOS.append([x,y,d+n])

        for i in range(3):
            x = random.random() * 900 + 50
            y = random.random() * 600 + 50
            d = ((self.para['MT_x'] - x) ** 2 + (self.para['MT_y'] - y)
** 2) ** 0.5
            BT_Los_x.append(x)
            BT_Los_y.append(y)
            n = random.gauss(self.para['ave'], self.para['sigma'])
            BT_LOS.append([x, y, d + n])

        self.para['BT_LOS']=BT_LOS
        self.para['BT_Los_x'], self.para['BT_Nlos_x'] = BT_Los_x,
BT_Nlos_x
        self.para['BT_Los_y'], self.para['BT_Nlos_y'] = BT_Los_y,
BT_Nlos_y
        self.para['BT_NLOS']=BT_NLOS
        self.para['Number_Los'] = len(BT_Los_x)
        self.para['Number_Nlos'] = len(BT_Nlos_x)
        self.para['BT'] = self.para['BT_LOS'] + self.para['BT_NLOS']
        self.drawbase()

    def Set_XJTU(self):
        self.para['map']='map1.png'
```

```
        self.Home()

    def Set_Cartoon(self):
        self.para['map']='map.png'
        self.Home()

    def Set_World(self):
        self.para['map']='world.png'
        self.Home()

    def Set_White(self):
        self.para['map']='White.png'
        self.Home()

    def Home(self):
        menu = Menu(self.root)
        self.root.config(menu=menu)

        scenario = Menu(menu)
        menu.add_cascade(label='Scenario', menu=scenario)
        scenario.add_command(label='Home', command=self.Home)

        Dynamic = Menu(menu)
        menu.add_cascade(label='Dynamic', menu=Dynamic)
        Dynamic.add_command(label='Dynamic', command=self.Dynamic)

        draw=Menu(menu)
        menu.add_cascade(label='Draw', menu=draw)
        draw.add_command(label='Circle', command=self.Circle)
        draw.add_command(label='Random', command=self.Randd)
        draw.add_command(label='Input', command=self.input)

        method=Menu(menu)
        NLOS = Menu(method)
        NLOS.add_command(label='RWGH', command=self.RWGH)
        NLOS.add_command(label='LLS', command=self.LLS_NLOS)

        LOS = Menu(method)
        ml=Menu(LOS)
        ml.add_command(label='AML',command=self.AML)
        ml.add_command(label='Newton_Gauss',
command=self.newton_gaussian)

        ls = Menu(LOS)
```

```
        ls.add_command(label='LLS_1',command=self.LLS_1)
        ls.add_command(label='LLS_AVE', command=self.LLS_AVER)
        ls.add_command(label='LLS_RS', command=self.LLS_RS)

        sdr = Menu(LOS)
        sdr.add_command(label='R_LS', command=self.R_LS)
        sdr.add_command(label='SR_LS', command=self.SR_LS)

        menu.add_cascade(label='Method', menu=method)
        method.add_cascade(label='LOS',menu=LOS)
        method.add_cascade(label='NLOS',menu=NLOS)

        LOS.add_cascade(label='ML', menu=ml)
        LOS.add_cascade(label='LS', menu=ls)
        LOS.add_cascade(label='SDR', menu=sdr)

        para=Menu(menu)
        menu.add_cascade(label='Para', menu=para)
        para.add_command(label='set_para',command=self.Set_Para)

        map = Menu(menu)
        menu.add_cascade(label='Map', menu=map)
        map.add_command(label='XJTU', command=self.Set_XJTU)
        map.add_command(label='Cartoon', command=self.Set_Cartoon)
        map.add_command(label='World', command=self.Set_World)
        map.add_command(label='White', command=self.Set_White)

        compare=Menu(menu)
        menu.add_cascade(label='Compare', menu=compare)
        compare.add_command(label='COMPARE',command=self.Compare)
        compare.add_command(label='HeatMap', command=self.Heatmap)
        compare.add_command(label='plot3D', command=self.plot3D)

        self.frame1 = Canvas(self.root)
        self.frame1.config(bg='white', height=680, width=1000)
        Label(self.frame1, text='frame1').place(in_=self.frame1,
anchor=NW)
        self.frame1.place(x=0, y=0)
        self.frame1.img_bg =
PIL.ImageTk.PhotoImage(PIL.Image.open(self.para['map']).resize((1000,
680)))
        mt_png = self.frame1.create_image((0, 0), anchor='nw')
        self.frame1.itemconfig(mt_png, image=self.frame1.img_bg)
```

```
    def DrawCar(self):
        interval, t_acc = 1, 0
        pos,v = [400,300],[0,0]

        BT_LOS1 = []
        for x_, y_ in zip(self.para['BT_Los_x'],
self.para['BT_Los_y']):
            d = ((pos[0] - x_) ** 2 + (pos[1] - y_) ** 2) ** 0.5
            n = random.gauss(self.para['ave'], self.para['sigma'])
            BT_LOS1.append([x_, y_, d + n])
        self.para['BT_LOS'] = BT_LOS1
        lls1 = LOS.Program(self.para['BT_LOS'])
        es_pos_init = lls1.LLS_1_E()

        #state estimate paras

F=np.array([[1,0,interval,0],[0,1,0,interval],[0,0,1,0],[0,0,0,1]])#t
ransfer matrix
        q=0.5#energy power
        mean=[0,0,0,0]#noise mean

Q=q*np.array([[interval**3/3,0,interval**2/2,0],[0,interval**3/3,0,in
terval**2/2],[interval**2/2,0,interval,0],[0,interval**2/2,0,interval
]])#covariance
        init_state = [es_pos_init[0], es_pos_init[1], 0, 0]
        P_init = np.array([[5, 0, 0, 0], [0, 5, 0, 0], [0, 0, 2, 0],
[0, 0, 0, 2]])

        while True:
            t_acc=t_acc+interval
            v=[100*math.cos(t_acc),100*math.sin(t_acc)]

            x=pos[0]+random.randint(-1,1)*v[0]*interval
            y=pos[1]+random.randint(-1,1)*v[1]*interval
            if x<10 or x>1000:
                v[0]=-v[0]
                x=pos[0]+v[0]*interval
            if y<10 or y>600:
                v[1]=-v[1]
                y=pos[1]+v[1]*interval
            self.Frame_Dynamic.img_mt =
PIL.ImageTk.PhotoImage(PIL.Image.open('MT.png').resize((30, 30)))
            mt_png = self.Frame_Dynamic.create_image((x, y),
anchor='nw')
```

```
            self.Frame_Dynamic.itemconfig(mt_png,
image=self.Frame_Dynamic.img_mt)
            time.sleep(interval)

self.Frame_Dynamic.create_line(pos[0],pos[1],x,y,fill='blue',width=2)
            pos=[x,y]
            print("pos:",pos)

            BT_LOS = []
            for x_, y_ in zip(self.para['BT_Los_x'],
self.para['BT_Los_y']):
                d = ((pos[0] - x_) ** 2 + (pos[1] - y_) ** 2) ** 0.5
                n = random.gauss(self.para['ave'], self.para['sigma'])
                BT_LOS.append([x_, y_, d + n])
            self.para['BT_LOS'] = BT_LOS

            length = len(self.para['BT_LOS'])-1
            H, Z = [], []
            R = np.zeros((length,length))
            x0=self.para['BT_LOS'][0][0]
            y0=self.para['BT_LOS'][0][1]
            dis0=self.para['BT_LOS'][0][2]
            for i,bt in enumerate(self.para['BT_LOS'][1:]):
                H.append([-2 * (bt[0] - x0), -2 * (bt[1] - y0), 0, 0])
                Z.append(bt[2] ** 2 - dis0 ** 2 - (
                          (bt[0] ** 2 + bt[1] ** 2) - (bt[0] ** 2 +
bt[1] ** 2)))
                R[i][i] = 4*(bt[2]-dis0)**2*self.para['sigma']
            H = np.array(H)
            Z = np.array(Z)

            state_noise = np.random.multivariate_normal(mean, Q)
            estimate_state = F.dot(init_state)
            print("estimate_pos:",estimate_state)
            P_estimate = F.dot(P_init).dot(F.T)+Q
            K =
P_estimate.dot(H.T).dot(np.linalg.inv(H.dot(P_estimate).dot(H.T)+R))
            X_state = estimate_state+ K.dot(Z-H.dot(estimate_state))
            P_state = P_estimate - K.dot(H).dot(P_estimate)

            self.Frame_Dynamic.img_es =
PIL.ImageTk.PhotoImage(PIL.Image.open('estimate.png').resize((30,
30)))
            es_png = self.Frame_Dynamic.create_image((X_state[0],
```

```
X_state[1]), anchor='nw')
            self.Frame_Dynamic.itemconfig(es_png,
image=self.Frame_Dynamic.img_es)
            self.Frame_Dynamic.create_line(init_state[0],
init_state[1], X_state[0], X_state[1], fill='green', width=2)
            init_state = X_state
            P_init = P_state
            print("X_state:",X_state)

    def Dynamic(self):
        self.Frame_Dynamic = Canvas(self.root)
        self.Frame_Dynamic.config(bg='white', height=680, width=1000)
        Label(self.Frame_Dynamic,
text='Dynamic').place(in_=self.Frame_Dynamic, anchor=NW)
        self.Frame_Dynamic.place(x=0, y=0)
        self.Frame_Dynamic.img_bg =
PIL.ImageTk.PhotoImage(PIL.Image.open('map1.png').resize((1000,
680)))
        bg_png = self.Frame_Dynamic.create_image((0, 0), anchor='nw')
        self.Frame_Dynamic.itemconfig(bg_png,
image=self.Frame_Dynamic.img_bg)

        self.Frame_Dynamic.img_bt_los =
PIL.ImageTk.PhotoImage(PIL.Image.open('BT.png').resize((30, 30)))
        for x, y in zip(self.para['BT_Los_x'], self.para['BT_Los_y']):
            bt_png = self.Frame_Dynamic.create_image((x, y),
anchor='nw')
            self.Frame_Dynamic.itemconfig(bt_png,
image=self.Frame_Dynamic.img_bt_los)

        T = threading.Thread(target=self.DrawCar)
        T.start()

    def AML(self):
        self.frame1.delete(tkinter.ALL)
        self.drawbase()
        aml = LOS.Program(self.para['BT_LOS'])
        es_pos = aml.estimate()
        self.draw_espos(es_pos)
        return es_pos

    def LLS_1(self):
        self.frame1.delete(tkinter.ALL)
        self.drawbase()
```

```
        lls1=LOS.Program(self.para['BT_LOS'])
        es_pos = lls1.LLS_1_E()
        self.draw_espos(es_pos)
        return es_pos

    def LLS_AVER(self):
        self.frame1.delete(tkinter.ALL)
        self.drawbase()
        lls_aver=LOS.Program(self.para['BT_LOS'])
        es_pos = lls_aver.LLS_AVE_E()
        self.draw_espos(es_pos)
        return es_pos

    def LLS_RS(self):
        self.frame1.delete(tkinter.ALL)
        self.drawbase()
        lls_rs=LOS.Program(self.para['BT_LOS'])
        es_pos = lls_rs.LLS_RS_E()
        self.draw_espos(es_pos)
        return es_pos

    def R_LS(self):
        self.frame1.delete(tkinter.ALL)
        self.drawbase()
        s_ls=LOS.Program(self.para['BT_LOS'])
        es_pos = s_ls.R_LS_E()
        self.draw_espos(es_pos)
        return es_pos

    def SR_LS(self):
        self.frame1.delete(tkinter.ALL)
        self.drawbase()
        sr_ls=LOS.Program(self.para['BT_LOS'])
        es_pos = sr_ls.SR_LS()
        self.draw_espos(es_pos)
        return es_pos

    def NewTon_Gauss(self):
        N_G=LOS.Program(self.para['BT_LOS'])
        es_pos = N_G.NewTon_Gauss()
        length=len(es_pos)
        self.frame1.img_es_gn =
PIL.ImageTk.PhotoImage(PIL.Image.open('MT.png').resize((30, 30)))
        for i in range(length-1):
```

```
            mt_png = self.frame1.create_image((es_pos[i][0],
es_pos[i][1]), anchor='nw')
            self.frame1.itemconfig(mt_png, image=self.frame1.img_es_gn)
            time.sleep(1)
            self.frame1.delete(mt_png)
        self.draw_espos([es_pos[length-1][0], es_pos[length-1][1]])
        return [es_pos[length-1][0], es_pos[length-1][1]]

    def newton_gaussian(self):
        self.frame1.delete(tkinter.ALL)
        self.drawbase()
        T=threading.Thread(target=self.NewTon_Gauss)
        T.start()

    def lsSteps(self):
        self.frame1.delete(tkinter.ALL)
        self.drawbase()
        bt = self.para['BT_LOS']
        for i in range(len(bt)):
            bt[i][2] = bt[i][2] + self.para['t0']
        lsteps = LOS.Program_t(bt)
        es_pos = lsteps.LS_steps()
        self.draw_espos(es_pos)
        return es_pos

#-----------------------------------------------------------------------
-------NLOS

    def LLS_NLOS(self):
        self.frame1.delete(tkinter.ALL)
        self.drawbase()
        lls_nlos = NLOS.Program(self.para['BT'])
        es_pos = lls_nlos.LLS_NLOS()
        self.draw_espos(es_pos)
        return es_pos

    def RWGH(self):
        self.frame1.delete(tkinter.ALL)
        self.drawbase()
        rwgh=NLOS.Program(self.para['BT'])
        NLOS_pos,es_pos = rwgh.RSWG()
        for pos in NLOS_pos:
            self.frame1.create_text((pos[0] + 10, pos[1] - 10),
anchor='nw',text='LOS',fill='#f20242')
```

```
        self.draw_espos(es_pos)
        return es_pos

#-------------------------------------------------------------------
--------------------------------Compare
    def Scenario_para(self):
        self.frame3_2 = Frame(self.frame3_1)
        self.frame3_2.config(bg='#bfd4f5', height=700, width=300)
        self.frame3_2.place(in_=self.frame3_1, x=20, y=50)
        Label(self.frame3_2, text='场景参数').place(in_=self.frame3_2,
anchor=NW, x=0, y=0)

        columns = ("base", "x pos", "y pos")
        headers = ("基站", "X 坐标", "Y 坐标")
        info = [('MT', round(self.para['MT_x'], 1),
round(self.para['MT_y'], 1)), ('LOS', '', '')]
        for i in range(self.para['Number_Los']):
            info.append(('BT' + str(i), round(self.para['BT_Los_x'][i],
1), round(self.para['BT_Los_y'][i], 1)))
        info.append(('NLOS', '', ''))

        for i in range(self.para['Number_Nlos']):
            info.append(('BT' + str(self.para['Number_Los'] + i),
round(self.para['BT_Nlos_x'][i], 1),
                        round(self.para['BT_Nlos_y'][i], 1)))

        tv = ttk.Treeview(self.frame3_2, show="headings",
columns=columns)
        for (column, header) in zip(columns, headers):
            tv.column(column, width=80, anchor="nw")
            tv.heading(column, text=header, anchor="nw")

        for i, data in enumerate(info):
            tv.insert('', i, values=data)

        tv.place(in_=self.frame3_2, anchor=NW, x=25, y=60)

        head=('误差','μ/λ','sigma')
        col=('noise','ave','sigma')
        tv1 = ttk.Treeview(self.frame3_2, show="headings",
columns=col)
        for (column, header) in zip(col, head):
            tv1.column(column, width=80, anchor="nw")
            tv1.heading(column, text=header, anchor="nw")
```

```
tv1.insert('',1,values=('Gauss',self.para['ave'],self.para['sigma']))
        tv1.insert('', 2, values=('Expoen', self.para['lamda'], ''))
        tv1.place(in_=self.frame3_2, anchor=NW, x=25, y=300)

    def Compare(self):

        self.frame3_1 = Frame(self.root)
        self.frame3_1.config(bg='#cfcdcc', height=680, width=1000)
        Label(self.frame3_1, text='frame1').place(in_=self.frame3_1,
anchor=NW)
        self.frame3_1.place(x=0, y=0)
        Label(self.frame3_1,text='算法性能评估
').place(in_=self.frame3_1,anchor=NW)
        self.Scenario_para()


self.frame3_1.MT_pos=PIL.ImageTk.PhotoImage(PIL.Image.open('MT_pos.pn
g').resize((250, 250)))
        picture=Label(self.frame3_1, image =
self.frame3_1.MT_pos).place(in_=self.frame3_1, x=350, y=50,
anchor=NW)

        self.frame3_1.Noise =
PIL.ImageTk.PhotoImage(PIL.Image.open('Noise.png').resize((250,
250)))
        picture = Label(self.frame3_1,
image=self.frame3_1.Noise).place(in_=self.frame3_1, x=650, y=50,
anchor=NW)

        self.frame3_1.Heat =
PIL.ImageTk.PhotoImage(PIL.Image.open('HeatMap.png').resize((250,
250)))
        picture = Label(self.frame3_1,
image=self.frame3_1.Heat).place(in_=self.frame3_1, x=350, y=350,
anchor=NW)

        self.frame3_1.Three_D =
PIL.ImageTk.PhotoImage(PIL.Image.open('3D.png').resize((250, 250)))
        picture = Label(self.frame3_1,
image=self.frame3_1.Three_D).place(in_=self.frame3_1, x=650, y=350,
anchor=NW)

        #ana=analysis.Analysis(self.para)
```

```
        #ana.Change_MT_Pos()
        #ana.Change_Noise()

    def Heatmap(self):
        ana=analysis.Analysis(self.para)
        ana.Change_Noise()

    def plot3D(self):
        ana = analysis.Analysis(self.para)
        ana.losNlos()

def main():
    root = Tk()
    s=simulation(root)
    mainloop()
if __name__ == "__main__":
    main()
```

附录 C.2　LOS 算法

```
import matplotlib.pyplot as plt
from matplotlib import cm
from sympy import *
import cvxopt as cvx
import cvxpy as cp
import numpy as np
import random
import time
import math
# c=299552816


def plot_cicle(centers, rads, mt, bt_x, bt_y):
    fig = plt.figure()
    ax = fig.add_subplot(111)
    for center, rad in zip(centers, rads):
        cir = plt.Circle((center[0], center[1]), radius=rad,
color='y', fill=False)
        ax.add_patch(cir)

    ax.plot(bt_x, bt_y, 'ro')
    ax.plot(mt[0], mt[1], 'ro')
    plt.axis('scaled')
    plt.axis('equal')
    plt.show()


class Program:

    def __init__(self, bt):
        self.BT = bt
        self.BT_x = [b[0] for b in self.BT]
        self.BT_y = [b[1] for b in self.BT]
        self.dis_ = [b[2] for b in self.BT]
        self.number = len(bt)

# ---------------------------------------------For ML algorithm

    def RSR(self, dis, pos, BT_x,BT_y):
        sum = 0
        for d, p, bt_x,bt_y in zip(dis, pos, BT_x,BT_y):
            elem = (d - ((pos[0] - bt_x) ** 2 + (pos[1] - bt_y) ** 2)
** 0.5) ** 2
```

```
        return sum

    def cal(self,tag, paras, cords, c_p=0):
        sum, g_i = 0, []
        if tag == 0:
            for para, cord in zip(paras, cords):
                if para:
                    g_i.append((c_p - cord) / (2 * para ** 2))
                else:
                    g_i.append(1)
            return g_i

        if tag == 1:
            for para, cord in zip(paras, cords):
                sum = sum + para * cord
            return sum

    def cal1(self,s, P, X, Y, D):
        sum = 0
        for p, x, y, d in zip(P, X, Y, D):
            sum = sum + p * (s + x ** 2 + y ** 2 - d ** 2)
        return sum

    def cal2(self,s, P, X, Y, D):
        sum = 0
        if isinstance(s,complex):
            s=s.real
        else:
            s=abs(s)
        for p, x, y, d in zip(P, X, Y, D):
            sum = sum + p * (s + x ** 2 + y ** 2 - d ** 2)
        return sum

    def init_value(self):
        # deal with BT in a line
        if self.BT_y[0]==self.BT_y[1] and self.BT_y[0]==self.BT_y[2]:
            x=((self.dis_[1]**2-self.dis_[0]**2)-(self.BT_x[1]**2-
self.BT_x[0]**2))/(2*(self.BT_x[0]-self.BT_x[1]))
            y=(self.dis_[0]**2-(x-self.BT_x[0])**2)**0.5+self.BT_y[0]

        elif self.BT_x[0]==self.BT_x[1] and
self.BT_x[0]==self.BT_x[2]:
            y = ((self.dis_[1] ** 2 - self.dis_[0] ** 2) -
(self.BT_y[1] ** 2 - self.BT_y[0] ** 2)) / (2 * (self.BT_y[0] -
```

```
self.BT_y[1]))
            x = (self.dis_[0] ** 2 - (y - self.BT_y[0]) ** 2) ** 0.5 +
self.BT_x[0]

        else:
            a,b=2*(self.BT_x[1]-self.BT_x[0]),2*(self.BT_y[1]-
self.BT_y[0])
            c=(self.dis_[0]**2-self.dis_[1]**2)-((self.BT_x[0]**2-
self.BT_x[1]**2)+(self.BT_y[0]**2-self.BT_y[1]**2))
            d,e=2*(self.BT_x[2]-self.BT_x[0]),2*(self.BT_y[2]-
self.BT_y[0])
            f=(self.dis_[0] ** 2 - self.dis_[2] ** 2) -
((self.BT_x[0]**2 - self.BT_x[2]**2) + (self.BT_y[0]**2 -
self.BT_y[2]**2))
            y = (f - d / a * c) / (e - d / a * b)
            x=(c-b*y)/a
        return x,y

    def estimate_one(self,x_ini,y_ini):

self.g_i,self.h_i=self.cal(0,self.dis_,self.BT_x,c_p=x_ini),self.cal(
0,self.dis_,self.BT_y,c_p=y_ini)

a,b,c,d=self.cal(1,self.g_i,self.BT_x),self.cal(1,self.g_i,self.BT_y)
,self.cal(1,self.h_i,self.BT_x),self.cal(1,self.h_i,self.BT_y)
        A=2*np.array([[a,b],[c,d]])
        s=Symbol('s')
        e=self.cal1(s,self.g_i,self.BT_x,self.BT_y,self.dis_)
        f = self.cal1(s, self.h_i, self.BT_x, self.BT_y, self.dis_)
        B=np.array([e,f]).T
        try:
            A_=np.linalg.inv((A.T).dot(A)).dot(A.T)
        except:
            print('矩阵不可逆')

s_=solve((A_[0][0]*e+A_[0][1]*f)**2+(A_[1][0]*e+A_[1][1]*f)**2-s,s)
        #print(s_)
        e_1=self.cal2(s_[0],self.g_i,self.BT_x,self.BT_y,self.dis_)
        f_1=self.cal2(s_[0],self.h_i,self.BT_x,self.BT_y,self.dis_)
        B_1= np.array([e_1, f_1]).T

        e_2 = self.cal2(s_[1], self.g_i, self.BT_x, self.BT_y,
self.dis_)
        f_2 = self.cal2(s_[1], self.h_i, self.BT_x, self.BT_y,
```

```
self.dis_)
        B_2 = np.array([e_2, f_2]).T
        try:
            position1=np.linalg.inv(A).dot(B_1)
            position2 = np.linalg.inv(A).dot(B_2)
            t1=self.RSR(self.dis_,position1,self.BT_x,self.BT_y)
            t2=self.RSR(self.dis_,position2,self.BT_x,self.BT_y)
            return position1 if t1<t2 else position2
        except:
            print("矩阵不可逆")

    def estimate(self):
        x_ini,y_ini=self.init_value()
        pos_,J_v = [],[]
        location = [x_ini, y_ini]
        for i in range(5):
            sum=0
            for d,cord_x,cord_y in zip(self.dis_,self.BT_x,self.BT_y):
                sum=sum+(d-((location[0]-cord_x)**2+(location[1]-
cord_y)**2)**0.5)**2
            J_v.append(sum)
            location = self.estimate_one(float(location[0]),
float(location[1]))
            pos_.append(list(location))
        id = J_v.index(min(J_v))
        #print("estimate_location:",pos_[id])
        pos_[id]=[round(pos_[id][0],1),round(pos_[id][1],1)]
        return pos_[id]

#-----------------------------------------------------------------------
-----for LS

    def LLS_1_E(self):
        A,p=[],[]
        for bt_x,bt_y,dis in
zip(self.BT_x[1:],self.BT_y[1:],self.dis_[1:]):
            A.append([bt_x-self.BT_x[0],bt_y-self.BT_y[0]])
            p.append([self.dis_[0]**2-dis**2-
((self.BT_x[0]**2+self.BT_y[0]**2)-(bt_x**2+bt_y**2))])
        A3=np.array(A)
        p3=np.array(p)
        pos=0.5*(np.linalg.inv((A3.T).dot(A3))).dot(A3.T).dot(p3)
        position=[round(pos[0][0],1),round(pos[1][0],1)]
        return position
```

```
    def LLS_AVE_E(self):
        A,p=[],[]
        d_ave=sum(list(map(lambda d: d**2,self.dis_)))/len(self.dis_)
        k_ave=sum(list(map(lambda x,y:
x**2+y**2,self.BT_x,self.BT_y)))/len(self.BT_x)
        x_ave=sum(self.BT_x)/len(self.BT_x)
        y_ave = sum(self.BT_y) / len(self.BT_y)
        for bt_x,bt_y,dis in zip(self.BT_x,self.BT_y,self.dis_):
            A.append([bt_x-x_ave,bt_y-y_ave])
            p.append([d_ave-dis**2+(bt_x**2+bt_y**2)-k_ave])
        A3=np.array(A)
        p3=np.array(p)
        pos=0.5*(np.linalg.inv((A3.T).dot(A3))).dot(A3.T).dot(p3)
        position=[round(pos[0][0],1),round(pos[1][0],1)]
        return position

    def LLS_RS_E(self):
        A,p=[],[]
        i_d=self.dis_.index(min(self.dis_))
        d_min,x_min,y_min=self.dis_[i_d],self.BT_x[i_d],self.BT_y[i_d]
        BT_X=self.BT_x[0:i_d]+self.BT_x[i_d+1:]
        BT_Y = self.BT_y[0:i_d] + self.BT_y[i_d + 1:]
        dis_=self.dis_[0:i_d]+self.dis_[i_d+1:]
        for bt_x,bt_y,dis in zip(BT_X,BT_Y,dis_):
            A.append([bt_x-x_min,bt_y-y_min])
            p.append([d_min**2-dis**2-((x_min**2+y_min**2)-
(bt_x**2+bt_y**2))])
        A3=np.array(A)
        p3=np.array(p)
        pos=0.5*(np.linalg.inv((A3.T).dot(A3))).dot(A3.T).dot(p3)
        position=[round(pos[0][0],1),round(pos[1][0],1)]
        return position

    def CWLS(self):
        B_ = self.dis_
        B = np.diag(B_)

    def WLS(self):
        B_ = [1/(d ** 2) for d in self.dis_]
        B = np.diag(B_)

#-------------------------------------------------------------------
----------------SDR
```

```
    def R_LS_E(self):
        num=len(self.dis_)
        X, G = cp.Variable((3, 3),PSD=true), cp.Variable((num + 1, num
+ 1),PSD=true)# PSD=true means semidefinite
        expr=0
        constraints = [G[num, num] == 1, X[2, 2] == 1]
        for i in range(num):
            C_i=[[1,0,-self.BT_x[i]],[0,1,-self.BT_y[i]],[-
self.BT_x[i],-self.BT_y[i],(self.BT_x[i]**2+self.BT_y[i]**2)]]
            expr=expr+G[i,i]-2*self.dis_[i]*G[num,i]+self.dis_[i]**2
            constraints += [G[i, i] == cp.trace(C_i@X)]
        obj = cp.Minimize(expr)
        prob = cp.Problem(obj,constraints)
        prob.solve()
        position = [round(X.value[0,2],1), round(X.value[1,2], 1)]
        #print(position)
        return position

    def kesi(self,lamda,A,D,b,f):
        kesi_lamda = 0
        if np.linalg.det((A.T).dot(A) + lamda*D) == 0:
            kesi_lamda = np.inf
        else:
            y_lamda = np.linalg.solve(A.T.dot(A) + lamda * D,
A.T.dot(b)  - lamda * f)
            kesi_lamda = y_lamda.T.dot(D).dot(y_lamda) + 2 *
(f.T) .dot(y_lamda)
        return y_lamda,kesi_lamda

    def SR_LS(self):
        A = np.array([[-2 * x, -2 * y, 1] for x, y in zip(self.BT_x,
self.BT_y)])
        D = np.array([[1, 0, 0], [0, 1, 0], [0, 0, 0]])
        b = np.array([d ** 2 - (x ** 2 + y ** 2) for x, y, d in
zip(self.BT_x, self.BT_y, self.dis_)])
        f = np.array([0,0,-0.5])
        A_T_A = (A.T).dot(A)
        A_T_A_inv = np.linalg.inv(A_T_A)
        eig = list(np.linalg.eig(A_T_A_inv)[0])
        A_T_A_eig = np.diag(eig)
        #print("A_T_A_eig:", A_T_A_eig)
        A_T_A_mat = np.linalg.eig(A_T_A_inv)[1].T
        #print("A_T_A_mat:", A_T_A_mat)
        square_eig = [x ** 0.5 for x in eig]
```

```
        # print("square_eig",square_eig)
        square_eig_mat = np.diag(square_eig)
        # print("square_mat", square_eig_mat)
        ATA_sq = A_T_A_mat.dot(square_eig_mat).dot(A_T_A_mat)
        lamda_array =
list(np.linalg.eig(ATA_sq.dot(D).dot(ATA_sq))[0])
        #print("lamda_array:", lamda_array)
        lamda1 = max(lamda_array)
        if lamda1 ==0:
            lamda1 = 0.1
        #print("lamda1:", lamda1)
        left = -1 / lamda1
        left_kesi = np.inf
        right, err = 10, 0.1
        _, right_kesi = self.kesi(right, A, D, b, f)
        #print("right_kesi:", kesi)
        while right_kesi>=0:
            left = right
            left_kesi = right_kesi
            right=10*right
            _, right_kesi = self.kesi(right, A, D, b, f)
        mid = (left+right)/2
        #print("right_kesi:", right_kesi)
        #print("left_kesi:",left_kesi)
        #print("origin_left:",left)
        while right-left > 0.01:
            if np.isfinite(left_kesi):
                p_rf = left + left_kesi * (right - left) / (left_kesi -
right_kesi)
                # interval midpoint
                p_mid = (left + right) / 2
                # weighted average of the midpoint and regula falsi
point
                bsfac = 0.95
                mid = bsfac * p_rf + (1 - bsfac) * p_mid#试位法
            else:
                mid = (right+left)/2
            _, kesi = self.kesi(mid, A, D, b, f)
            if kesi < -err:
                right = mid
                mid = (left + right) / 2
            elif kesi > err:
                left = mid
                mid = (left + right) / 2
```

```
            else:
                break
        y,kesi_lamda = self.kesi(mid,A,D,b,f)
        pos = [round(abs(y[0]),1),round(abs(y[1]),1)]
        #print("pos:",pos)
        return pos

#-----------------------------------------------------NewTon-Gaussian

    def Grad(self,f,x1,x2,x_i,y_i):
        f1=diff(f,x1)
        f2=diff(f,x2)

grad=np.array([[f1.subs([(x1,x_i),(x2,y_i)])],[f2.subs([(x1,x_i),(x2,
y_i)])]])
        return grad

    def Hess(self,f,x1,x2,x_i,y_i):
        f1=diff(f,x1)
        f2=diff(f,x2)

f11,f12,f21,f22=diff(f1,x1),diff(f1,x2),diff(f2,x1),diff(f2,x2)

hess=np.array([[f11.subs([(x1,x_i),(x2,y_i)]),f12.subs([(x1,x_i),(x2,
y_i)])],

[f21.subs([(x1,x_i),(x2,y_i)]),f22.subs([(x1,x_i),(x2,y_i)])]])
        hess = np.array(hess, dtype='float')
        return hess

    def NewTon_Gauss(self):
        err,count=0.01,0
        X=self.init_value()
        x_k,x_k1=np.array([[X[0]],[X[1]]]),np.array([[0],[0]])
        x_p,y_p=symbols('x,y')
        target_fun=sum(list(map(lambda d,bt_x,bt_y:(d-((x_p-
bt_x)**2+(y_p-bt_y)**2)**0.5)**2,self.dis_,self.BT_x,self.BT_y)))
        pos=[]
        while True:
            #print(x_k)
            pos.append([round(x_k[0][0],1),round(x_k[1][0],1)])
            x_err=x_k-x_k1
            if x_err[0][0]**2+x_err[1][0]**2<=err**2:
                break
```

```
            grad = self.Grad(target_fun, x_p, y_p, x_k[0][0],
x_k[1][0])
            hess = self.Hess(target_fun, x_p, y_p, x_k[0][0],
x_k[1][0])
            if grad[0][0]**2+grad[1][0]**2<err*2:
                break
            x_k1=x_k
            x_k=x_k-(np.linalg.inv(hess)).dot(grad)
            count=count+1
            if count>10:
                break
        return pos[-1]

    def SAA(self):
        X_target = [1000,600]
        E_1 = sum([self.dis_[i]-((self.BT_x[i]-
X_target[0])**2+(self.BT_y[i]-X_target[1])**2)**0.5 for i in
range(len(self.BT_x))])
        X = [(0.9+random.random()/10)*X_target[0],
(0.9+random.random(0,0.1)/10)*X_target[1]]
        E_2 = sum([self.dis_[i]-((self.BT_x[i]-X[0])**2+(self.BT_y[i]-
X[1])**2)**0.5 for i in range(len(self.BT_x))])
        if E_2<E_1:
            X_target = X
        else:
            math.exp(-(E_2-E_1)/1)
#-------------------------------------------------------------------
-------------

    def CRLB_LOS(self,MT_x,MT_y,sigma):
        a, b, c = 0, 0, 0
        for k in range(len(self.BT_x)):
            deno = sigma ** 2 * ((MT_x - self.BT_x[k]) ** 2 + (MT_y -
self.BT_y[k]) ** 2)
            a = a + (MT_x - self.BT_x[k]) ** 2 / deno
            b = b + ((MT_x - self.BT_x[k]) * (MT_y - self.BT_y[k])) /
deno
            c = c + (MT_y - self.BT_y[k]) ** 2 / deno
        try:
            Fisher = np.array([[a, b], [b, c]])
            crlb = 1 / (Fisher)
            #print(crlb)
        except:
            print("矩阵不可逆")
```

```
        # datas=list(np.array(datas)[10:40,10:40])
        return crlb[0,0],crlb[1,1]
class Program_t:

    def __init__(self,bt):
        self.BT = bt
        self.BT_x = [b[0] for b in self.BT]
        self.BT_y = [b[1] for b in self.BT]
        self.T=[b[2] for b in self.BT]
        self.number = len(bt)

    def lsT(self):
        A = []
        b = []
        x1, y1, d1 = self.BT_x[0], self.BT_y[0], self.T[0]
        for i in range(1,self.number):
            A.append([2*(self.BT_x[i]-x1), 2*(self.BT_y[i]-y1), 2*(d1-
self.T[i])])
            b.append(self.BT_x[i]**2-x1**2+self.BT_y[i]**2-y1**2+d1**2-
self.T[i]**2)
        A = np.array(A)
        b = np.array(b)
        pos = (np.linalg.inv((A.T).dot(A))).dot(A.T).dot(b)
        position = [round(pos[0], 1), round(pos[1], 1)]
        print(pos)
        return position

    def LS_steps(self):
        num = len(self.T)
        up_control = 2*max(self.T)**2
        Q = cp.Variable((num,num))
        tao = cp.Variable((num,1))
        y_ = cp.Variable((2,1))
        y_s = cp.Variable((1,1))
        yita = 0.00005*sum(self.T) / num
        G = np.eye(num)-np.ones((num,num))
        t = np.array([self.T]).T
        expr1 = cp.trace((cp.transpose(G)) @ G @ (Q- cp.multiply(2,t @
(cp.transpose(tao)))+t @ (cp.transpose(t))))
        expr2 = yita*cp.sum(Q)
        expr = expr1+ expr2
        Q_ = cp.bmat([[Q,tao],[cp.transpose(tao),[[1]]]])
        Y = cp.bmat([[np.eye(2),y_],[cp.transpose(y_),y_s]])
        constraints = [Q_ >> 0, Y >> 0, cp.max(Q)<=up_control]
```

```
        for i in range(num):
            X = np.array([self.BT_x[i],self.BT_y[i],-1]).T
            constraints += [Q[i, i] == cp.transpose(X) @ Y @ X]
            for j in range(i+1,num):
                X_j = np.array([self.BT_x[j], self.BT_y[j],-1]).T
                constraints += [Q[i, j] >= cp.abs(cp.transpose(X) @ Y @
X_j)]
        obj = cp.Minimize(expr)
        prob = cp.Problem(obj, constraints)
        prob.solve()
        position = y_.value
        #print(expr1.value)
        #print(expr2.value)
        #print(prob.value)
        #print(prob.status)
        print(position)
        return position


#-----------------------------------------------------------------
----画圆函数

    def plot_cicle(centers, rads, MT, BT_x, BT_y):
        fig = plt.figure()
        ax = fig.add_subplot(111)
        for center, rad in zip(centers, rads):
            cir = plt.Circle((center[0], center[1]), radius=rad,
color='y', fill=False)
            ax.add_patch(cir)

        ax.plot(BT_x, BT_y, 'ro')
        ax.plot(MT[0], MT[1], 'ro')
        plt.axis('scaled')
        plt.axis('equal')
        plt.show()

def exe_time():
    BT_x, BT_y, Dis, MPE, dis_ = [], [], [], [], []
    ave, sigma = 0, 2
    MT = [random.random()*600+200, random.random()*400+100]
    bt = []
    number = random.randint(3,8)
    for i in range(number):
        x = random.random() * 800
```

```
    BT_x.append(x)
    y = random.random() * 500
    BT_y.append(y)
    d = ((MT[0] - BT_x[i]) ** 2 + (MT[1] - BT_y[i]) ** 2) ** 0.5
    Dis.append(d)
    n = random.gauss(ave, sigma)
    dis_.append(d + n)
    bt.append((x, y, d + n))

#los = Program_t(bt)

#start = time.time()
#los.LS_steps()
#end = time.time()
#lsteps_exe = end - start
los = Program(bt)

start = time.time()
los.estimate()
end = time.time()
aml_exe = end-start

start = time.time()
los.LLS_1_E()
end = time.time()
lls_1_exe = end - start

start = time.time()
los.NewTon_Gauss()
end = time.time()
N_G_exe = end - start

start = time.time()
los.SR_LS()
end = time.time()
sr_ls_exe = end - start

start = time.time()
los.R_LS_E()
end = time.time()
r_ls_exe = end - start

print("aml:",aml_exe)
print("lls:",lls_1_exe)
```

```
    print("NG:",N_G_exe)
    print("sr_ls:",sr_ls_exe)
    print("r_ls:",r_ls_exe)
    #return aml_exe,lls_1_exe,N_G_exe,sr_ls_exe,r_ls_exe
    bias = [aml_exe,lls_1_exe,N_G_exe,sr_ls_exe,r_ls_exe]
    name=['AML','LLS','NG','SR-LS','R-LS']
    plt.bar(range(len(bias)), bias, tick_label=name)
    plt.xlabel('Algorithm')
    plt.ylabel('Execution Time/s')
    plt.title('Execution Time')
    plt.show()

def main_():
    aml, lls_1, N_G, sr_ls, r_ls=[],[],[],[],[]
    for i in range(30):
        aml_exe,lls_1_exe,N_G_exe,sr_ls_exe,r_ls_exe = exe_time()
        aml.append(aml_exe)
        lls_1.append(lls_1_exe)
        N_G.append(N_G_exe)
        sr_ls.append(sr_ls_exe)
        r_ls.append(r_ls_exe)
        print(i)
    aml.pop(aml.index(max(aml)))
    lls_1.pop(lls_1.index(max(lls_1)))
    N_G.pop(N_G.index(max(N_G)))
    sr_ls.pop(sr_ls.index(max(sr_ls)))
    r_ls.pop(r_ls.index(max(r_ls)))
    bias = [sum(aml) / 29, sum(lls_1) / 29, sum(N_G) / 29, sum(sr_ls)
/ 29, sum(r_ls)/29]
    name = ['AML', 'LLS1', 'N_G', 'SR_LS', 'R_LS']
    plt.bar(range(len(bias)), bias, tick_label=name)
    plt.xlabel('Algorihm')
    plt.ylabel('Exe_time/s')
    plt.title('Exe_time')
    plt.show()
    #sdr.test()

def CRLB_LOS():
    #BT_x=[250,250,750,750]
    #BT_y=[250,750,250,750]
    BT_x=[int(500+250*math.cos(i*math.pi/4)) for i in range(8)]
    BT_y=[int(500+250*math.sin(i*math.pi/4)) for i in range(8)]
    print(BT_x,BT_y)
    MT_x,MT_y=0,0
```

```
    sigma = 5
    datas1, datas2 = [], []
    for i in range(100):
        print(i)
        data1, data2 = [], []
        MT_x = 10 * i
        for j in range(100):
            MT_y = 10 * j
            a, b, c = 0, 0, 0
            for k in range(len(BT_x)):
                deno = sigma ** 2 * ((MT_x - BT_x[k]) ** 2 + (MT_y -
BT_y[k]) ** 2)
                if deno:
                    a = a + (MT_x - BT_x[k]) ** 2 / deno
                    b = b + ((MT_x - BT_x[k]) * (MT_y - BT_y[k])) / deno
                    c = c + (MT_y - BT_y[k]) ** 2 / deno
                else:
                    a,b,c=0.1,0.1,0.1
                    break
            try:
                Fisher = np.array([[a, b], [b, c]])
                crlb = 1 / (Fisher)
                #crlb=np.linalg.inv(Fisher)
                # print(crlb)
            except:
                print("矩阵不可逆")
            # datas=list(np.array(datas)[10:40,10:40])
            #return crlb[0, 0], crlb[1, 1]
            #noise1 = (crlb[0, 0]+crlb[1, 1])**0.5
            noise1 = (crlb[0, 0] + crlb[1, 1]) ** 0.5/sigma
            #print(noise1)
            data1.append(float(noise1))
        datas1.append(data1)
        #print(datas1)

    x_tick = [100 * i for i in range(10)]
    y_tick = [100 * i for i in range(10)]
    ax1 = plt.subplot()
    for l in range(len(BT_x)):
        #plt.scatter(BT_x[l], BT_y[l],s=y,  # 设置散点x大小color='r',  #
设置散点颜色(也可以是一个列表，控制每个点的颜色)marker="o",  # 设置散点形状
linewidths=3,  # 设置边框的宽度edgecolors='r',  # 设置边框的颜色alpha=0.9,
# 设置透明度 )
        #plt.Rectangle([50, 50], 20,20,color='r',fill=True)
```

```
        circle = plt.Circle((BT_x[l]/10, BT_y[l]/10), 1, color='y',
fill=True)
        plt.gcf().gca().add_artist(circle)
    #    ax1.add_patch(rec)
    im1 = ax1.imshow(datas1, cmap=cm.binary)
    #plt.text(50, 50, 'FT', color='red',fontsize=5)
    #print(im1)
    ax1.set_xticks(np.arange(5, 100, 10), labels=x_tick)
    ax1.set_yticks(np.arange(5, 100, 10), labels=y_tick)

    plt.title('GDOP Heat Map')
    plt.xlabel('X/m')
    plt.ylabel('Y/m')
    cbar = plt.colorbar(im1)
    cbar.set_label('RMSE/m')
    plt.show()
def main():
    bt_x = [0,0,800,800]
    bt_y = [0,600,0,600]
    bt = []
    stepbias,lsbias,count =0,0,3
    for i in range(count):
        d0 = random.random() * 50
        mt = [random.random() *800, random.random()*600]
        for x, y in zip(bt_x,bt_y):
            d = ((x - mt[0])**2+(y-
mt[1])**2)**0.5+d0+random.gauss(0,15)
            bt.append([x, y, d])
        item = Program_t(bt)
        #print(d0)
        lsstep_pos=item.LS_steps()
        print("lsstep_pos:",lsstep_pos)
        lspos=item.lsT()
        stepbias+=((lsstep_pos[0][0] - mt[0]) ** 2 + (lsstep_pos[1][0]
- mt[1]) ** 2) ** 0.5
        lsbias+=((lspos[0] - mt[0]) ** 2 + (lspos[1] - mt[1]) ** 2) **
0.5
    bias = [stepbias/count,lsbias/count]
    name = ['LS-Step','LLS']
    plt.bar(range(len(bias)), bias, tick_label=name)
    plt.xlabel('Algorihm')
    plt.ylabel('Average Bias/m²')
    plt.title('SDP And LLS')
    plt.show()
```

```
if __name__=="__main__":
    #CRLB_LOS()
    exe_time()
```

附录 C.3　NLOS 算法

```
from sympy import *
import numpy as np
import random
import math
import LOS

class Program:

    def __init__(self,bt):
        self.BT=bt
        self.BT_x = [b[0] for b in self.BT ]
        self.BT_y = [b[1] for b in self.BT]
        self.dis_ = [b[2] for b in self.BT]

    def LLS_NLOS(self):
        dis_, BT_x, BT_y = [b[2] for b in self.BT], [b[0] for b in
self.BT], [b[1] for b in self.BT]
        A, p = [], []
        for bt_x, bt_y, dis in zip(BT_x[1:], BT_y[1:], dis_[1:]):
            A.append([bt_x - BT_x[0], bt_y - BT_y[0]])
            p.append([dis_[0] ** 2 - dis ** 2 - ((BT_x[0] ** 2 +
BT_y[0] ** 2) - (bt_x ** 2 + bt_y ** 2))])
        A3 = np.array(A)
        p3 = np.array(p)
        pos = 0.5 * (np.linalg.inv((A3.T).dot(A3))).dot(A3.T).dot(p3)
        position = [round(pos[0][0], 1), round(pos[1][0], 1)]
        return position

    def target_fun(self,x_p,y_p,BT):
        dis, BT_x, BT_y = [b[2] for b in BT], [b[0] for b in BT],
[b[1] for b in BT]
        target_fun=sum(list(map(lambda d,bt_x,bt_y:(d-((x_p-
bt_x)**2+(y_p-bt_y)**2)**0.5)**2,dis,BT_x,BT_y)))
        return target_fun

    def lls(self,BT):
        dis_, BT_x, BT_y = [b[2] for b in BT], [b[0] for b in BT],
[b[1] for b in BT]
        A, p = [], []
        for bt_x, bt_y, dis in zip(BT_x[1:], BT_y[1:], dis_[1:]):
            A.append([bt_x - BT_x[0], bt_y - BT_y[0]])
            p.append([dis_[0] ** 2 - dis ** 2 - ((BT_x[0] ** 2 +
BT_y[0] ** 2) - (bt_x ** 2 + bt_y ** 2))])
        A3 = np.array(A)
```

```
        p3 = np.array(p)
        pos = 0.5 * (np.linalg.inv((A3.T).dot(A3))).dot(A3.T).dot(p3)
        position = [round(pos[0][0], 1), round(pos[1][0], 1)]
        return position

    def RSWG(self):
        target_BT=self.BT
        length = len(self.BT)
        x_min=self.lls(target_BT)
        Res=self.target_fun(x_min[0],x_min[1],target_BT)/length
        for i in range(0,length-3):
            temp_BT=target_BT
            for j in range(len(temp_BT)):
                temp=temp_BT[0:j]+temp_BT[j+1:]
                x_temp=self.lls(temp)
                res=self.target_fun(x_temp[0],x_temp[1],temp)/len(temp)
                if res<Res:
                    Res=res
                    target_BT=temp
                    target_X=x_temp
            if temp_BT==target_BT:
                break
        return target_BT,target_X


    def CRLB(self):
        print('crlb')


def Randd():
    BT=[]
    BT_Los_x, BT_Los_y, Dis_Los, MPE_Los, dis_Los = [], [], [], [],
[]
    BT_Nlos_x, BT_Nlos_y, Dis_Nlos, MPE_Nlos, dis_Nlos = [], [], [],
[], []
    for i in range(8):
        x = random.random() * 1000
        y = random.random() * 680
        d = ((400 - x) ** 2 + (300 - y) ** 2) ** 0.5
        tag = random.randint(0, 1)
        if tag==0:
            BT_Los_x.append(x)
            BT_Los_y.append(y)
            Dis_Los.append(d)
```

```
            n = math.log(d) * random.gauss(0, 1)
            MPE_Los.append(n)
            dis_Los.append(d + n)
            BT.append([x,y,d+n])
        else:
            BT_Nlos_x.append(x)
            BT_Nlos_y.append(y)
            Dis_Nlos.append(d)
            n = math.log(d) * 3*random.expovariate(3)
            MPE_Nlos.append(n)
            dis_Nlos.append(d + n)
            BT.append([x,y,d+n])
    return BT


def main():
    BT=Randd()
    NLOS=Program(BT)
    NLOS.RSWG()

if __name__=="__main__":
    main()
```

附录 C.4　仿真分析

```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from matplotlib import patches
from matplotlib import cm
import numpy as np
import random
import math
import NLOS
import LOS

class Analysis:

    def __init__(self, para):
        self.para={}
        self.para['BT'] = para['BT']
        self.para['BT_LOS'] = para['BT_LOS']
        self.para['BT_NLOS'] = para['BT_NLOS']
        self.para['lamda'], self.para['ave'], self.para['sigma'] =
para['lamda'], para['ave'], para['sigma']
        self.para['MT_x'], self.para['MT_y'] = para['MT_x'],
para['MT_y']

    def update(self):
        aml, lls1, lls_ave, lls_rs, ng, r_ls, lls_nlos, rwgh,sr_ls =
0,0,0,0,0,0,0,0,0
        Algo_Noise = {}
        for i, bt_para in enumerate(self.para['BT_LOS']):
            d=((self.para['MT_x'] - bt_para[0]) ** 2 +
(self.para['MT_y'] - bt_para[1]) ** 2) ** 0.5
            n=math.log(d) * random.gauss(self.para['ave'],
self.para['sigma'])
            self.para['BT_LOS'][i][2]=d+n

        '''
        for i, bt_para in enumerate(self.para['BT_NLOS']):
            d = ((self.para['MT_x'] - bt_para[0]) ** 2 +
(self.para['MT_y'] - bt_para[1]) ** 2) ** 0.5
            n = math.log(d) * (self.para['lamda'] *
np.random.exponential(self.para['lamda']) +
random.gauss(self.para['ave'],self.para['sigma']))
            self.para['BT_NLOS'][i][2] = d + n

        self.para['BT'] = self.para['BT_LOS'] + self.para['BT_NLOS']
        '''
```

```
        instance = LOS.Program(self.para['BT_LOS'])

        aml_x, aml_y = instance.estimate()
        aml = ((aml_x - self.para['MT_x']) ** 2 + (aml_y -
self.para['MT_y']) ** 2) ** 0.5

        lls1_x, lls1_y = instance.LLS_1_E()
        lls = ((lls1_x - self.para['MT_x']) ** 2 + (lls1_y -
self.para['MT_y']) ** 2) ** 0.5

        llsave_x, llsave_y = instance.LLS_AVE_E()
        lls_ave = ((llsave_x - self.para['MT_x']) ** 2 + (llsave_y -
self.para['MT_y']) ** 2) ** 0.5

        llsrs_x, llsrs_y = instance.LLS_RS_E()
        lls_rs = ((llsrs_x - self.para['MT_x']) ** 2 + (llsrs_y -
self.para['MT_y']) ** 2) ** 0.5

        sr_ls_x,sr_ls_y=instance.SR_LS()
        sr_ls = ((sr_ls_x - self.para['MT_x']) ** 2 + (sr_ls_y -
self.para['MT_y']) ** 2) ** 0.5

        r_ls_instance = LOS.Program(self.para['BT_LOS'])
        r_ls_x, r_ls_y = r_ls_instance.R_LS_E()
        r_ls = ((r_ls_x - self.para['MT_x']) ** 2 + (r_ls_y -
self.para['MT_y']) ** 2) ** 0.5
        #lls_aver_instance = LOS.Program(self.para['BT_LOS'])
        #lls_ave_x, lls_ave_y = lls_aver_instance.LLS_AVE_E()
        #lls_ave = ((lls_ave_x - self.para['MT_x']) ** 2 + (lls_ave_y
- self.para['MT_y']) ** 2) ** 0.5
        #lls_rs_instance = LOS.Program(self.para['BT_LOS'])
        #lls_rs_x, lls_rs_y = lls_rs_instance.LLS_RS_E()
        #lls_rs = ((lls_rs_x - self.para['MT_x']) ** 2 + (lls_rs_y -
self.para['MT_y']) ** 2) ** 0.5
        ng_x, ng_y=  instance.NewTon_Gauss()
        ng = ((ng_x - self.para['MT_x']) ** 2 + (ng_y -
self.para['MT_y']) ** 2) ** 0.5

        CRLB = LOS.Program(self.para['BT_LOS'])
        crlb_x, crlb_y =
CRLB.CRLB_LOS(self.para['MT_x'],self.para['MT_y'],self.para['sigma'])
        crlb=(crlb_x+crlb_y)**0.5
        #s_ls_instance=LOS.Program(self.para['BT_LOS'])
```

```
        #r_ls_x, r_ls_y = s_ls_instance.R_LS_E()
        #r_ls = (r_ls_x-self.para['MT_x'])**2+(r_ls_y-
self.para['MT_y'])**2
        '''
        lls_nlos_instance = NLOS.Program(self.para['BT'])
        lls_nlos_x, lls_nlos_y = lls_nlos_instance.LLS_NLOS()
        lls_nlos = ((lls_nlos_x - self.para['MT_x']) ** 2 +
(lls_nlos_y - self.para['MT_y']) ** 2) ** 0.5
        rwgh_instance = NLOS.Program(self.para['BT'])
        NLOS_pos, rwgh_pos = rwgh_instance.RSWG()
        rwgh_x, rwgh_y = rwgh_pos[0], rwgh_pos[1]
        rwgh = ((rwgh_x - self.para['MT_x']) ** 2 + (rwgh_y -
self.para['MT_y']) ** 2) ** 0.5
        '''
        Algo_Noise = {'AML': aml, 'LLS':lls,'Newton':
ng,'SR_LS':sr_ls,'CRLB':crlb,'LLS_AVE':lls_ave,'LLS_RS':lls_rs,'R_LS'
:r_ls}
        return Algo_Noise

    def Change_Noise(self):
        aml, lls1,lls_ave,lls_rs,sr_ls, r_ls, ng,crlb, sigma
=[],[],[],[], [], [], [], [], []
        count, skip,time = 20, 1,10
        for i in range(time):
            print('1')
            self.para['sigma'] = 2+skip * i
            sigma.append(2+skip * i)
            aml1, lls11, sr_ls1, r_ls1,crlb1,
lls_ave1,lls_rs1,ng1=[],[],[], [], [], [], [], []
            for i in range(count):
                x = random.random() * 450+200
                y = random.random() * 450+50
                self.para['MT_x'], self.para['MT_y'] = x, y
                Noise = self.update()
                aml1.append(Noise['AML'])
                lls11.append(Noise['LLS'])
                sr_ls1.append(Noise['SR_LS'])
                #r_ls1.append(Noise['R_LS'])
                lls_ave1.append(Noise['LLS_AVE'])
                lls_rs1.append(Noise['LLS_RS'])
                ng1.append(Noise['Newton'])
                crlb1.append(Noise['CRLB'])
            #print(sr_ls1)
            bias =
```

```
{'LLS_ave':sum(lls_ave1)/count,'LLS_rs':sum(lls_rs1)/count,'CRLB':sum
(crlb1)/count,'AML': sum(aml1) / count, 'LLS': sum(lls11) /
count,'R_LS':sum(r_ls1)/count, 'SR_LS': sum(sr_ls1) / count,
'Newton': sum(ng1) / count}
            aml.append(bias['AML'])
            lls1.append(bias['LLS'])
            lls_rs.append(bias['LLS_rs'])
            lls_ave.append(bias['LLS_ave'])
            sr_ls.append(bias['SR_LS'])
            #r_ls.append(bias['R_LS'])
            ng.append(bias['Newton'])
            crlb.append(bias['CRLB'])

        print(aml,lls1,sr_ls,r_ls,ng,crlb)
        plt.plot(sigma, aml, label='AML')
        plt.plot(sigma, lls1, label='LLS1')
        plt.plot(sigma, lls_ave, label='LLS_Ave')
        plt.plot(sigma, lls_rs, label='LLS_RS')
        plt.plot(sigma, sr_ls, label='SR_LS')
        #plt.plot(sigma, r_ls, label='R_LS')
        plt.plot(sigma, ng, label='Newton')
        plt.plot(sigma, crlb, label='CRLB')
        plt.xlabel('Gauss_Noise Var/m')
        plt.ylabel('Average_Bias/m')
        plt.title('Change Noise')
        plt.legend()
        plt.show()

    def Change_MT_Pos(self):
        aml, lls1, lls_ave, lls_rs, ng, r_ls, lls_nlos, rwgh,sr_ls
=[], [], [], [], [], [], [], [], []
        count = 100
        for i in range(count):
            print('1')
            x = random.random() * 800
            y = random.random() * 600
            self.para['MT_x'], self.para['MT_y'] = x, y
            Noise = self.update()
            aml.append(Noise['AML'])
            lls1.append(Noise['LLS'])
            lls_ave.append(Noise['LLS_AVE'])
            lls_rs.append(Noise['LLS_RS'])
            ng.append(Noise['Newton'])
            r_ls.append(Noise['R_LS'])
```

```
            sr_ls.append(Noise['SR_LS'])
            #lls_nlos.append(Noise['LLS_NLOS'])
            #rwgh.append(Noise['RWGH'])
        r_ls.sort()
        r_ls1=r_ls[0:10]
        bias = [sum(aml) / count, sum(lls1) / count, sum(lls_ave) /
count, sum(lls_rs) / count, sum(ng) /
count,sum(sr_ls)/count,sum(r_ls1)/10]
        name = ['AML', 'LLS1', 'LLS_AVE', 'LLS_RS',
'N_G','SR_LS','R_LS']
        plt.bar(range(len(bias)), bias, tick_label=name)
        plt.xlabel('Algorihm')
        plt.ylabel('Average_Bias')
        plt.title('Change MT')
        plt.show()

    def Bt_pos(self):
        print('hello')

    def Heatmap(self):
        datas1, datas2 = [], []
        for i in range(50):
            print(i)
            data1, data2 = [], []
            self.para['MT_x'] = 20*i
            for j in range(50):
                self.para['MT_y'] =20*j
                for k, bt_para in enumerate(self.para['BT_LOS']):
                    d = ((self.para['MT_x'] - bt_para[0]) ** 2 +
(self.para['MT_y'] - bt_para[1]) ** 2) ** 0.5
                    n = math.log(d+1) * random.gauss(self.para['ave'],
self.para['sigma'])
                    self.para['BT_LOS'][k][2] = d + n
                lls1_instance = LOS.Program(self.para['BT_LOS'])

                lls1_x, lls1_y = lls1_instance.estimate()
                noise1 = ((lls1_x - self.para['MT_x']) ** 2 + (lls1_y -
self.para['MT_y']) ** 2)**0.5
                #noise1 = (lls1_x + lls1_y)
                #print(noise1)
                data1.append(float(noise1))
            datas1.append(data1)
            #print(datas1)
        x_tick=[100*i for i in range(10)]
```

```
        y_tick=[100*i for i in range(10)]
        ax1=plt.subplot()
        for l in range(len(self.para['BT_LOS'])):
            # plt.scatter(BT_x[l], BT_y[l],s=y,  # 设置散点x大小
color='r',  # 设置散点颜色(也可以是一个列表，控制每个点的颜色)marker="o",  #
设置散点形状linewidths=3,  # 设置边框的宽度edgecolors='r',  # 设置边框的颜色
alpha=0.9,  # 设置透明度 )
            # plt.Rectangle([50, 50], 20,20,color='r',fill=True)
            circle = plt.Circle((self.para['BT_LOS'][l][0] / 20,
self.para['BT_LOS'][l][1] / 20), 1, color='y', fill=True)
            plt.gcf().gca().add_artist(circle)
        im1=ax1.imshow(datas1,cmap=cm.binary)
        ax1.set_xticks(np.arange(2,50,5), labels=x_tick)
        ax1.set_yticks(np.arange(2,50,5), labels=y_tick)

        plt.title('AML Error Heat Map')
        plt.xlabel('X/m')
        plt.ylabel('Y/m')
        cbar=plt.colorbar(im1)
        cbar.set_label('RMSE/m')
        plt.show()

    def plot3D(self):
        figure = plt.figure()
        ax = Axes3D(figure)
        X = np.arange(0, 1000, 10)
        Y = np.arange(0, 600, 10)
        X, Y = np.meshgrid(X, Y)
        Z = sum([(self.para['BT_LOS'][i][2]-
((self.para['BT_LOS'][i][0]-X)**2+(self.para['BT_LOS'][i][1]-
Y)**2)**0.5)**2 for i in range(len(self.para['BT_LOS']))])
        ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap='rainbow')
        ax.text(self.para['MT_x'],self.para['MT_y'],15,"MT")
        plt.title('3D')
        plt.xlabel('X/m')
        plt.ylabel('Y/m')
        plt.show()

    def losNlos(self):
        bias = []
        losNoise,nlosNoise=0,0
        name = ['LOS','NLOS']
        count = 100
        for i in range(count):
```

```
            print('1')
            x = random.random() * 800
            y = random.random() * 600
            self.para['MT_x'], self.para['MT_y'] = x, y
            for i, bt_para in enumerate(self.para['BT_LOS']):
                d = ((self.para['MT_x'] - bt_para[0]) ** 2 +
(self.para['MT_y'] - bt_para[1]) ** 2) ** 0.5
                n = math.log(d) * random.gauss(self.para['ave'],
self.para['sigma'])
                self.para['BT_LOS'][i][2] = d + n

            for i, bt_para in enumerate(self.para['BT_NLOS']):
                d = ((self.para['MT_x'] - bt_para[0]) ** 2 +
(self.para['MT_y'] - bt_para[1]) ** 2) ** 0.5
                n = math.log(d) * (self.para['lamda'] *
np.random.exponential(self.para['lamda']) + random.gauss(
                    self.para['ave'], self.para['sigma']))
                self.para['BT_NLOS'][i][2] = d + n

            self.para['BT'] = self.para['BT_LOS'] +
self.para['BT_NLOS']
            lls = LOS.Program(self.para['BT_LOS'])
            nlos = NLOS.Program(self.para['BT'])
            lpos = lls.LLS_1_E()
            losn = ((lpos[0] - self.para['MT_x']) ** 2 + (lpos[1] -
self.para['MT_y']) ** 2) ** 0.5
            npos = nlos.LLS_NLOS()
            nlosn = ((npos[0] - self.para['MT_x']) ** 2 + (npos[1] -
self.para['MT_y']) ** 2) ** 0.5
            losNoise += losn
            nlosNoise += nlosn
        bias = [losNoise/count,nlosNoise/count]
        plt.bar(range(len(bias)), bias, tick_label=name)
        plt.xlabel('Scenario')
        plt.ylabel('Average_Bias/m')
        plt.title('LLS for LOS and NLOS')
        plt.show()

def distribution():
    lls_Noise,Aml_noise, SR_LS_noise= [],[],[]
    for time in range(2):
        print(time)
        Num = 8
        lls_noise,aml_noise,sr_ls_noise=[],[],[]
```

```
        for count in range(10000):
            print(count)
            BT_LOS_c,BT_LOS_s=[], []
            theta=random.random()*2*math.pi
            #MT_x,MT_y=random.random()*1200,random.random()*1200
            if time==0:
                MT_x, MT_y =400+math.sin(theta) *
400*random.random(),400 + math.cos(theta) * 400*random.random()
            else:
                MT_x, MT_y = 800 + abs(math.sin(theta)) * 400 *
random.random(), 800 + abs(math.cos(theta)) * 400 * random.random()
            for i in range(Num):
                x = 400+math.sin(2 * math.pi / Num * i) * 400
                y = 400 + math.cos(2 * math.pi / Num * i) *400
                d = ((MT_x - x) ** 2 + (MT_y - y) ** 2) ** 0.5
                n = random.gauss(0, 10)
                BT_LOS_c.append([x,y,d+n])
            '''
            d0=((MT_x-200)  ** 2 + (MT_y-200) ** 2) ** 0.5
            d1=((MT_x-200)  ** 2 + (MT_y - 600) ** 2) ** 0.5
            d2=((MT_x - 600) ** 2 + (MT_y - 200) ** 2) ** 0.5
            d3=((MT_x - 600) ** 2 + (MT_y - 600) ** 2) ** 0.5
            BT_LOS_s=[[200,200,d0+random.gauss(0,
10)],[200,600,d1+random.gauss(0, 10)],[600,200,d2+random.gauss(0,
10)],[600,600,d3+random.gauss(0, 10)]]
            lls_s=LOS.Program(BT_LOS_s)
            pos1=lls_s.LLS_1_E()
            noises = ((pos1[0] - MT_x) ** 2 + (pos1[1] - MT_y) ** 2) **
0.5
            squ.append(noises)
            '''
            lls=LOS.Program(BT_LOS_c)
            pos_lls=lls.LLS_1_E()
            noise=((pos_lls[0]-MT_x)**2+(pos_lls[1]-MT_y)**2)**0.5
            lls_noise.append(noise)

            #pos_aml = lls.estimate()
            #noise = ((pos_aml[0] - MT_x) ** 2 + (pos_aml[1] - MT_y) **
2) ** 0.5
            #aml_noise.append(noise)

            #pos_srls = lls.estimate()
            #noise = ((pos_srls[0] - MT_x) ** 2 + (pos_srls[1] - MT_y)
** 2) ** 0.5
```

```
            #sr_ls_noise.append(noise)

        lls_Noise.append(sum(lls_noise)/10000)
        #Aml_noise.append(sum(aml_noise)/100)
        #SR_LS_noise.append(sum(sr_ls_noise) / 100)
        #print(sum(squ)/10000)
    name=['Inside','Outside']
    plt.bar(range(len(lls_Noise)), lls_Noise, tick_label=name)
    plt.xlabel('inside and Outside')
    plt.ylabel('Average_Bias/m')
    plt.title('Noise-Inside and Outside')
    plt.show()
    '''
    print(lls_Noise)
    print(lls_Noise)
    x=[i for i in range(200,1100,100)]
    plt.plot(x, lls_Noise,label='LLS')
    #plt.plot(x, Aml_noise, label='AML')
    #plt.plot(x, sr_ls_noise, label='SR-LS')
    plt.xlabel('Radius')
    plt.ylabel('Average_Bias/m')
    plt.title('Noise-Radius')
    plt.show()
    '''

def plot_cicle(centers,rads,MT,BT_x,BT_y):
    fig=plt.figure()
    ax = fig.add_subplot(111)
    i=0
    for center,rad in zip(centers,rads):

cir=plt.Circle((center[0],center[1]),radius=rad,color='y',fill=False)
        plt.plot([center[0], MT[0]], [center[1], MT[1]])
        plt.text((center[0]+MT[0])/2-
5,(center[1]+MT[1])/2+8,'d'+str(i))
        ax.add_patch(cir)
        i=i+1
    plt.plot([centers[0][0] - rads[0], centers[0][0]],
[centers[0][1], centers[0][1]])
    plt.text((centers[0][0] - rads[0] + centers[0][0]) / 2,
(centers[0][1] + centers[0][1]) / 2 + 5, 'd0' + '\'')

    plt.plot([centers[1][0], centers[1][0]], [centers[1][1]-rads[1],
centers[1][1]])
```

```
    plt.text((centers[1][0] + centers[1][0]) / 2+5, (centers[1][1]-
rads[1] + centers[1][1]) / 2, 'd1' + '\'')

    plt.plot([centers[2][0], centers[2][0]], [centers[2][1] +rads[2],
centers[2][1]])
    plt.text((centers[2][0] + centers[2][0]) / 2 + 5, (centers[2][1]
+rads[2] + centers[1][1]) / 2, 'd2' + '\'')

    l3,=ax.plot(BT_x,BT_y,'bs')
    l1,=ax.plot(MT[0],MT[1],'ro')
    l2,=ax.plot(390,290,'k*')
    plt.legend(handles=[l1, l2,l3], labels=['true position',
'estimate','base station'], loc='best')
    plt.axis('equal')
    plt.axis('off')
    plt.show()

def main():
    para={}
    para['BT']=0
    para['BT_NLOS']=0
    para['MT_x'],para['MT_y']=0,0
    para['ave']=0
    para['sigma']=5
    para['lamda']=4

para['BT_LOS']=[[251,251,0],[751,251,0],[251,751,0],[751,751,0],[351,
351,0],[400,200,0],[400,500,0]]
    #print('hello')
    ana = Analysis(para)
    #ana.Heatmap()
    ana.Change_MT_Pos()
    #ana.losNlos()
if __name__=='__main__':
    main()
```