基于无人机编队飞行中的纯方位无源定位

摘 要

本文针对不同编队要求, 基于 AOA 测向定位方法和机器学习算法等理论, 采用 TDOA 法获得有效定位, 结合禁忌搜索算法、**Leader-follower** 算法来提取无人机发射信号的方向信息得到最佳队形. 这不仅满足在日常生活中对于定位场景的需求, 也能够在电子对抗、区域保护等方面提升国家的军事实力, 拥有重大的军事意义.

针对问题 1, 已知编队由 10 架无人机组成, 其中 9 架无人机均匀分布在圆周上, 以编号为 FY00 无人机为中心.

(1) 利用 AOA 测向定位方法, 将二站无源定位拓展为三站无源定位, 结合**内心法**得到接收信号的无人机定位模型; 并使用均方根误差和**几何精度因子**等评价指标评价模型性能, 通过仿真实验验证了模型的有效性.

(2) 以 FY01 的位置为基准, FY00 为圆心建立极坐标系, 得到 FY02~FY09 与 FY01 的位置关系. 基于问题(1)的模型, 利用 **Fisher 信息矩阵**得到目标无人机位置的预测值, 建立**最小均方根误差**最优模型, 并采用蒙特卡洛实验模拟数据, 计算得到至少还需要 **2 架**无人机发射信号, 才能实现有效定位.

(3) 基于问题(1)和(2)的分析, 考虑**时间差**, 采用**联合 TDOA-AOA 加权最小二乘法**以及 RMSE 得到最优预测值, 然后基于**模拟退火思想**以一定概率接受至多两个编号放入黑箱, 最后使用**禁忌搜索算法**迭代得到圆周上无人机的理想位置, 使 9 架无人机均匀分布在半径为 100m 的圆周上.

针对问题 2, 考虑无人机呈锥形编队, 且直线上相邻两架无人机的间距相等. 故采用 Leader-Follower 编队控制方法来完成锥型编队. 我们将锥形边缘的 **FY01, FY11, FY15** 为 Leader 机, 其余无人机为相对 Leader 机或者 follower 机, 然后基于 **DTOA 法**让 follower 机按照规定的偏航角 φ 和相对距离 l 对 Leader 机进行发射信号定位, 形成了规定的拓扑网状结构—锥形编队.

关键词: AOA 测角交叉定位; 内心法; 最小均方根误差; 模拟退火思想

1. 问题重述

1.1 问题背景

运动多站无源定位[1]是实现无人机高精度定位的有效方式. 与单站无源定位相比, 可以更快速的达到高精度定位; 与固定多站无源定位相比, 提高了定位的灵活性, 能够根据具体需要移动到指定方位.

在无人机遂行编队飞行中, 各无人机保持相对静止. 为保持编队队形, 对位置有偏差的无人机进行定位与调整有重要作用, 而采用运动多站无源定位来调整运动中无人机的位置更值得我们研究.

1.2 待解决的问题

在模型的建立过程中, 主要解决的问题有 4 个.

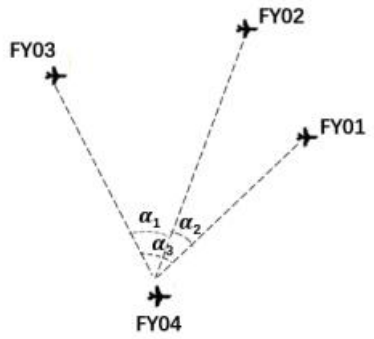


图 1 无人机接收到的方向信息示意图

问题一: 在无人机遂行编队飞行中, 各无人机保持相对静止. 为保持编队队形, 建立模型对位置偏差的无人机精准定位, 使其恢复为固定的编队形态, 问题一需要解决以下三个问题:

(1) 已知由 10 架无人机组成的圆形编队, 位于圆心的无人机和编队中另 2 架无人机给有偏差的无人机发射信号, 建立接收信号无人机的定位模型.

(2) 基于问题一, 在无人机尽可能保持电磁静默的条件下, 已知编号为 FY00 和 FY01 的无人机发射信号, 至少还需要几架无人机发射信号才可以将位置偏离的无人机精准定位?

(3) 已知无人机初始位置的情况下, 每次选择编号为 FY00 的无人机和圆周上最多 3 架无人机发射信号, 调整位置偏差的无人机的位置, 使得无人机最终均匀分布在圆周上.

问题二：无人机集群的编队队形改变为锥形，仍采用纯方位无源定位，设计无人机位置调整方案.

2. 问题分析

2.1 问题 1 的分析

在无人机遂行编队飞行中，为保持编队队形，建立多站无源定位模型对位置偏差的无人机精准定位.

(1) 分析运动多站无源定位精度的影响因素，基于AOA测向定位方法，对二站无源定位进行初步探究；在考虑误差的情况下，三站无源定位结果将不会交于一点，而是围绕位置偏离的无人机形成了一个三角形的区域，根据二站无源定位得到三角形的顶点，再利用内心法，得到有偏差无人机的位置. 最后评价该模型的性能指标并做仿真实验.

(2) 已知编号为 FY00 的无人机在发射信号后与接收信号的无人机形成的正切夹角为固定值；设编号为 FY01 的无人机形成的夹角为自变量，确定其他无人机在可能发送信号的情况下所形成的正切夹角的值，建立函数方程. 基于问题一中多站无源定位模型，得出目标无人机的位置；再根据多站定位性能评价指标，建立 $n-1$ 架无人机与 n 架无人机的均方根误差最小值之差的最大值模型，从而确定 n 的数量.

(3) 利用极坐标与直角坐标的关系，将表 1 中编号为 0~9 的无人机坐标转化为直角坐标，以编号 0 为圆心，则编号 1 在圆周上. 基于问题(1)和(2)的分析，首先采用 AOA 测向定位法以及 RMSE 得到预测值，然后基于模拟退火思想以一定概率接受至多两个编号放入黑箱，最后利用机器学习算法调整 FY02 的理想为位置，剩余无人机的理想位置采用三战时差法逐个得到理想为位置，最终使得 9 架无人机君越分布在半径为 100m 的圆周上.

2.2 问题 2 的分析

考虑无人机呈锥形编队队形，且直线上相邻两架无人机的间距相等. 故采用 Leader-Follower 编队控制方法来完成锥型编队. 我们将锥形边缘的 FY01, FY11, FY15 为Leader机，其余无人机为相对Leader机或者follower机，然后基于DTOA法让 follower 机按照规定的偏航角 φ 和相对距离 l 对 leader 机进行发射信号定位，

形成了规定的拓扑网状结构—锥形编队, 具体见图 2.

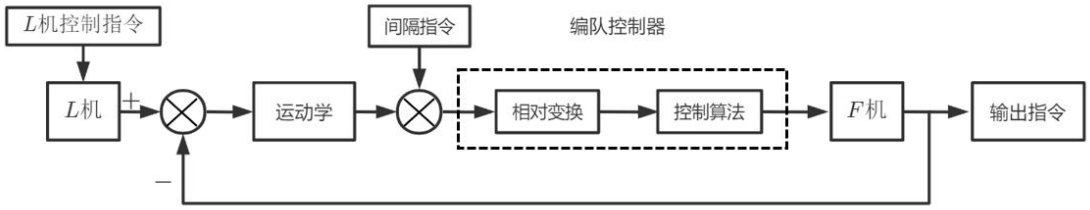


图 2 锥形无人机编队示意图

3. 模型假设

(1) 假设所有发送信号的无人机具有相同的噪声方差;

理由: 保证位置偏离的无人机接收到的信号同步且相等.

(2) 假设从发送信号的无人机到被动接收信号的无人机之间的距离 d_{ji} 固定;

理由: 在各无人机完全相对静止的状态下探究, 不免不必要的误差.

(3) 方向信息的偏差服从正态分布;

理由: 为了简化运算.

(4) **假设标准圆周的圆心为(0, 0), 极坐标为(100, 0)的无人机在圆周上.**

4. 符号说明

符号	说明
$P(x, y)$	被动接收信号无人机位置坐标
$S_i(x_i, y_i)$	圆周上发射信号的无人机
$O(x_0, y_0)$	圆心上的无人机坐标
$P_j(x_j, y_j)$	某个位置有偏差的无人机坐标
$S_{\mathrm{i}}(x_{\mathrm{i}}, y_{\mathrm{i}})$	发送信号的无人机坐标
$d_{ji}(\mathrm{j} \neq i)$	第 i 架无人机与第 j 架无人机之间的距离

θ_{ji}	第 i 架无人机与第 j 架无人机的水平夹角
n_{ji}	方差为 σ^2 的高斯噪声
minRMSE	最小均方根误差
v_{ix}	无人机在 x 轴方向的速度分量
v_{iy}	无人机在 y 轴方向的速度分量
ω_i	无人机的偏航角的角速度

5. 问题(1)模型的建立和求解

5.1 定位精度影响因素分析

基于运动多站纯方位无源定位方法, 被动接收信号无人机的定位精度主要由以下两个因素影响:

位置无偏差的无人机发出的方向信号会受到环境影响以及噪声干扰, 导致接收信号的无人机收到的方向信息有偏差.

$$\tilde{\theta}_i = \theta_i + n_i . \tag{1}$$

为了简化运算, 假设方向信息的偏差服从正态分布 $n_i \sim N(0,\sigma^2)$, 其相应的仰角信息见图 3.

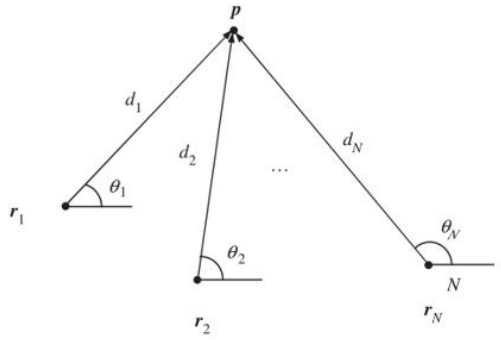


图 3 发射信号无人机的角度

与发送信号无人机的分布方式密切相关. 一个合理的无人机网络几何结构, 会使得被动接收信号无人机的定位性能大大提高. 根据 CRLB 的推导分析, 给出

不同的分布方式对被动接收信号无人机定位精度的影响，进而得出在特定条件下的最优分布方式.

5.2 AOA 测向定位方法

AOA 测向定位方法[2]的基本原理是在没有任何误差的情况下，两台无人机发出的方向信号交于一点，该点即为被动接收信号无人机的位置，算法如下:

利用图 3 的简单集合关系，可得两个无人机定位的方程组

$$\begin{cases} tan\,\alpha_1 = \dfrac{y-y_1}{x-x_1}, \\ tan\,\alpha_2 = \dfrac{y-y_2}{x-x_2}. \end{cases} \tag{2}$$

对于该无人机定位模型的方程组求解，即可实现定位目标功能. 公式(1)可改写为矩阵形式

$$\begin{cases} tan\,\alpha_1 x - y = tan\,\alpha_1 x_1 - y_1 \\ tan\,\alpha_2 x - y = tan\,\alpha_2 x_2 - y_2 \end{cases} \Leftrightarrow \begin{bmatrix} tan\,\alpha_1 & -1 \\ tan\,\alpha_2 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} tan\,\alpha_1 x_1 - y_1 \\ tan\,\alpha_2 x_2 - y_2 \end{bmatrix}. \tag{3}$$

可知当 $det \begin{vmatrix} tan\,\alpha_1 & -1 \\ tan\,\alpha_2 & -1 \end{vmatrix} \neq 0$ 时，方程有唯一解，可求解出对应无人机的位置.

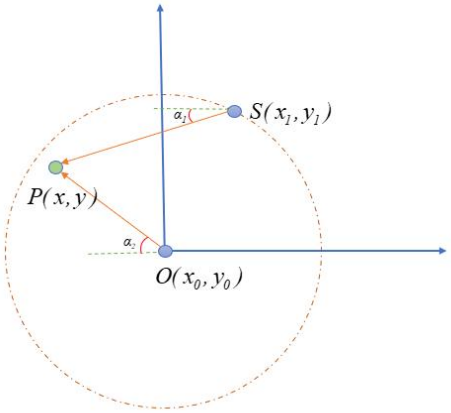


图 4 两点侧向定位图

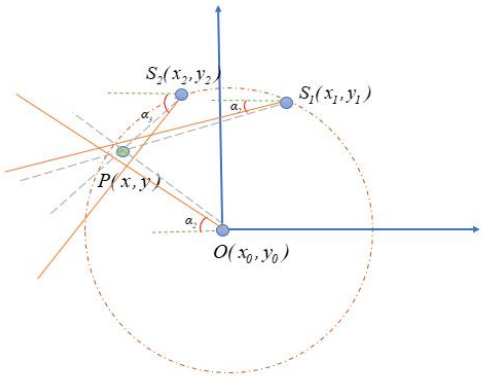


图 5 三点侧向定位图

5.3 三站纯方位三角定位法

针对问题(1)，圆心上的无人机与圆周上的另两架无人机对接收信号的无人机进行定位，在上述定位精度影响因素的分析中得出外界环境与噪声的干扰会影响无人机发出信号的精确度，使三条信号线不会相交于一点，而是形成一个三

角形.

三站纯方位三角定位法是基于 AOA 测向定位方法, 分别求解出三角形的三个顶点坐标, 再利用内心法得到被动接收信号无人机的位置坐标.

图 4 的坐标系中, 发射信号的无人机位置坐标分别为 $O(x_0,y_0)$, $S_1(x_1,y_1)$, $S_2(x_2,y_2)$, 被动接收无人机信号的坐标为 $P(x,y)$. 首先通过 AOA 测向定位方法得到如下方程

$$tan\theta_o=\frac{y-y_0}{x-x_0},\ tan\theta_1=\frac{y-y_1}{x-x_1},\ \tan\theta_2=\frac{y-y_2}{x-x_2}. \tag{4}$$

$S_1(x_1,y_1)$与$S_2(x_2,y_2)$的交点坐标(x_{12},y_{12})

$$\begin{cases} x_{12}=\dfrac{x_2\ sin\theta_1\ cos\theta_2-x_1\ cos\theta_1\ sin\theta_2+(y_1-y_2)\ sin\theta_1\ sin\theta_2}{sin(\theta_1-\theta_2)}, \\ y_{12}=\dfrac{y_1\ sin\theta_1\ cos\theta_2-y_2\ cos\theta_1\ sin\theta_2+(x_2-x_1)\ sin\theta_1\ sin\theta_2}{sin(\theta_1-\theta_2)}. \end{cases} \tag{5}$$

$O(x_0,y_0)$与$S_1(x_1,y_1)$的交点坐标(x_{10},y_{10})

$$\begin{cases} x_{10}=\dfrac{x_0\ sin\theta_1\ cos\theta_0-x_1\ cos\theta_1\ sin\theta_0+(y_1-y_0)\ sin\theta_1\ sin\theta_0}{sin(\theta_1-\theta_0)}, \\ y_{10}=\dfrac{y_1\ sin\theta_1\ cos\theta_0-y_0\ cos\theta_1\ sin\theta_0+(x_0-x_1)\ sin\theta_1\ sin\theta_0}{sin(\theta_1-\theta_0)}. \end{cases} \tag{6}$$

$O(x_0,y_0)$与$S_2(x_2,y_2)$的交点坐标(x_{02},y_{02})

$$\begin{cases} x_{02}=\dfrac{x_2\ sin\theta_0\ cos\theta_2-x_0\ cos\theta_0\ sin\theta_2+(y_0-y_2)\ sin\theta_0\ sin\theta_2}{sin(\theta_0-\theta_2)}, \\ y_{02}=\dfrac{y_1\ sin\theta_0\ cos\theta_2-y_2\ cos\theta_0\ sin\theta_2+(x_2-x_0)\ sin\theta_0\ sin\theta_2}{sin(\theta_0-\theta_2)}. \end{cases} \tag{7}$$

再由内心法解得最终偏差无人机的估计位置坐标为

$$\hat{x}=\frac{ax_{01}+bx_{02}+cx_{12}}{a+b+x},\ \hat{y}=\frac{ay_{01}+by_{02}+cy_{12}}{a+b+c}, \tag{8}$$

其中 a,b,c 是三角形三边的长度

$$
\begin{aligned}
a &= \sqrt{(x_{02}-x_{12})^2+(y_{02}-y_{12})^2},\\
b &= \sqrt{(x_{01}-x_{12})^2+(y_{01}-y_{12})^2},\\
c &= \sqrt{(x_{01}-x_{02})^2+(y_{01}-y_{02})^2}.
\end{aligned}
\tag{9}
$$

由此推广，当有大于三架无人机对位置偏差无人机基于内心法定位时，可以先计算出任意三条信号线构成的三角形内心，再将求得的内心坐标取平均值即为最终偏差无人机的估计位置.

5.4 多站定位性能评价指标

本文针对位置偏差无人机的定位，主要基于以下几种性能指标[3]来评价多站纯方位的定位性能.

(1) 均方根误差 RMSE:

是坐标估计值与真实值偏差的平方与估计次数的比值的平方根. 其估计次数是有限的，真实值只能用最佳值来代替. 通常基于 RMSE 来衡量一组数自身的离散程度，其对极大或极小的数敏感度较高.

$$
RMSE = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\left\| \hat{u}_n - u \right\|_2^2}\ . \tag{10}
$$

(2)几何精度因子 GDOP

描述发送信号的无人机位置和接收信号的无人机之间的几何位置关系对无人机定位的影响. 当 AOA 测向定位出几何区域时，区域的几何精度因子数值越小，则位于该区域的无人机的定位精度越好.

被动接受信号的无人机的 GDOP 表达式为

$$
GDOP = \sqrt{tr(P)}\ . \tag{11}
$$

P 为定位误差的协方差矩阵，tr 表示求迹运算.

(3)圆概率误差 CEP: 从衡量导弹命中尺度引申为以接收信号无人机的真实位置为圆心，发射信号的无人机由于环境影响和噪声误差的影响，将接收信号无人机的估计位置分布在真实位置附近的情况.

5.5 性能试验及分析

(1) 数据模拟生成

对于本问题无人机情况，本文采用服从高斯分布的生成数据，生产 $(0,360)$

中的随机数生成圆周上的一个点坐标θ，接着依次生成圆周上的其他数据点，得到标准位置圆周上无人机点的坐标，接着依据等可能性，随机抽取一个点p_0做为被动接受无人机的点，以其为圆心，以一定干扰参数为标准(详细参数见附录)，生成被测点的坐标p，接着依次在圆周上抽取两个点作为发射信号的无人机的点s_1，s_2，见表 1.

表 1 随机生成的数据点坐标

生成/抽取的点	坐标
生成的圆周上初始点θ	(239.804, 100)
圆周上第一个发射点点s_1	(279.804, 100)
圆周上第二个发射点点s_2	(119.8041, 100)
P 点标准位置	(159.8041, 100)
P 点生成位置	(156.5, 103.07)
仰角α_1	-33.3498
仰角α_2	-11.2360
仰角α_3	87.0339

基于以上生成的数据，给仰角添加一定噪声，做$num = 5000$次蒙特卡洛模拟实验，生成的部分仰角数据特征见图 6.

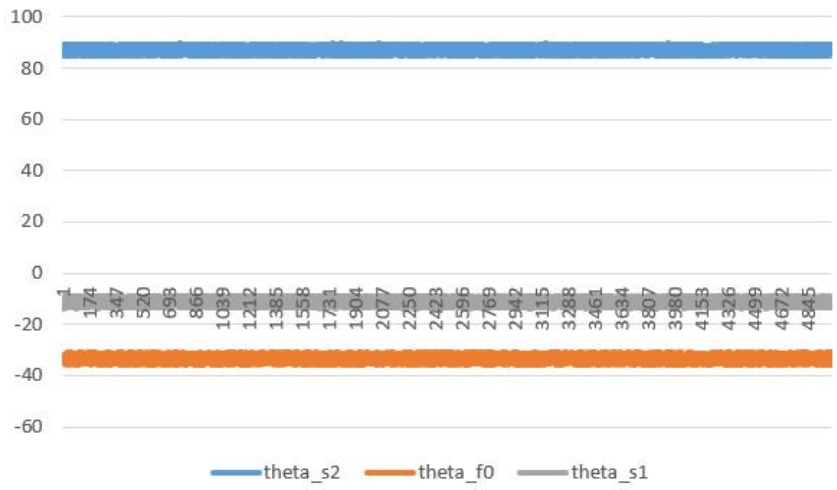


图 6 部分生成的模拟数据点波动图

生成 $1^{o} \sim 10^{o}$ 的仰角干扰数据，基于以上数据进行模拟实验.

(2) 仿真实验结果

依据模拟数据，本文进行相应实验，模拟 5000 次仰角数据，基于以上模型计算相应的估计坐标 (X_i, Y_i),计算相应的 GDOP 与 RMSE，结果见图 7-8.

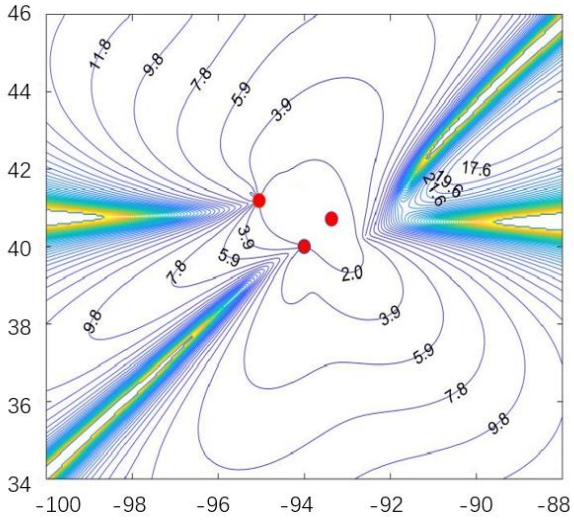


图 7 接受位置 GDOP 分布图

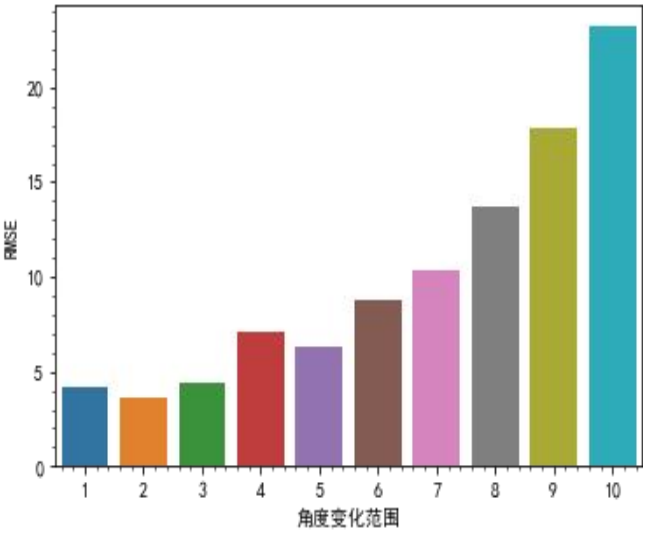


图 8 角度干扰后的 RMSE 图

图7 表示真实点之间的GDOP图，从图中可以看出，添加仰角误差之后的多次模拟实验数据中，等高线面偏向中心点，且图中表示的红色点为最终的平均三角顶点，对于真实数据的估计效果较为精准，从图 8 看出，随着仰角干扰数据幅度的增大，数据点的 RMSE 变化较为明显，即本模型中，对于发射点的坐标依赖较大，这表明，尽量提高发射点的仰角精度可以比较明显有效的降低误差.

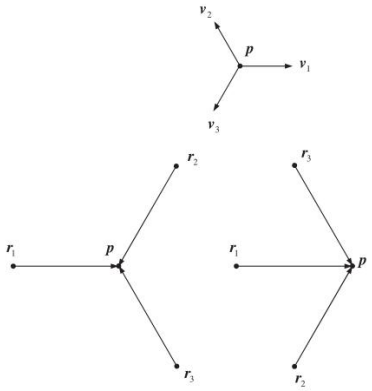
6. 问题(2)模型的建立与求解

6.1 基于 FY01 确定发射信号无人机角度

针对问题(2)，已知两架发射信号无人机 FY00, FY01，其中编号为 FY00 的无人机位于圆心，发射信号后与接收信号的无人机形成的正切夹角已固定. 编号为 FY01 的无人机在圆周上，且圆周上的其他无人机均匀分布，那么任意一个圆周上的发射无人机，都可以定义为 FY01. 以 FY00-FY01 作为极坐标系的 x 轴，建立公式方程.

选择数量不同的无人机，发射的角度不同，且会对结果产生一定影响，本文

分析了几个常见角度(详细见附录)，见图 9-10，角度的会导致选择的仰角不同，且随着无人机数量的增加，分析会越来越困难，但仰角都满足 $tan\theta_1 = \dfrac{y_j - y_1}{x_j - x_1}$，基于此本文建立基于加权最小二乘的定位模型



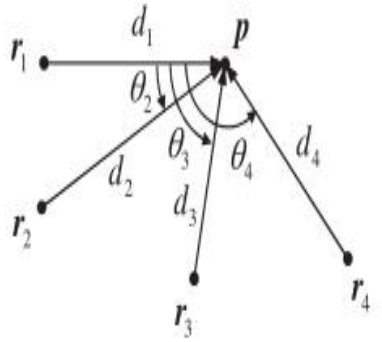


图 9 不同发射位置角度偏差　　**图 10 四个点发射无人机角度分析**

6.2 基于 TDOA 与 AOA 加权最小二乘误差模型

(1) AOA 角度模型

设接收信号的无人机编号为 FY0j，以编号为 FY01 的无人机形成的夹角为自变量，确定其他无人机在可能发送信号的情况下所形成的正弦夹角的值. 假设至少还需要 n 架飞机，接收信号无人机的位置为 (x_j, y_j)，建立函数方程:

$$\begin{aligned}
&\tan\theta_0 = \frac{y_j - y_0}{x_j - x_0},\\
&\tan\theta_1 = \frac{y_j - y_1}{x_j - x_1},\\
&\tan\left(\theta_1 + 20^\circ i\right) = \frac{y_j - y_{1(1+i)}}{x_j - x_{1(1+i)}}, i = 1\cdots 8\left(i \neq j\right).
\end{aligned} \tag{12}$$

圆周上的无人机均匀分布，两个相邻的无人机与圆心构成的圆心角为 40 度，已知编号为 FY01 无人机，当圆周上编号为 FY0j 无人机发射信号时，FY01 与 FY0j 发射的信号线与目标无人机的正确位置，形成圆周角，夹角为 20 度.

当新增发射信号的无人机数量 $n=1$ 时，基于问题一的模型，求出任意两条信号线交点的坐标为 (x_{01}, y_{01})，$(x_{0(1+i)}, y_{0(1+i)})$ 和 $(x_{1(1+i)}, y_{1(1+i)})$.

当新增发射信号的无人机数量 $n=2$ 时，除了上述公式，针对第四架无人机

的公式有如下约束

$$\tan\left(\theta_1+20^{\circ}k\right)=\frac{y_j-y_{1(1+k)}}{x_j-x_{1(1+k)}},k=2\cdots9\left(k\neq i+1\right). \tag{13}$$

任意两条信号线交点的坐标为

$$\left(x_{01},y_{01}\right),\ \left(x_{0(1+i)},y_{0(1+i)}\right),\ \left(x_{1(1+i)},y_{1(1+i)}\right),\ \left(x_{(1+i)k},y_{(1+i)k}\right).$$

(2) TDOA 模型

将被接受无人机的位置点与第 i 个发射点的之间距离表示为d_i，即

$$d_i=\sqrt{(x-x_i)^2+(y-y_i)^2},i=1,2,...,N, \tag{14}$$

其中位置(x,y)为模拟位置点，第i个结点的距离

$$r_i=\left(d_i-d_1\right)+n_{di}=\sqrt{\left(x-x_i\right)^2+\left(y-y_i\right)^2}-\sqrt{\left(x-x_1\right)^2+\left(y-y_1\right)^2}+n_{di},i=2,3,\cdots N.$$

其中n_{di}为 TDOA 测量误差.

忽略定位过程中的 TDOA 测量误差，所涉及的距离差能够表示为

$$r_i+\sqrt{\left(x-x_1\right)^2+\left(y-y_1\right)^2}=\sqrt{\left(x-x_i\right)^2+\left(y-y_i\right)^2}, \tag{15}$$

上式两边同时平方，并通过使用等价关系$x-x_i=\left(x-x_1\right)-\left(x_i-x_1\right)$，可以得到

$$\left(x-x_1\right)\left(x_i-x_1\right)+\left(y-y_1\right)\left(y_i-y_1\right)+r_id_i=\frac{1}{2}\left[\left(x_i-x_1\right)^2+\left(y_i-y_1\right)^2-r_i^2\right],\ \mathrm{i}=2,3,\cdots,N.$$

(3) 联合 TDOA-AOA 加权最小二乘法

使用上述 TDOA 和 AOA 分析方差，考虑所有参考节点的测量值，同时考虑测量误差，构造定位的矩阵方程组.

在使用加权最小二乘法对目标节点的求解过程中，只有在知道目标节点的真实位置坐标时才能获得加权矩阵，这在定位实施过程中是无法提前获知的. 为了获取目标节点的初始位置估计，可以先使用单位矩阵代替加权矩阵，获取初始位置估计后，进而估计出目标节点的位置. 通过仿真表明仅需一次迭代即可完成目标节点的位置估计，多次迭代在定位精度上没有实质性的改进.

6.3 最优无人机数量的模型建立

随着发射信号无人机数量的增加，对目标无人机的定位也越来越精确，但为

了避免外界干扰，应该让无人机少向外发射电磁波信号．于是问题二将转化为最优化问题，在发射信号的无人机数量尽可能少的情况下，使得对接收信号无人机的位置定位精确程度更高．

以最小均方根误差来衡量 n 在不同取值情况下的精确程度．当 $n=1$ 时，基于评价指标，得到 minRMSE_1；当 $n=2$ 时，得到 minRMSE_2，以此类推，建立如下最优递归模型

$$\max = \min RMSE_n - \min RMSE_{n-1}(n \geq 2). \tag{16}$$

计算新增 n 架与 $n-1$ 架发射信号的无人机定位目标无人机时的均方根误差之差，得到一组数据，观察这组数据中的最大值，即误差变化最快的 n 为最优解．

6.4 仿真模拟及结果分析

跟据问题(1) 模拟数据方法，仿真模拟圆周上选取不同数量的无人机，计算其仰角，结果见图 11.

图 11 选择不同数量无人机部分仿真模拟数据

对于上述叙述的模型，本文对其真实位置与模型模拟计算的位置坐标差值，即使用 RMSE 来衡量模型的误差情况，结果见图 12.

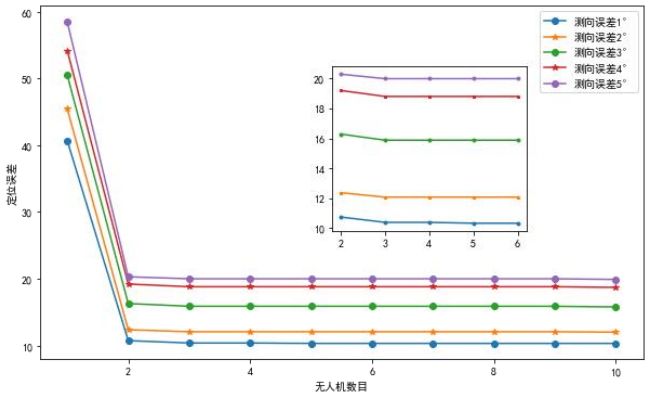


图 12 最优无人机数量

由上图可知，当$n=2$时，误差的变化率最大，所以至少还需要两架无人机发射信号，才能实现无人机的有效定位.

并且随着圆周上的无人机数量增加，其误差逐渐降低，即相同情况下，无人机的数量越多，定位越精准，效率越高，结合问题(1)可知，需要准确的角度定位和较多的无人机数量，即可实现圆形区域定位的精准性.

7. 问题(3)模型的建立与求解

7.1 无人机初始位置分析

问题(3)是对初始位置存在偏差的无人机位置进行调整，为多变量规划模型.已知圆的半径为100m，通过设置调整步长和调整规划，结合目标规划模型求解.

图 13 初始位置与标准位置的距离矩阵

从上图可以清晰看出，除编号为FY00与FY01的无人机不需要移动外，其他的无人机都需要移动. 为了更直观的对数据做对比，制作偏差无人机的初始坐标以标准坐标的对比表，见表2.

表 2 偏差无人机所需要移动到的标准位置

无人机编号	初始位置极坐标	标准位置极坐标
0	(0, 0)	(0, 0)
1	(100, 0)	(100, 0)
2	(98, 40.10)	(40, 100)
3	(112, 80.21)	(80, 100)
4	(105, 119.75)	(120, 100)
5	(98, 159.86)	(160, 100)

6	(112, 199.96)	(200, 100)
7	(105, 204.07)	(240, 100)
8	(98, 280.17)	(280, 100)
9	(112, 320.28)	(320, 100)

在二维空间上，绘制散点图，观察无人机的初始坐标与标准坐标的分布情况，见图 14.

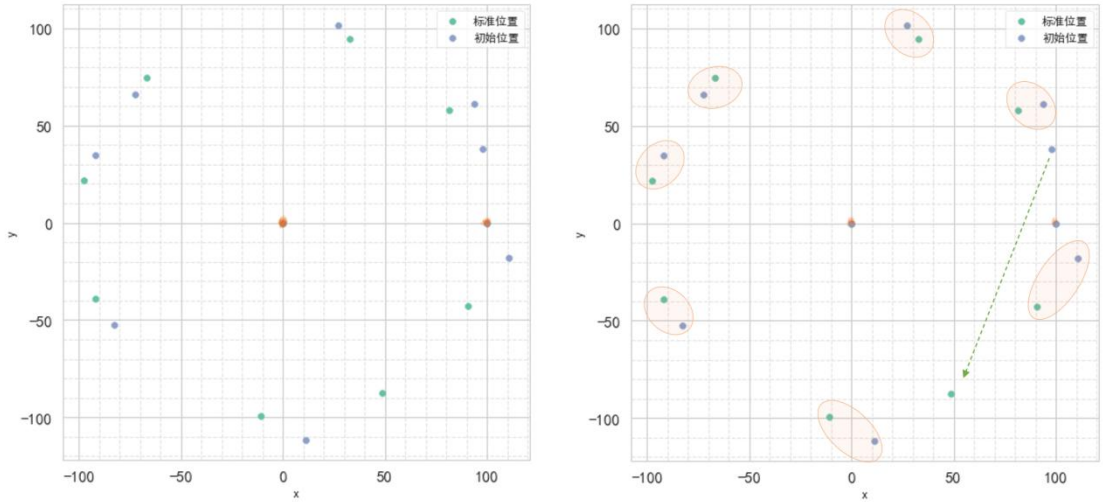


图 14 初始位置与标准位置分布图

7.2 对无人机角度的估计

基于问题(1)和(2)，多个无人机通过 AOA 测量来进行二维定位. 问题一中已知三架特定的无人机，来确定其余七架有偏差无人机的位置. 使用三点定位目标无人机的情况如图 4 所示，其中 $P_j\left(x_j, y_j\right), j=1,2,\cdots 9$ 是某个位置有偏差的无人机坐标；$S_{\mathrm{i}}\left(x_{\mathrm{i}}, y_{\mathrm{i}}\right)$ 是发射信号的无人机坐标，其中 $i=0,1$ 和 1~9 中任意一个数字；$d_{ji}\left(\mathrm{j} \neq i\right)$ 为第 i 架发送信号的无人机与第 j 架有位置偏差的无人机之间的距离.

发送信号的无人机 i 处的 AOA 测量值公式[4]为

$$\widetilde{\theta}_{\mathrm{ji}}=\theta_{ji}+n_{ji}, \theta_{ji}=\tan^{-1}\frac{y_j-y_i}{x_j-x_i}, \tag{17}$$

$n_{ji} \sim N\left(0, \sigma^2\right)$ 是带有方差为 σ^2 的高斯噪声，即无人机发射信号时产生的高斯误差，服从正态分布.

针对 AOA 测向定位, 建立 Fisher 信息矩阵来评估我们对θ_{ji}估计的好坏:

$$\phi = \begin{bmatrix} \phi_{11} & \phi_{12} \\ \phi_{21} & \phi_{22} \end{bmatrix} = J_0^{\,T} \sum^{-1} J_0, \tag{18}$$

J_0是雅可比矩阵, 用来计算被动接收信号无人机的位置

$$J_0 = \begin{bmatrix} u_{12}^{\,T} / d_{12} \\ u_{23}^{\,T} / d_{23} \\ \vdots \\ u_{ji}^{\,T} / d_{ji} \end{bmatrix}, \quad u_{ji} = \begin{bmatrix} -\sin\theta_{ji} \\ \cos\theta_{ji} \end{bmatrix}, \quad d_{ji} = \left\| P_j - S_i \right\| (i \neq j), \tag{19}$$

$\sum$是噪声协方差矩阵

$$\sum = \sigma^2 \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix}_{N \times N}. \tag{20}$$

于是 Fisher 信息矩阵可以变形为:

$$\varphi = \frac{1}{\sigma^2} \sum_{i=1}^{N} \frac{1}{d_i^{\,2}} u_{ji} u_{ji}^{\,T}, \tag{21}$$

u_{ji}是第i个无人机给第j个无人机发送信号的承载向量正交的单位向量,

$$u_{ji} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \frac{P_j - r_i}{d_{ji}}. \tag{22}$$

最终建立 Fisher 行列式最大化等价模型:

$$\min_{\theta_1 \cdots \theta_N} \left\| \sum_{i=1}^{N} \frac{1}{d_{ji}^{\,2}} v_i \right\|^2. \tag{23}$$

7.3 问题(3)模型的建立

考虑编号 0 和 1 为位置无偏差的发射器来估计下一个节点的目标位置. 引入布尔向量$P = [p_1, p_2, \ldots p_k]^T$, $p_i \in \{0,1\}$来表示编号为 2~8 的无人机是否被选择. 如果P中的第i个元素值为 1, 表示其处于接收信号状态. 因此, 待解决的问题转化为: 从 8 个无人机位置中, 选择出参与定位的$K-1$($K<9$)个节点, 使得该无人机编队拥有最好的定位性能. 定义矩阵Φ_P, 表示从 diag(P)中删除掉未被选择的无人机所在行而形成的子矩阵. 其中, Φ_P与P的关系为

$$\Phi_P \Phi_P^T = I_{\|P\|_1}, \Phi_P \Phi_P^T = \mathrm{diag(P)}, \tag{24}$$

其中，$\|P\|_1$ 表示无人机选择向量 P 的 l_1 范数.

引入无人机的选择向量之后，定位估计误差的协方差矩阵的逆可以表示为

$$\begin{aligned} J_p &= E[\Delta v_P \Delta v_P^T]^{-1} = D_P^T (G_P R_{\eta_{\partial P}} G_P^T)^{-1} D_P \\ &= \mathrm{D}^T \Phi_P^T (\Phi_P G \Psi \Psi^T G^T \Phi_P^T)^{-1} \Phi_P D, \end{aligned} \tag{25}$$

其中 $\Psi = I_3 \otimes \Phi_P^T, G_P$ 表示利用布尔向量从系统矩阵 G 中筛选出来的矩阵. D_P 表示从矩阵 D 中去掉未被选择的无人机节点所在行的子矩阵.

因此，基于 A-optimality 模型[5]，将定位估计误差协方差矩阵的逆矩阵的迹作为目标函数. 即可建立无人机优选问题的模型

$$\begin{cases} \min \ tr(J_P^{-1}), \\ \text{s.t. } 1^T P = K - 1, \\ P_i \in \{0,1\} \ \ i = 2,3,...,9. \end{cases} \tag{26}$$

7.4 问题(3)模型的求解

禁忌搜索算法是一种元启发性的随机搜索算法，它从已知信息出发，朝着特定的方向移动，是局部搜索算法的扩展. 针对问题(3)，利用禁忌搜索算法求解，具体流程如图×

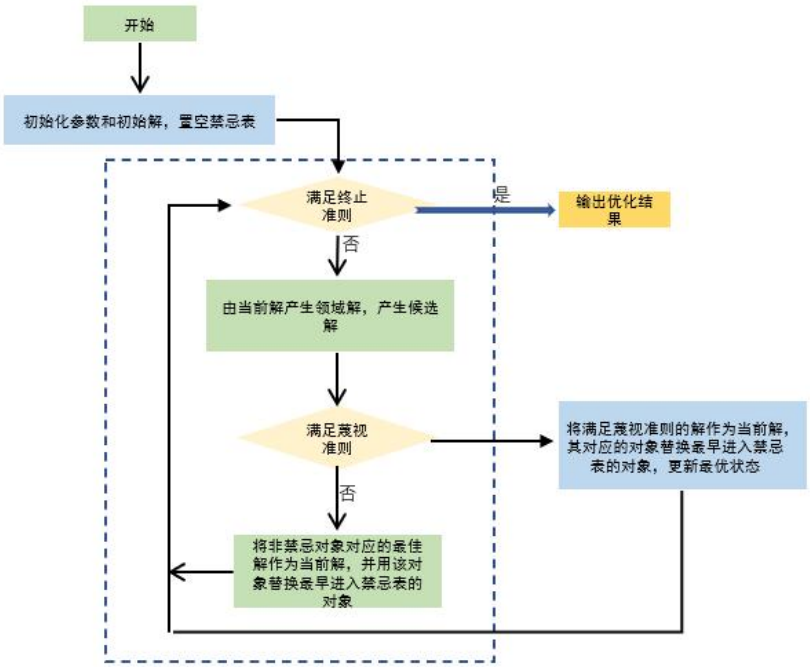


图 15 禁忌搜索算法实现流程图

由(26)式可知, 约束条件中的向量 P 只能取 0 或 1, 即为经典的 0-1 规划问题, 考虑禁忌搜索算法具有灵活的记忆功能, 在搜索过程中可以接受劣解, 从而跳出局部最优解, 具有很强的局部搜索能力. 我们基于禁忌搜索算法给出了 TDOA 无源定位的无人机优选方法. 禁忌搜索算法的关键参数选取见表 3.

表 3 禁忌搜索算法的关键参数的选取

禁忌搜索算法的关键参数选取
A 解的表示方法 选取布尔向量 P 来说明无人机编队结构中的无人机能否被选择. 当 P 中的某一个元素取值为 1 时, 对应的无人机为被选状态; 当值取 0 时, 对应的无人机为未选状态.
B 初始解 禁忌搜索算法不仅可以得到随机的初始解, 还能够将其余的启发式算法结果视为算法的初始解.
C 适配值函数 选用定位误差协方差矩阵的逆矩阵的迹为适配值函数.
D 邻域结构 通过互换当前解中的一对被选状态与未选状态的无人机来生成当前解的邻域解. 故任意一个当前解共有 $C_a = C_{K-1}^1 \cdot C_{M-K}^1 = (K-1)\cdot(M-K)$ 个邻域解.
E 禁忌对象的选取 选取简单的解变换作为禁忌对象, 也就是每一步产生的最优的当前解的布尔向量 P 选取.
F 禁忌长度 指在不考虑藐视准则的情况下禁忌对象不能被选取的最大次数, 即禁忌长度可以看作禁忌对象在禁忌表中的任期. 当且仅当被禁忌对象任期为 0 时, 才可以被解禁. 本文中, 选取禁忌长度为 $TabuL = \sqrt{C_a}$.
G 候选解的选择 在仿真中, 我们选择邻域解中配适值函数相对较低的前 $TabuL$ 个解, 当作当前状态下的候选解.
H 终止准则 设最大送代次数为 $Iter_{\max} = M$, 其中 M 为传感器网络规模的大小.

表 4 无人机调整方案表

调整次数	无人机编号		RMSE
第 1 次调整	发射信息的无人机编号	FY00、FY01	2.59
	调整位置的无人机编号	FY02	
第 2 次调整	发射信息的无人机编号	FY00、FY01、FY02	2.28
	调整位置的无人机编号	FY07	
第 3 次调整	发射信息的无人机编号	FY00、FY01、FY02、FY07	1.56
	调整位置的无人机编号	FY05	
第 4 次调整	发射信息的无人机编号	FY00、FY01、FY02、FY05	1.33
	调整位置的无人机编号	FY04	
第 5 次调整	发射信息的无人机编号	FY00、FY01、FY04、FY07	1.72
	调整位置的无人机编号	FY08	
第 6 次调整	发射信息的无人机编号	FY00、FY01、FY02、FY04	1.69
	调整位置的无人机编号	FY03	
第 7 次调整	发射信息的无人机编号	FY00、FY03、FY05、FY08	1.93
	调整位置的无人机编号	FY09	
第 8 次调整	发射信息的无人机编号	FY00、FY04、FY07、FY09	1.25
	调整位置的无人机编号	FY06	

根据以上模型，求出结果如表 3，从求解结果看出总的 RMSE 为 14.35，整个调整过程误差较小.

8. 问题 2 模型的建立与求解

8.1 模型的建立

首先我们用一个有向非循环图 $F=(V,E,D)$ 表示编队控制图，有限集 $V=\{v_1,v_2,...,v_n\}$ 为包含 N 个顶点且指定每个无人机的控制系统，是 N 个控制的集合，$\dot{x}=f_i(t,x_i,u_i)$，其中，$u_i\in R^m$ 表示无人机是否选中，边集 $E\in V\times V$ 表示无人机之间 leader 机和 follower 机的关系. 如果 u_i 取决于无人机 i，x_i，则令 $(v_i,v_j)=e_{ij}$ 就属于 E. 集合 D 表示每一个 $(v_i,v_j)\in E,v_i\in V$ 的控制目标. 然后根据 leader-follower 理论，建立以下编队运动模型，将其编队的位置定义为水平面(XOY)，从而获得无人机发射信号的型号模型.

编队系统的发射模型为

$$\dot{x}=v_{ix}\cos(\varphi_i)-v_{iy}\sin(\varphi_i)\,, \tag{27}$$

$$\dot{y} = v_{ix}\sin(\varphi_i) - v_{iy}\sin(\varphi_i), \tag{28}$$

$$\dot{\varphi}_i = \omega_i, \tag{29}$$

其中，i 是编队中无人机的编号，当 i 或 L 或者 F 时分别代表是 leader 机或 follower 机，λ 和 φ 分别是 leader 机和 follower 机之间期望保持的距离和偏航角.

假设 leader 机和 follower 机在水平面上的坐标为 (x_L, y_L) 和 (x_F, y_F)，将 λ_x, λ_y 为 leader 到 follower 划线的矢量坐标值，则有

$$\lambda_x = -(x_L - x_F)\cos(\varphi_L) - (y_L - y_F)\sin(\varphi_L), \tag{30}$$

$$\lambda_y = -(x_L - x_F)\sin(\varphi_L) - (y_L - y_F)\cos(\varphi_L), \tag{31}$$

$$\lambda_x = \lambda\cos(\varphi), \ \lambda_y = \lambda\sin(\varphi). \tag{32}$$

因此，基于(27)-(32)式，建立编队发射器的具体模型为

$$e_x = k_x(x_F - x), \tag{33}$$

$$e_y = k_y(y_F - y), \tag{34}$$

$$e_\varphi = k_\varphi(\varphi_F - \varphi). \tag{35}$$

其中 k_x, k_y, k_φ 为各个通道控制器的系数.

8.2 模型的求解

利用 leader-follower 模型，得到编队的位置策略为：利用 TDOA 方法首先调整 leader 机的理想位置，然后用基于禁忌搜索的 TDOA 方法调整每一个 leader 机控制的 follower 机，最终得到锥形编队.

9. 总 结

9.1 模型评价

9.1.1 模型的优点

(1) 多站纯方位三角定位法显著提高了定位接收信号无人机的精确度, 减少了控制误差, 具有很好的解释性.

(2) 最小均方根误差最优模型能够准确反映估计值与真实值之间的误差, 通过误差值的变化率大小衡量最优解说服力强且直观.

(3) 确立发射信号无人机的位置准则, 避免发射信号无人机选取的盲目性.

9.1.2 模型的缺点

发射信号的无人机进行多点定位后, 根据形成区域来确定目标无人机的位置, 仅讨论了利用内心法来解决, 形式过于单一, 没有将多种定位方法进行比较探究.

9.2 模型的改进

多站无人机发射信息后, 接收无人机对数据融合的容错能力应该深入研究; 模型在无人机分布式的结构变化情况还有待提高.

参考文献

[1] 徐海源, 苏成晓, 刘亚奇. 运动多站无源定位中的时差及其变化率估计方法[J]. 电子信息对抗技术, 2022, 37(03): 26-30.

[2] 李谦. 基于TDOA的无源定位技术及其节点优选方法研究[D]. 西安电子科技大学, 2021.

[3] 杨巍. 基于贝叶斯估计的多站纯方位无源定位及优化[D]. 南京理工大学, 2015.

[4] Doğançay K, Hmam H. Optimal Angular Sensor Separation for AOA Localization. Signal Processing, 2008, 88(5): 1248–1260.

[5] Yang X, Niu R. Adaptive Sensor Selection for Nonlinear Tracking via Sparsity-Promoting Approaches[J]. IEEE Transactions on Aerospace and Electronic Systems, 2018: 1-1.

[6] 王晶, 顾维博, 窦立亚. 基于 Leader-Follower 的多无人机编队轨迹跟踪设计[J]. 航空学报, 2020, 41(S1): 723758.

[7] 袁学敏. 基于 Leader-Follower 理论的四旋翼无人机编队控制系统研究[D]. 南京邮电大学, 2016.

[8] 杜国栋. 基于 Leader-follower 理论的四旋翼无人机编队控制方法研究[D]. 重庆邮电大学, 2019.

[9] 李谦. 基于TDOA的无源定位技术及其节点优选方法研究[D]. 西安电子科技大学, 2021.

附 录

附录 1：附件清单

数据表：
模拟数据.xlsx
仿真结果.xlsx
误差.xlsx
位置坐标.xlsx
距离矩阵.xlsx
代码：
code1.py　　问题 1(3)模拟数据及仿真计算
code2.py　　问题 1(2)模拟数据及仿真计算
code3.py　　问题 1(3)计算程序
craph.py　　禁忌搜索代码
state.py　　禁忌搜索代码
plot1.py　　问题 1(1)画图程序
plot2.py　　问题 1(2)及(3)画图程序
plot3.py　　问题 2 画图程序
Leader_Follower.m　　leader-follower 算法代码
calc_LeaderFollower.m　　leader- followe 算法代码
path_plan.m

附录 2：模拟数据参数

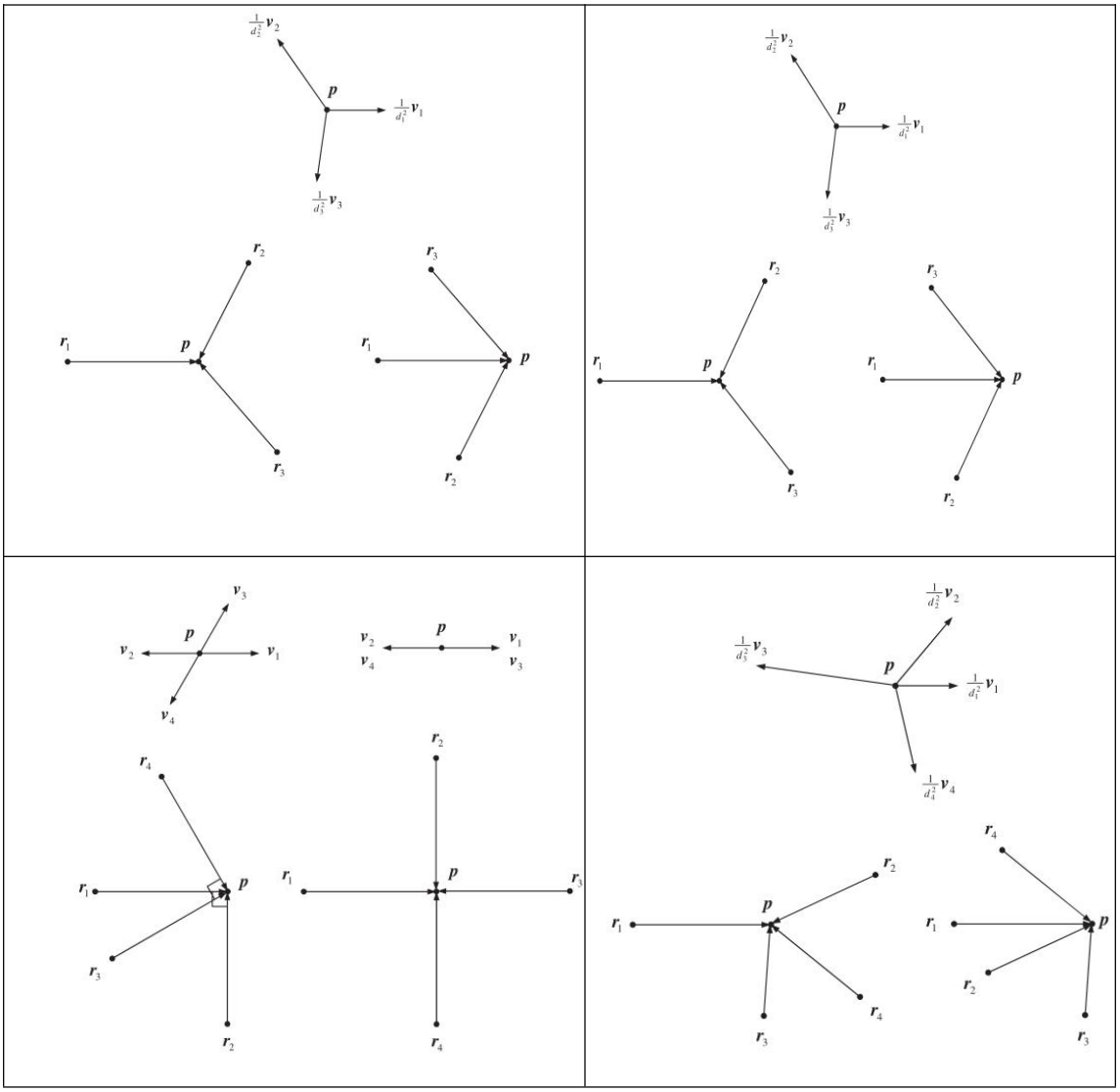
表 1 问题 1(1)模拟参数

生成/抽取的点	坐标
生成的圆周上初始点θ	(239.804, 100)
圆周上第一个发射点点s_1	(279.804, 100)
圆周上第二个发射点点s_2	(119.8041, 100)
P 点标准位置	(159.8041, 100)
P 点生成位置	（156.5, 103.07）
仰角α_1	-33.3498
仰角α_2	-11.2360
仰角α_3	87.0339

表二 问题 1(2)模拟参数

生成/抽取的点	坐标
生成的圆周上初始点 θ	(49.3886, 100)
圆周上第 1 个发射点点 s_1	(9.3886, 100)
圆周上第 2 个发射点点 s_2	(129.3886, 100)
圆周上第 3 个发射点点 s_3	(209.3886, 100)
圆周上第 4 个发射点点 s_4	(289.3886, 100)
圆周上第 5 个发射点点 s_5	(329.3886, 100)
圆周上第 6 个发射点点 s_6	(89.3886, 100)
P 点标准位置	(249.3886, 100)
P 点生成位置	（247.432, 101.07）
仰角 α_1	-41.9028
仰角 α_2	13.7862
仰角 α_3	76.9102
仰角 α_4	23.2128
仰角 α_5	34.4271
仰角 α_6	21.3452

附录 3：不同角度分析图


$\frac{1}{d_2^2}v_2$
p
$\frac{1}{d_1^2}v_1$
$\frac{1}{d_3^2}v_3$
r_2
r_1
p
r_3
r_3
r_1
p
r_2
v_3
p
v_2
v_1
v_4
v_2
v_4
p
v_1
v_3
r_4
r_1
p
r_3
r_2
r_2
r_1
p
r_3
r_4
$\frac{1}{d_2^2}v_2$
$\frac{1}{d_3^2}v_3$
p
$\frac{1}{d_1^2}v_1$
$\frac{1}{d_4^2}v_4$
r_2
r_1
p
r_4
r_3
r_4
r_1
p
r_2
r_3

附录 4：问题 1(3)源程序代码(详细见支撑材料)

python 程序	问题 1(1)数据生成与仿真计算

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
import random
from random import choice
from numpy import sin,cos
mpl.rcParams['font.sans-serif'] = ['FangSong']
mpl.rcParams['axes.unicode_minus'] = False
np.set_printoptions(precision=2) #设置小数位置为 4 位


###############
##模拟次数
number=5000
###
###
#被接受无人机信号点的变化范围半径
theta_p_0=6
polar_p_0=5
###
###
#发射信号无人机角度变化范围
# theta_s_0=3
#
a=float(np.random.uniform(low=0.0, high=360.0, size=1))     ##第一个随机角度
theta=[float((i*40+a)%360) for i in range(0,8)]          ##根据均匀分布计算出其他角度
theta.append(a)      ## 所有角度
print('标准位置的极角依次为：',theta)


t1=np.random.randint(0,8)          ## 选择一个无人机作为 s1
t2=choice([i for i in range(0,8) if i not in [t1]])       ## 从剩下的其中选择一个作为 s2
t3=choice([i for i in range(0,8) if i not in [t1,t2]])   #从剩下的其中选择一个作为 s3



theta_random=np.random.uniform(low=-theta_p_0, high=theta_p_0, size=1)    #极角从-3 度变化到 3 度
polar_random = np.random.uniform(low=-polar_p_0, high=polar_p_0, size=1)   #极径从-5 变化到 5
FY00_x,FY00_y=(0,0) #FY00 的极坐标
s1_theta,s1_r=(theta[t1],100) #s1 的极坐标
```

```
s2_theta,s2_r=(theta[t2],100)  #s2 的极坐标
p_theta,p_r = (theta[t3],100)   #正常标准位置极坐标
p_theta_c,p_r_c=(theta[t3]+theta_random,100+polar_random) #被接收信号的无人机的坐标（加模拟干扰之后）
def to_cartesian(polar_vector):    #接收一对极坐标（长度和弧度）返回相应的笛卡尔坐标
    length, angle = polar_vector[1], polar_vector[0]
    return (length*cos(angle), length*sin(angle))


print('s1 的极坐标为：',(s1_theta,s1_r))
print('s1 的极坐标为：',(s2_theta,s2_r))
print('p 的标准位置坐标为：',(p_theta,p_r))
print('p 的模拟坐标为：',(p_theta_c,p_r_c))

##转换成直角坐标
FY00_x,FY00_y=to_cartesian([0,0]) #FY00 的极坐标
s1_zhi_x,s1_zhi_y=to_cartesian([theta[t1],100]) #s1 的直角坐标
s2_zhi_x,s2_zhi_y=to_cartesian([theta[t2],100])  #s2 的直角坐标
p_zhi_x,p_zhi_y=to_cartesian([p_theta_c,p_r_c]) #被接收信号的无人机的直角坐标


theta_f0=float(np.degrees(np.arctan((FY00_y-p_zhi_y)/(FY00_x-p_zhi_x)))) #FY00 与接收信号的无人机的极角
theta_s1=float(np.degrees(np.arctan((s1_zhi_y-p_zhi_y)/(s1_zhi_x-p_zhi_x))))      #s1 与接收信号的无人机的极角
theta_s2=float(np.degrees(np.arctan((s2_zhi_y-p_zhi_y)/(s2_zhi_x-p_zhi_x))))      #s2 与接收信号的无人机的极角

print('对应的仰角为',(theta_f0,theta_s1,theta_s2))

table = pd.ExcelWriter(r".\data\模拟数据.xlsx")
for num in range(5):
    num=num+1
    result = pd.DataFrame(columns={'theta_s2','theta_s1','theta_f0'})
    low=-num/2
    high=num/2
    for i in range(number):
        result.at[i,'theta_s2']=np.random.uniform(low, high)+theta_s2
        result.at[i,'theta_f0']=np.random.uniform(low, high)+theta_f0
        result.at[i,'theta_s1']=np.random.uniform(low, high)+theta_s1
    result.to_excel(table,f"{num}度范围数据")
table.save()
#s1_theta,s2_theta,p_theta_c,p_theta,p_r_c

## 模拟要定位点的极坐标  （前面文件生成的随机数）
```

```
S1=[s1_theta,100]
S2=[s2_theta,100]
P_V=[p_theta,100]    ##真实点坐标
P=[p_theta_c,p_r_c]    ##模拟误差 p 点坐标
################################
##################直角坐标系
x1,y1=(0,0)    ##圆心点无人机
x2,y2=to_cartesian(S1)    ##s1 无人机
x3,y3=to_cartesian(S2)    ##s2 无人机
p_zhi_v=to_cartesian(P_V)    #真实直角坐标
p_zhi=to_cartesian(P)      #p 模拟直角坐标

def interior_point(theta):
    x12=(x2*sin(theta['theta_f0'])*cos(theta['theta_s1'])-x1*cos(theta['theta_f0'])*sin(theta['theta_s1'])+(y1
-y2)*sin(theta['theta_f0'])*sin(theta['theta_s1']))/sin(theta['theta_f0']-theta['theta_s1'])
    y12=(y1*sin(theta['theta_f0'])*cos(theta['theta_s1'])-y2*cos(theta['theta_f0'])*sin(theta['theta_s1'])+(x2
-x1)*sin(theta['theta_f0'])*sin(theta['theta_s1']))/sin(theta['theta_f0']-theta['theta_s1'])
    x13=(x3*sin(theta['theta_f0'])*cos(theta['theta_s2'])-x1*cos(theta['theta_f0'])*sin(theta['theta_s2'])+(y1
-y3)*sin(theta['theta_f0'])*sin(theta['theta_s2']))/sin(theta['theta_f0']-theta['theta_s1'])
    y13=(y1*sin(theta['theta_f0'])*cos(theta['theta_s2'])-y3*cos(theta['theta_f0'])*sin(theta['theta_s2'])+(x3
-x1)*sin(theta['theta_f0'])*sin(theta['theta_s2']))/sin(theta['theta_f0']-theta['theta_s2'])
    x23=(x3*sin(theta['theta_s1'])*cos(theta['theta_s2'])-x2*cos(theta['theta_s1'])*sin(theta['theta_s2'])+(y2
-y3)*sin(theta['theta_s1'])*sin(theta['theta_s2']))/sin(theta['theta_f0']-theta['theta_s1'])
    y23=(y2*sin(theta['theta_s1'])*cos(theta['theta_s2'])-y3*cos(theta['theta_s1'])*sin(theta['theta_s2'])+(x3
-x2)*sin(theta['theta_s1'])*sin(theta['theta_s2']))/sin(theta['theta_s1']-theta['theta_s2'])
    a=np.sqrt((x13-x23)**2+(y13-y23)**2)
    b=np.sqrt((x12-x23)**2+(y12-y23)**2)
    c=np.sqrt((x12-x13)**2+(y12-y13)**2)
    X=(a*x12+b*x13+c*x23)/(a+b+c)
    Y=(a*y12+b*y13+c*y23)/(a+b+c)
    return X,Y
theta1=pd.read_excel(r'C:/Users/17630/Desktop/ 建 模 国 赛 /B 题 / 模 拟 数
据.xlsx',sheet_name=None,usecols=[1,2,3])
sheetname=list(theta1.keys())
Pre_Table=pd.ExcelWriter(r".\data\\pre_table.xlsx")
for name in sheetname:
#       table_theta_pre(theta1[i],i)
    pre_table=pd.DataFrame(columns={'X_pre','Y_pre'})
    for i in range(len(theta1[name])):
        pre_table.at[i,'X_pre'],pre_table.at[i,'Y_pre']=interior_point(theta1[name].loc[i,:])
    pre_table.to_excel(Pre_Table,name)
Pre_Table.save()

pre_table=pd.read_excel(r".\data\\pre_table.xlsx",sheet_name=None)
```

```
sheetname=list(theta1.keys())
rmse_table=pd.DataFrame(columns=sheetname)
for name in sheetname:
    for i in range(len(pre_table[name])):
        rmse_table.loc[i,name]=float(np.sqrt((pre_table[name].at[i,"Y_pre"]-p_zhi[1])**2+(pre_table[name].at[i,"X_pre"]-p_zhi[0])**2))
```

附录 5：问题 1(2)源程序代码（详细见支撑材料）

python 程序	问题 1(2)处理相关程序

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
from pandas.core.frame import DataFrame
r=[0,4.209110740024201, 3.594069233644708, 4.443126245304123, 7.144918643082947,
6.3582011156060805, 8.7698, 10.3452, 13.679, 17.89, 23.212]
c={"rmse" : r}#将列表 a，b 转换成字典
data=DataFrame(c)#将字典转换成为 dataframe
plt.figure(figsize=(15,6))
mpl.rcParams['font.sans-serif'] = ['FangSong']
mpl.rcParams['axes.unicode_minus'] = False
plt.grid(visible=True,which='major',linestyle='-')
plt.grid(visible=True,which='minor',linestyle='--',alpha=0.5)
plt.minorticks_on()

plt.plot(data.iloc[1:10],'o-.',color='red',linewidth=1,markersize=10,label='rmse')
plt.xlabel("角度变化范围")
plt.ylabel("RMSE")
plt.legend()
plt.show()


plt.title('') # 折线图标题
plt.rcParams['font.sans-serif'] = ['SimHei'] # 显示汉字
plt.xlabel('无人机数目') # x 轴标题
plt.ylabel('定位误差') # y 轴标题
plt.plot(x, y0, marker='o', markersize=6)
plt.plot(x, y1, marker='*', markersize=6) # 绘制折线图，添加数据点，设置点的大小
plt.plot(x, y2, marker='o', markersize=6)
plt.plot(x, y3, marker='*', markersize=6)
plt.plot(x, y4, marker='o', markersize=6)
plt.legend(['测向误差 1°', '测向误差 2°', '测向误差 3°', '测向误差 4°', '测向误差 5°']) # 设置折线名称


left, bottom, width, height = 0.5,0.4,0.25,0.35
plt.axes([left,bottom,width,height])
plt.plot(x[1:6], y0[1:6], marker='o', markersize=3)
plt.plot(x[1:6], y1[1:6], marker='*', markersize=3) # 绘制折线图，添加数据点，设置点的大小
```

```
plt.plot(x[1:6], y2[1:6], marker='o', markersize=3)
plt.plot(x[1:6], y3[1:6], marker='*', markersize=3)
plt.plot(x[1:6], y4[1:6], marker='o', markersize=3)
```

附录 6：问题 1 (3)源程序代码（详细见支撑材料）

python 程序	问题 1 (3)处理相关程序

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import gc
import re
import matplotlib as mpl
import random
from random import choice
from numpy import sin,cos
mpl.rcParams['font.sans-serif'] = ['FangSong']
mpl.rcParams['axes.unicode_minus'] = False
np.set_printoptions(precision=2) #设置小数位置为 4 位
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
sns.set(palette="muted", color_codes=True)
plt.style.use('ggplot')

theta=[0,0,40.10,80.21,119.75,159.86,119.96,240.07,280.17,320.28]
r=[0,100,98,112,105,98,112,105,98,112]

chushi=[[theta[i],r[i]] for i in range(10)]
biaozhun=[[0,0] for i in range(10)]

for i in range(1,10):
    biaozhun[i]=[(40*(i-1))%360,100]

def to_cartesian(polar_vector):    #接收一对极坐标（长度和弧度）返回相应的笛卡尔坐标
    length, angle = polar_vector[1], polar_vector[0]
    return (length*cos(angle), length*sin(angle))
to_cartesian([320.28,112])
chushizhijiao=[to_cartesian(chushi[i]) for i in range(10)]
biaozhun_zhijiao=[to_cartesian(biaozhun[i]) for i in range(10)]

biaozhun_x=[0 for _ in range(10)]
biaozhun_y=[0 for _ in range(10)]
chushi_x=[0 for _ in range(10)]
```

```
chushi_y=[0 for _ in range(10)]
for i in range(10):
    biaozhun_x[i],biaozhun_y[i]=to_cartesian(biaozhun[i])
    chushi_x[i],chushi_y[i]=to_cartesian(chushi[i])
L=[biaozhun_x,biaozhun_y,chushi_x,chushi_y]
L=list(map(list, zip(*L)))
zuobiao=pd.DataFrame(L,columns={'初始位置 x','初始位置 y','标准位置 x','标准位置 y'})
zuobiao.to_excel('.\data\位置坐标.xlsx')



sns.scatterplot(biaozhun_x,biaozhun_y,label="标准位置")
sns.scatterplot(chushi_x,chushi_y,label="初始位置")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.figure(figsize=(8,8))
# sns.scatterplot(zuobiao.loc[:,{'初始位置 x','初始位置 y'}])
plt.show()


l1=biaozhun_zhijiao
l2=chushizhijiao
for i in range(len(l1)):
    for j in range(len(l2)):
        table.at[i,j]=float(np.sqrt((l1[i][0]-l2[j][0])**2+(l1[i][1]-l2[j][1])**2))
formater="{0:.02f}".format
df = table.applymap(formater)
df.to_excel('.\data\与标准位置距离矩阵.xlsx')


# 设置绘图风格
# 设置字体格式
plt.rc('font', family='Times New Roman')
plt.rcParams["font.weight"] = "bold"
plt.rcParams["axes.labelweight"] = "bold"
plt.style.use('ggplot')
sns.set_style('whitegrid')
# 设置画板尺寸
plt.subplots(figsize = (18,12))
mask = np.zeros_like(table, dtype=np.bool)

#               )
pearson=sns.heatmap(table,
                cmap=sns.diverging_palette(20, 220, n=200),
```

```
                annot=True,
                center = 0,
                fmt='.10g',
                annot_kws={'size':20},
               )
cax = plt.gcf().axes[-1]
cax.tick_params(labelsize=20)
plt.xticks(fontsize=15)
      plt.yticks(fontsize=15)
```

附录 7：问题 2 源程序代码（部分代码详细见附件）

matlab 程序	问题四处理相关程序

```
function [err_rho_temp,err_alpha_temp,x_temp,y_temp] = calc_LeaderFollower(x,y,R_loc,j,phi,p_LA,FA,dT,k,rho,alpha,rho_d,alpha_d,fullIntegral)

x_temp(1) = x(j,1);
y_temp(1) = y(j,1);

for i = 2:length(x)
        L_LV = sqrt((x(R_loc,i)-x(R_loc,i-1))^2 + (y(R_loc,i)-y(R_loc,i-1))^2)/dT;
        LA = atan2((y(R_loc,i)-y(R_loc,i-1)),(x(R_loc,i)-x(R_loc,i-1)));
        L_AV = (LA-p_LA)/dT;
        p_LA = LA;
        L_Speed = [L_LV; L_AV];

        formationDerivative = (k*eye(2))*[(rho_d(j-1)-rho);(alpha_d(j-1)-alpha)];
        followerSpeed = [-(1/cos(alpha)) , 0; -(tan(alpha)/rho) , -1] * ...
            (formationDerivative - [-(cos(phi)) , 0; (sin(phi)/rho) , 0] * L_Speed);
        fullDerivative = [-cos(alpha) , 0; sin(alpha)/rho , -1; sin(alpha)/rho , 0]*...
            followerSpeed + [-cos(phi) , 0; sin(phi)/rho , 0; sin(phi)/rho , -1]*...
            L_Speed;
        fullIntegral = fullIntegral + fullDerivative*dT;
        rho = fullIntegral(1);
        alpha = fullIntegral(2);
        phi = fullIntegral(3);
        err_rho_temp(1,i-1) = rho_d(j-1)-rho;
        err_alpha_temp(1,i-1) = alpha_d(j-1)-alpha;
        x_temp(1,i) = x(R_loc,i) - rho*cos(alpha + FA);
        y_temp(1,i) = y(R_loc,i) - rho*sin(alpha + FA);
        FA = phi - alpha + LA - pi;

        newHeading = FA + followerSpeed(2)*dT;
        dS = followerSpeed(1)*dT;
end

end
```