无线电干扰源的
快速自动定位与清除

摘　要

无线电干扰源的测向与规划问题在生产和军事上的应用十分广泛。`no_signal`、`near`、`direction` 为唯一信息来源，建立 **有界误差交会定位—主动选点—覆盖证明搜索—动作耗时调度**的统一框架：四个问题复用同一几何内核，以覆盖证明与严格停止条件保证“清除所有干扰源”的硬约束，再优化时间指标。

针对问题 1，将 $\pm 1^\circ$ 有界误差下的测向观测建模为以检测点为顶点的前向角域 W_i，交会定位区域即诸角域的半平面交 $P=\bigcap_i W_i$，给出直径 $D(P)=\max_{i,j}\|v_i-v_j\|$ 的顶点对枚举算法，并处理空集、无界区域等退化情形。对“直径圆能否覆盖定位区域”，给出只需检查最远顶点对中点的充要判定，并以符合题设测向条件的等边三角形反例给出否定回答；以最小包围圆半径 $\rho(K)\le 20$ 米为可保证清除的充分判据，用 Welzl 算法求解并以支撑点枚举对照。官方演练三点观测复算：$\delta=1^\circ$ 时 $D=23.9967$ 米、$\rho=12.0686$ 米，直径圆不覆盖定位区域，但存在一点可保证清除。

针对问题 2，首次测向后取保守候选集合 $K_1=\Omega\cap W(s_1,\theta_1\pm\delta)\cap B(s_1,1500)$，将接收保证写成一族闭圆盘之交并给出非空判据；进而证明截断角域的最小包围圆恰为 $B(c,a)$ $(a=R/(2\cos\delta)$、$c=s_1+a\boldsymbol{u})$，故安全候选圆盘 $B(c,1000-a)\subseteq\mathcal{Q}$ 严格非空（$\delta=1^\circ$ 时半径约 249.89 米）。在安全区域内以“最坏保守后验包围半径”为主目标、移动与检测成本为次目标主动选点，交会角仅作局部解释与同分筛选指标。

针对问题 3，建立以虚拟时间 $T=L/5+N_{\text{switch}}+5N_{\text{measure}}+5N^{\text{s}}_{\text{clear}}+3N^{\text{f}}_{\text{clear}}$ 为目标的全向源搜索清除模型。构造间距 700 米的 $7\times 7=49$ 点覆盖基线，由 $700\sqrt{2}\approx 989.95<1000$ 米证明任何位置的源均能被某格点发现，从而保证不漏源；维护逐频道覆盖记录与“发现—定位—清除”闭环，给出严格停止条件与请求数预算控制。【演练统计值待填入】

针对问题 4，将定向源建模为位置—半径—朝向的联合假设：`no_signal` 不单独删除空间候选点，避免误删真源；证明 49 点覆盖基线对定向源仍然有效。对再次失去信号的源，给出候选集合的 20 米网格有限圆盘覆盖后备清除方法及请求次数上界 $N\le(\lceil w/20\rceil+2)(\lceil h/20\rceil+2)$，并纳入现实时间预算约束。【演练统计值与正式测试结果（表 1）待填入】

最后对模型的保守性与效率进行评价：几何内核可复用、覆盖保证可验证；局限在于主动选点为采样启发式、后备清除请求数较大，可通过共享测点与自适应覆盖加速。

关键词：交会定位；半平面交；最小包围圆；覆盖证明；主动选点；动作调度

1　问题分析

四个问题构成一条递进的主线：问题 1 的交会定位几何是全部后续问题的基础；问题 2 在单次观测下研究“下一步去哪里测”的主动测量设计；问题 3、问题 4 则是在数量未知的条件下，把定位几何组织成“全域覆盖发现—逐个定位—清除—证明完成”的闭环调度问题。因此不必为每问单独建模，而是复用同一几何内核与观测更新机制。

问题 1 的分析。题面附录 2 第（1）项把测向误差限定在 $[-1^\circ, 1^\circ]$，且同一地点重复检测不改变误差：没有误差分布可利用、原地重复测量不产生新信息，故采用有界误差的集合相交建模——单次测向只能确定源位于以示向度方向为轴、半角 1° 的前向角域内，多检测点角域的交集即为定位区域。题目要求“采用交会定位法”计算“多边形定位区域”的直径（附录 2 第（3）项图 2），角域边界为直线，求交即半平面交这一经典计算几何问题；直径在凸多边形顶点上取得，可用顶点对枚举精确求得。对“直径圆能否覆盖区域”的追问，需要区分“直径”与“外接”两个概念：先给出只需检查最远顶点对中点的充要判定，再构造符合题设测向条件的等边三角形反例说明一般不覆盖。最后，清除以 20 米距离为判据（附录 2 第（8）项），正确的保证清除工具是最小包围圆。

问题 2 的分析。题目要求同时给出第二检测点的“选择策略”与“候选区域”，这是在目标位置未知条件下的主动测量设计，方法取舍由两条题设决定。其一，有效接收半径在 1000 至 1500 米之间且具体值未知（附录 2 第（2）项）：无论真实源位于当前候选集合何处，新检测点都必须保证落在其接收半径内，否则该次检测退化为无效动作，故先构造保守候选集合，再解析证明存在“任何情况下都收到信号”的安全区域。其二，题面只给有界误差而无误差分布与源位置分布（附录 2 第（1）项），概率类方法没有严格依据，故在安全区域内以最坏情形的后验包围半径为主目标、移动与检测成本（附录 2 第（5）、（6）项）为次目标选取检测点，交会角仅作局部几何解释与同分筛选。此外，1800 米圆域限制的是源位置而非机器狗（附录 3），安全选点允许越出圆域；返回 `near` 的分支按附录 2 第（9）项就地清除，不视为策略失败。

问题 3 的分析。全清是硬约束：发现 10 个源或连续多次无信号都不构成停止依据。需要一组对任何源位置都能保证“至少一个检测点落在其有效接收范围内”的覆盖点集，并对每个频道维护覆盖记录，使“未发现”严格等价于“已证明不存在”。在该保证之上，用虚拟时间公式统一度量移动、切换、检测与清除成本，安排发现与处理的调度。

问题 4 的分析。定向源的发射半平面使得“无信号”不能单独否定位置假设：源可能恰在检测点背后发射。把观测视为对“位置—半径—朝向”联合假设的约束即可保持不漏源的保守性；同时证明覆盖基线对定向源依然成立，失去信号后的已定位源可用有限圆盘覆盖逐点清除作为后备。

2　模型假设

1. 示向度误差满足 $|\varepsilon| \le 1^\circ$，且同一检测点重复检测误差不变；仅在不同地点误差才呈现统计规律（题设）。本文取有界误差模型，不假设误差的分布形式。

2. 每个频道至多存在一个干扰源，源总数为 10–16，频道编号 1–20；不同源信号互不影响（题设）。

3. 各源有效接收半径未知但属于 $[1000, 1500]$ 米；信号场强仅随距离衰减，与角度无关（题设）。

4. 定向源有效覆盖角度为定向方向两侧各 90°（含边界）；光学精确定位与清除只依赖距离、不受覆盖角度限制（题设）。

5. 策略仅依赖接口返回的三值观测（`no_signal`、`near`、`direction` 及示向度读数），不引入场强、距离等接口未提供的观测量（工程约定）。

6. 机器狗沿直线匀速行进、停机检测，虚拟时间按题设动作耗时累计；程序运行时间与虚拟时间相互独立（题设）。

3　符号说明

4　问题 1：有界误差交会定位模型

4.1　建模思路来源

本问每种方法的取舍都由题面描述直接确定，对应关系如下。

由误差描述确定集合模型。题面附录 2 第（1）项指出：示向度与真实方位角之间的误差“在一段时间内，同一地点的电磁环境干扰是固定的，所以重复检测不会改变检测误差”，且“从目标区域全局来看，所有这些误差在 $[-1^\circ, 1^\circ]$ 范围内”。因此单次观测不能确定一条精确射线，却可以确定一个必含真实方向的角域；题面没有给出误差的分布形式，原地重复测量又不产生新信息，故采用有界误差下的集合相交，而非直线最小二乘或多次读数平均——后两者既不能保证真源不被排除，也与“同点误差不变”的题设相悖。

由交会定位的规定确定半平面交。问题 1 要求“采用交会定位法（见附录 2）计算多边形定位区域直径”，附录 2 第（3）项的图 2 把定位区域画成两个检测点示向度实线两侧各 1° 虚线所夹的四边形。角域边界是直线，故定位区域是二维线性不等式组的可行域，可用半平面交显式构造，问题随之成为凸几何问题 [6]。

由直径的定义确定顶点枚举。题目规定直径是“区域内任意两点之间距离的最大值”。两点距离关于其变元是凸函数，在紧凸集上的最大值必在极点处取得，故只需在顶点对

表 1　主要符号说明

符号	说明
Ω	目标区域 $\{(x,y): x^2+y^2 \le 1800^2\}$（米）
R_c	干扰源 c 的有效接收半径，$R_c \in [1000, 1500]$ 米
δ	示向度误差界，$\delta = 1^\circ$
s_i, θ_i	第 i 个检测点坐标及其测得的示向度读数
W_i	检测点 s_i 处的前向角域 $[\theta_i - \delta, \theta_i + \delta]$
P	交会定位区域（诸角域的半平面交）
$D(P)$	定位区域 P 的直径
K	某频道源的可能位置候选集合
$\rho(K)$	K 的最小包围圆半径
$\mathcal{Q}$	第二检测点的安全候选区域
$B(c,r)$	以 c 为圆心、r 为半径的闭圆盘（米）
R_-, R_+	有效接收半径的下、上限，1000、1500 米
$J(q)$	第二检测点 q 的最坏保守后验包围半径（米）
$C(q)$	在 q 处完成一次检测的虚拟时间增量（秒）
L	机器狗移动总路程（米）
$N_{\text{switch}}, N_{\text{measure}}, N_{\text{clear}}$	频道切换、检测、清除动作次数
T	完成任务的虚拟总时间（秒）

上枚举，无需任何近似。

由覆盖之问确定“先判定、后反例”。题目追问“以定位区域直径为直径的圆能否覆盖此定位区域”。“直径”是最长弦概念，与“外接”并非同一对象，因此先给出存在覆盖圆的充要判定，再构造符合题设测向条件的反例，据此回答“不一定”。

由清除条件确定最小包围圆。附录 2 第（8）项规定光学精确定位与清除“在距离干扰源不超过 20 米”时进行，且成功与否仅与清除点到干扰源的距离有关。定位精度的最终判据因而是“存在一点到所有可能源位置都不超过 20 米”，即最小包围圆半径 $\rho \le 20$，而非直径不超过 40 米。

4.2 测向观测的前向角域与半平面约束

设第 i 个检测点 $s_i = (s_{ix}, s_{iy})$ 处测得示向度 θ_i（题面附录 2 第（1）项：示向度自 x 轴正向逆时针计量，范围 $[0^\circ, 360^\circ)$）。由假设 1，真实方位角位于 $[\theta_i - \delta, \theta_i + \delta]$ 内，故源必然落在以 s_i 为顶点、以该方向区间为边界的前向角域

$$W_i = \big\{ s_i + t\,u(\varphi) : t \ge 0,\ \varphi \in [\theta_i - \delta, \theta_i + \delta] \big\}, \qquad u(\varphi) = (\cos\varphi, \sin\varphi) \tag{1}$$

内。记二维叉积 $[a, b] = a_x b_y - a_y b_x$ 与 $u_i^{\pm} = u(\theta_i \pm \delta)$。当 $0 < \delta < 90^\circ$ 时，“位于两条边界射线之间”等价于

$$[\,u_i^{-},\, g - s_i\,] \ge 0, \qquad [\,u_i^{+},\, g - s_i\,] \le 0, \tag{2}$$

即 $g - s_i$ 位于下边界射线的左侧、上边界射线的右侧。展开即标准线性约束

$$\sin(\theta_i - \delta)\, x - \cos(\theta_i - \delta)\, y \le \sin(\theta_i - \delta)\, s_{ix} - \cos(\theta_i - \delta)\, s_{iy}, \tag{3}$$

$$-\sin(\theta_i + \delta)\, x + \cos(\theta_i + \delta)\, y \le -\sin(\theta_i + \delta)\, s_{ix} + \cos(\theta_i + \delta)\, s_{iy}. \tag{4}$$

另保留前向约束 $u(\theta_i) \cdot (g - s_i) \ge 0$：它在 $0 < \delta < 90^\circ$ 时冗余，但 $\delta = 0$ 时前两条约束只限定一条直线，加入它才得到正确的射线。角域顶点本身包含在闭集合内，这是保守处理，并不表示在该点仍能获得示向度。采用向量不等式后，359° 与 0° 跨界无需分段比较角度。示向度读数保留两位小数时，可取 $\delta = 1.005^\circ$ 作显式声明的保守裕量；本节推导对任意 $\delta \in (0, 90^\circ)$ 成立，本题算例始终取 $\delta = 1^\circ$。

4.3 定位区域的构造与分类

交会定位区域为诸角域之交

$$P = \bigcap_i W_i = \{ g : Ag \le b \}. \tag{5}$$

每个 W_i 都是闭凸集，故 P 也是闭凸集；它可能为空、为单点、为线段、为有界凸多边形，也可能无界，算法必须显式区分这五种情形，不得混同处理。

边界交点枚举。把约束统一写成 $a_j x + b_j y \le c_j$，两条边界的行列式 $\Delta_{jk} = a_j b_k - a_k b_j \neq 0$ 时，交点为 $x = (c_j b_k - c_k b_j)/\Delta_{jk}$、$y = (a_j c_k - a_k c_j)/\Delta_{jk}$。保留满足全部约束的交点，去重后构造凸包即得 P 的顶点集。

命题 1（极点存在保证判空）　至少有一次观测时，非空的角域交集一定存在极点，因此可通过可行边界交点枚举判空。

证明　取第一个角域的轴向 u_1，其中任意 g 满足 $u_1 \cdot (g - s_1) \ge \cos\delta_1 \, \|g - s_1\|$，且 $\cos\delta_1 > 0$。在非空 P 上最小化线性函数 $u_1 \cdot (g - s_1)$：由该不等式，以任一可行点的函数值截得的下水平集有界，从而是非空紧集，最小值可达；最小化点集位于一条支撑直线上且有界，因而是点或线段，取其点或端点即得 P 的极点。二维线性约束的极点至少有两条法向量线性无关的有效边界（否则沿所有有效边界的公共切向可作双向小扰动，与极点定义矛盾），故必出现在上述交点枚举中。 □

无观测时 $P = \mathbb{R}^2$，应直接判为无界，不能用“没有交点”判为空集。

有界性判定。对非空 P 定义衰退锥 $\mathcal{R} = \{d : Ad \le 0\}$。若存在 $d \neq 0$，则任取 $g_0 \in P$ 都有 $g_0 + td \in P$（$t \ge 0$），P 无界；反之若 P 无界，取 $g_k \in P$ 使 $\|g_k - g_0\| \to \infty$，将 $Ag_k \le b$ 减去 $Ag_0 \le b$、除以 $\|g_k - g_0\|$ 并取极限，知单位向量 $(g_k - g_0)/\|g_k - g_0\|$ 的任一聚点 d 满足 $Ad \le 0$。故 P 有界当且仅当 $\mathcal{R} = \{0\}$。本模型的衰退锥被一个小于 180° 的闭角域约束，若含非零向量必含一条极射线，该射线平行于某条有效约束边界，故只需检查每个法向量 (a_j, b_j) 的两个切向方向 $(b_j, -a_j)$、$(-b_j, a_j)$ 是否满足全部齐次不等式。

分类约定。可行交点为空时返回空集；存在非零衰退方向时返回无界；否则对可行交点去重求凸包，按顶点数与仿射维数区分单点、线段和二维多边形。空集的直径约定为无定义，无界区域的直径为 $+\infty$。显示窗口只用于绘图裁剪：任何大矩形都不能取代衰退锥判定，更不能把截断图形的直径称为原无界区域的直径。

4.4　定位区域直径的顶点定理

定理 1（直径在顶点上取得）　若 $P = \mathrm{conv}\{v_1, \ldots, v_m\}$ 非空且有界，则

$$D(P) = \max_{1 \le i < j \le m} \|v_i - v_j\|. \tag{6}$$

证明　任取 $x, y \in P$，写成凸组合 $x = \sum_i \lambda_i v_i$、$y = \sum_j \mu_j v_j$（权重非负且各自求和为 1），则 $x - y = \sum_{i,j} \lambda_i \mu_j (v_i - v_j)$。由三角不等式，$\|x - y\| \le \sum_{i,j} \lambda_i \mu_j \|v_i - v_j\| \le \max_{i,j} \|v_i - v_j\|$；最远顶点对本身属于 P，上界可达。 □

m 很小（为检测点数的常数倍），直接枚举顶点对即可，复杂度 $O(m^2)$；这是精确算法而非近似，缺点仅在于大规模时的计算量，确有需要时可换旋转卡壳算法 $O(m)$[1, 3, 5]。本文以顶点对枚举为基准实现，旋转卡壳仅作对照。单点的直径为 0，线段的直径为端点距离。

4.5 直径圆覆盖的充要条件与反例

定理 2（覆盖判定） 取任意一组最远顶点 a, b，记 $D = \|a-b\|$、$o = (a+b)/2$。存在直径为 D 的圆覆盖 P，当且仅当所有顶点满足 $\|v_i - o\| \le D/2$。

证明 若半径 $D/2$、圆心 z 的圆同时包含 a, b，则 $D \le \|a-z\| + \|z-b\| \le D$，两处不等号必须同时取等，故 z 在 a, b 连线上且到两点等距，只能是中点 o；$D = 0$ 时结论直接成立。又圆盘是凸集，包含全部顶点等价于包含其凸包，故只需在这唯一可能的圆心处检查全部顶点。 □

该判定同时回答“以某条直径线段为直径的圆能否覆盖”与“能否移动一个同样大小的圆使其覆盖”：检查失败时，移动圆心或更换最远顶点对均不可能成功。定义最大超出量 $E = \max_i \|v_i - o\| - D/2$：精确算术下不会出现严格负值（最远顶点对就在圆周上），$E > 0$ 即覆盖失败，取达到最大值的顶点即得可独立复核的失败证据。

结论：直径圆不一定覆盖定位区域。反例如表 2 所示（误差半角均取 1°，符合题设测向条件）。

表 2 符合题设测向条件的反例：三点观测

检测点	坐标（米）	示向度
s_1	$(-1000, 0)$	1°
s_2	$(540, -500\sqrt{3})$	121°
s_3	$(520, 520\sqrt{3})$	241°

三个角域的下边界方向分别为 $0^\circ, 120^\circ, 240^\circ$，对应左半平面 $y \ge 0$、$\sqrt{3}\,x + y \le 40\sqrt{3}$、$y \le \sqrt{3}\,x$，其交为顶点 $A = (0,0)$、$B = (40,0)$、$C = (20, 20\sqrt{3})$ 的等边三角形。以 s_1 为例，它到三顶点的方向分别为 $0^\circ, 0^\circ, \arctan(20\sqrt{3}/1020) \approx 1.945^\circ$，均在 $[0^\circ, 2^\circ]$ 内，故上边界不切去该三角形；其余两站由绕三角形中心旋转 120° 的对称性同理成立，三个完整前向角域之交恰为该三角形。取真实源为三角形中心 $g_* = (20, 20\sqrt{3}/3)$：以 s_1 为例，真实方位约 0.649°，与读数 1° 的偏差不超过 1°，检测距离约 1020 米，不超过 1500 米，另两站同理，故该反例符合题设的误差界与有效接收半径范围。此时 $D = 40$ 米，以 AB 为直径的圆中心为 $(20, 0)$，到 C 的距离为 $20\sqrt{3} > 20$ 米，不能覆盖；换任何最远顶点对亦然。该三角形的最小包围圆即其外接圆，半径 $40/\sqrt{3} \approx 23.094011$ 米。这同时说明“直径不超过 40 米”不足以保证一次清除。

4.6 最小包围圆与可保证清除

定位的最终目的是清除（附录 2 第（8）项的 20 米距离判据），故正确工具是最小包围圆

$$\rho(K)=\min_{z}\max_{g\in K}\|g-z\|. \tag{7}$$

对有限顶点集 V，$f(z)=\max_{v\in V}\|v-z\|$ 连续且随 $\|z\|\to\infty$ 趋于无穷，最小值存在；若两个不同中心 c_1,c_2 都以半径 r 覆盖 V，则中点 $c=(c_1+c_2)/2$ 对每个顶点满足 $\|v-c\|^2=(\|v-c_1\|^2+\|v-c_2\|^2)/2-\|c_1-c_2\|^2/4<r^2$，与最小性矛盾，故最小包围圆唯一。在最优中心处，圆周上的顶点集 T 满足 $c\in\mathrm{conv}(T)$（否则沿严格分离方向小幅移动中心，可使所有圆周点的距离同时减小，产生矛盾）；二维凸组合经仿射相关权重消元至多保留 3 个点，故最小包围圆由至多 3 个圆周顶点支撑。进而若 $c=\sum_j\lambda_j p_j$ 为支撑点凸组合，则对任意 z 有 $\sum_j\lambda_j\|p_j-z\|^2=r^2+\|c-z\|^2\geq r^2$，给出任何覆盖圆半径不能小于 r 的直接证据。

求解采用带固定随机种子的 Welzl 随机增量算法 [2, 4]：打乱点序，若当前圆不包含新点，则在该点成为边界点的约束下求解较小子问题，边界点至多 3 个；实现用显式栈避免递归深度限制，3 个强制边界点必须用外接圆，不能替换成其无约束最小圆。其期望线性复杂度依赖随机排列分析，不等于任意固定排列的最坏时间保证；另保留支撑点枚举法（单点零半径圆、两点直径圆、三点外接圆，最坏 $O(m^4)$）作独立对照。

$\rho(K)\leq 20$ 时，最小包围圆圆心即可作为清除点；更一般地，所有可保证清除的位置组成

$$C_{20}(K)=\bigcap_{g\in K}B(g,20)=\bigcap_{v\in V}B(v,20), \tag{8}$$

第二个等号来自圆盘的凸性；该集合非空当且仅当 $\rho(K)\leq 20$，非空时可取其中距机器狗最近者以节省移动。相比之下，$D\leq 40$ 只是必要条件（见上节反例）。

4.7 算法流程与演练复算

综合上述结果，问题 1 的完整算法如下。

算法 1（角域交、直径与覆盖证据）

1. 校验观测并进行坐标平移、尺度归一化，保留原误差半角；
2. 按 (2) 生成每次观测的半平面约束（含前向约束）；
3. 枚举非平行边界交点，保留满足全部约束的点；无观测与空集分别处理；
4. 检验非零衰退方向；存在时报告无界并停止有限圆计算；
5. 对可行点去重并求凸包，识别单点、线段或多边形；

6. 枚举顶点对求直径及任意一组最远顶点；

7. 检查其中点到全部顶点的距离，输出最大超出量 E 与失败顶点；

8. 求最小包围圆，换回原坐标，重新检查顶点包含性，输出清除半径余量。

设半平面数为 k（约为观测数的 3 倍）、顶点数为 m：边界交点枚举及全约束检查为 $O(k^3)$，衰退方向检查为 $O(k^2)$，交点排序上界为 $O(k^2 \log k)$，直径计算为 $O(m^2)$，覆盖证据检查为 $O(m)$；随机增量最小圆在随机排列下期望为 $O(m)$。整体仍由半平面交点枚举主导，不能称为总体线性算法。数值实现上，数学结论使用精确不等式，浮点计算先平移、归一化，再以位置容差、平行判定阈值和衰退方向容差分别处理不同问题；接近平行或退化的输入应作精度与容差敏感性检查，任意追加固定小数裕量不能替代严格的误差传播分析。清除判断采用 $\rho + \varepsilon_m \le 20$，其中 ε_m 为工程数值裕量。

演练复算。对官方演练案例 `U5ZW-P9KN-TCAG-R7ST` 的三点观测离线复算，结果如表 3。两种口径下直径圆均不覆盖定位区域（$E > 0$），但最小包围圆半径均小于 20 米，即存在一点可保证清除，与可视化结果一致。该演练未公开源坐标且未执行清除，故仅用于观测交会与计算一致性复核，不能据此声称实测定位误差或清除成功率。【此处可补充定位区域图（figures/）】

表 3　官方演练三点观测的离线复算结果

误差半角	直径 D / 米	最小包围半径 ρ / 米	直径圆最大超出 E / 米
1°	23.996723	12.068583	0.604002
1.005°	24.119088	12.130377	0.608486

5　问题 2：第二检测点的主动选点模型

5.1　建模思路来源

问题 2 给定“某全向干扰源在一个检测点处测得的示向度”，要求“给出第二个检测点的选择策略，以获得关于该干扰源较好的定位效果，并给出第二个检测点的候选区域”。注意源类型明确为全向，不能混入问题 4 的朝向可见性条件。本问方法的取舍同样由题面描述逐项确定。

先接收、后优化。附录 2 第（2）项规定各源有效接收半径在 1000 至 1500 米之间且具体值未知。真实源位置未知时，若第二检测点距真源超过其实际接收半径，第二次检测将返回 `no_signal`，任何定位收益都无从谈起。因此把本问拆成先后两个问题：先构造“无论真源在候选集合内何处都能收到信号”的安全候选区域，再在其中优化定位效果。若只追求较大交会角而不控制到真实源的距离，预期几何收益可能根本无法实现。

由误差描述确定最坏情形指标。题面只给出有界误差（附录 2 第（1）项），没有给出源位置的概率分布，也不存在可独立重抽样的测向噪声模型（同一地点误差固定，重复检测不产生新信息）。因此不能把卡尔曼滤波、协方差传播或费舍尔信息直接用作严格保证，本文以最坏情形的后验包围半径作为选点主目标 [6]；交会角仍有明确的几何意义，但只作局部解释与候选筛选依据。

由区域描述确定活动范围。题面第 1 页的 1800 米圆域限制的是干扰源位置；附录 3 说明机器狗位置通常不构成约束（接口坐标分量绝对值上限仅为数值防护）。故第二检测点允许位于 Ω 之外，安全候选区域不受源位置圆域限制。

由 near 机制确定分支处理。附录 2 第（9）项规定检测点距源不超过 5 米时“无法获得示向度信息，此时可直接进行光学精确定位并清除干扰源”。返回 `near` 不是策略失败，而是定位效果更好的分支（不确定半径不超过 5 米），应单独建模并就地清除。

由动作耗时确定成本项。附录 2 第（5）、（6）项规定检测耗时 5 秒、移动速度 5 米/秒，故第二点的成本取移动与检测耗时之和，作为后验半径择优之后的次目标，避免任意混合米与秒。

5.2 首次观测后的候选集合

设在 s_1 处首次测得 `direction`，示向度 θ_1，记 $u = u(\theta_1)$。由角域模型、题面第 1 页的源位置圆域与附录 2 第（2）项的接收半径上界，真实源 g 满足

$$g \in K_1 = \Omega \cap W(s_1, \theta_1, \delta) \cap B(s_1, R_+), \qquad R_+ = 1500, \tag{9}$$

即保守位置候选集合 K_1 为闭凸紧集；观测一致时它必含真源，从而非空。首次返回 `direction` 还意味着 $\|g - s_1\| > 5$（附录 2 第（9）项），但排除这个开圆盘会破坏凸性，本文暂不使用该信息，保留包含真源的凸外包集合：由此得到的保证可能更保守，但绝不误删真实位置。若计算得到 $K_1 = \varnothing$，说明输入、频道、误差界或数值处理与题设不一致，应返回不一致状态，不能继续给空集计算圆心。

5.3 安全候选区域的定义与非空判据

第二个检测点 q 必须保证：无论真源位于 K_1 内何处，都有 $\|q - g\| \le R_-$ （$R_- = 1000$），从而对任何 $R_c \ge R_-$ 都能收到信号。定义

$$F_K(q) = \max_{g \in K} \|q - g\|, \qquad \mathcal{Q}(K) = \{q : F_K(q) \le R_-\} = \bigcap_{g \in K} B(g, R_-). \tag{10}$$

保证接收不等于保证返回示向度：距离不超过 5 米时可能返回 `near`，该分支定位效果更好（见 6.6 节），不能视为策略失败。

命题 1（F_K 的凸性与稳定性） 对任意非空紧集 K，F_K 为凸函数，且 $|F_K(q_1) - F_K(q_2)| \le \|q_1 - q_2\|$。

证明　对每个固定的 g，距离函数关于 q 凸，逐点取最大值保持凸性；由三角不等式 $\|q_1-g\|\le\|q_2-g\|+\|q_1-q_2\|$，对 g 取最大得到一个方向的不等式，再交换两点即得 Lipschitz 性。 □

故 $\mathcal{Q}(K)$ 为闭凸集；非空时它含于任一 $B(g_0,R_-)$ 内，因而有界。

定理 1（非空判据）　$\mathcal{Q}(K)\neq\varnothing$ 当且仅当 $\rho(K)\le R_-$。

证明　由定义 $\rho(K)=\min_q F_K(q)$；F_K 连续且随 $\|q\|\to\infty$ 趋于无穷，最小值可达，故“存在 q 使 $F_K(q)\le R_-$”与“该最小值不超过 R_-”完全等价。 □

当 $\rho(K)=R_-$ 时，安全区域仅含最小包围圆中心这个单点（唯一性见 5.6 节），因此判据必须使用不严格不等式，不能把等号情形误判为空。

外包圆给出的充分区域。若已知 $K\subseteq B(c,r)$ 且 $r\le R_-$，则对任意 $\|q-c\|\le R_--r$ 的点都有 $\|q-g\|\le\|q-c\|+\|c-g\|\le R_-$，故

$$B(c,R_--r)\subseteq\mathcal{Q}(K). \tag{11}$$

这给出容易计算的充分候选区域；但一个满足 $r>R_-$ 的宽松外包圆只能说明该外包圆不能提供证书，不能据此判定真实的 $\mathcal{Q}(K)$ 为空。

5.4　截断角域的解析最小包围圆

首次角域狭窄（$\delta=1^\circ$）这一题设特征，使 (11) 所需的外包圆可以解析地算到最小。暂时去掉只会缩小集合的 Ω 约束，考虑完整截断角域 $S=W(s_1,\theta_1,\delta)\cap B(s_1,R_+)$，其点可写成 $g=s_1+t\,u(\theta_1+\varphi)$（$0\le t\le R_+$，$|\varphi|\le\delta$）。令

$$a=\frac{R_+}{2\cos\delta},\qquad c=s_1+a\,u. \tag{12}$$

定理 2（截断角域的最小包围圆）　对 $0\le\delta\le45^\circ$，S 的最小包围圆恰为 $B(c,a)$。

证明　先证包含。对 $g=s_1+t\,u(\theta_1+\varphi)\in S$，

$$\|g-c\|^2=t^2+a^2-2at\cos\varphi. \tag{13}$$

固定 φ 时右端关于 t 为凸二次函数，区间最大值在端点取得：$t=0$ 时等于 a^2；$t=R_+$ 时由 $\cos\varphi\ge\cos\delta$ 得 $\|g-c\|^2\le R_+^2+a^2-2aR_+\cos\delta=a^2$，最后一步代入了 $a=R_+/(2\cos\delta)$。故 $S\subseteq B(c,a)$。

再证最小。取 $p_0=s_1$、$p_\pm=s_1+R_+u(\theta_1\pm\delta)$，三点都在 $B(c,a)$ 的圆周上。令 $\lambda_+=\lambda_-=1/(4\cos^2\delta)$、$\lambda_0=1-1/(2\cos^2\delta)$；$\delta\le45^\circ$ 时三权重非负且和为 1，直接代入得 $c=\lambda_0p_0+\lambda_+p_++\lambda_-p_-$。于是对任意中心 z，由加权距离恒等式

$$\sum_j\lambda_j\|p_j-z\|^2=a^2+\|c-z\|^2\ge a^2, \tag{14}$$

故任何覆盖这三点的圆半径不小于 a；而覆盖 S 的圆必须覆盖这三点，结合包含性知最小半径恰为 a。$\delta = 0$ 时两个远端点重合，结论退化为长度 R_+ 的线段直径圆，仍成立。□

该推导不依赖源位置先验与任何概率分布；它给出完整截断角域的精确最小圆，与 Ω 相交后，真实 K_1 的最小圆可能更小。

5.5 本题可直接采用的安全候选圆盘

由 $K_1 \subseteq S \subseteq B(c, a)$ 与 (11)，取

$$\mathcal{Q}_0 = B(c, b), \qquad b = R_- - a = 1000 - \frac{1500}{2\cos\delta}, \tag{15}$$

数值见表 4。

表 4　安全候选圆盘的解析参数

误差半角 δ	轴向中心距离 a / 米	安全候选半径 b / 米
1°	750.114246	249.885754
1.005°	750.115391	249.884609

两种口径下 $b > 0$，故本题首次有效测向后的安全候选区域由解析证明严格非空，无需把“若安全区域为空就去最小包围圆圆心”作为保底步骤（对一般候选集合，$\rho(K) > R_-$ 时根本不存在保证接收的点）。几何上 $\mathcal{Q}_0$ 位于首测示向方向前方约 750 米处、半径约 250 米，与首测点相距至少约 500 米，还天然避免了同点重复测量不产生新信息的问题（附录 2 第 (1) 项）。机器狗不受源位置圆域限制，即使 c 位于 Ω 外，选点也合法。为抵消坐标计算与动作提交中的工程舍入，可取缩小区域 $\mathcal{Q}_\eta = B(c, b-\eta)$（$0 < \eta < b$，如 $\eta = 0.01$ 米）：若圆心与提交位置的合计偏差不超过 η，三角不等式保证仍有 $\|q - g\| \le 1000$；这是可调工程选择，不是题面要求。若有更精细的 K_1 外包圆，可得更大的充分区域；这些扩展均属性能优化，不影响 $\mathcal{Q}_0$ 的正确性。

5.6 “较好的定位效果”的数学定义

保守后验集合。对 $q \in \mathcal{Q}_0$，若返回 `direction` 及读数 β，定义 $K_d(q, \beta) = K_1 \cap W(q, \beta, \delta)$（继续忽略距离超过 5 米的非凸排除信息），真源必在其中；若返回 `near`，定义 $K_n(q) = K_1 \cap B(q, 5)$，故 $\rho(K_n) \le 5$，可直接在当前位置光学定位并清除，满足 20 米清除条件，不必强行再取一个角度。以 $d_{\mathbb{S}^1}$ 记圆周角距离，对 $\|g - q\| > 5$ 的候选位置记真实方位 $h_q(g) = \operatorname{atan2}(g_y - q_y, g_x - q_x)$，则可能读数集合为 $\mathcal{B}(q) = \{\beta : \exists\, g \in K_1,\ \|g - q\| > 5,\ d_{\mathbb{S}^1}(\beta, h_q(g)) \le \delta\}$。这里用所有允许误差描述可能读数，而不是假设误差可以随机独立重抽样。

最坏后验半径指标。定义 $J(q)$ 为所有可能 `direction` 分支 $\rho(K_d(q,\beta))$ 与非空 `near` 分支 $\rho(K_n(q))$ 的上确界，即未来所有允许观测下最坏的保守包围半径；不存在的观测分支不参与取最大。选点主目标即在安全候选区域内降低 $J(q)$。该指标直接服务一次清除要求：$J(q)\le 20$ 时每个可能分支都有半径不超过 20 米的保守定位圆；它比面积更贴合任务——面积较小的集合仍可能十分狭长。注意后验总是 K_1 的子集，故 $J(q)\le\rho(K_1)$，但不一定严格小于：不能仅由"增加一次观测"推断精度必然严格改善，更不能保证两次测向即达 20 米阈值。

移动成本与字典序。连续测量同一频道时，本次动作的虚拟时间增量为

$$C(q)=\|q-s_1\|/5+5\ \text{秒}. \tag{16}$$

以 $\widehat{J}(q)$ 表示实际计算的评估量（6.8 节取保守上界 $\overline{J}(q)$），推荐先按后验半径择优，再保留满足 $\widehat{J}(q)\le J_{\min}+\tau$ 的近优点并取其中 $C(q)$ 最小者，其中 $\tau\ge 0$ 是显式声明的定位容忍量（米），$\tau=0$ 对应严格字典序；本文不预设未经实验支持的最优 τ。

5.7 交会角的局部几何解释

在假定真实位置 g 附近，记两站距离为 r_1,r_2，真实视线单位向量为 u_1,u_2，对应法向量为 n_1,n_2。角误差很小且位置扰动相对站距很小时，方位函数的一阶展开给出 $n_i\cdot\Delta g\approx r_i e_i$（$|e_i|\le\delta_{\text{rad}}$，弧度），即局部不确定区域近似由两条带状约束 $|n_i\cdot\Delta g|\le r_i\delta_{\text{rad}}$ 限定。若视线交会角为 ϕ，法向量矩阵的行列式绝对值为 $|\sin\phi|$，对法向坐标中的矩形作线性逆变换，得近似面积

$$A_{\text{loc}}\approx\frac{4\,r_1r_2\,\delta_{\text{rad}}^2}{|\sin\phi|}. \tag{17}$$

这解释了两点：视线近于平行或反向时误差被显著放大；在距离不变的局部比较下，接近直角的交会较有利。但真实源位置未知，r_2 与 ϕ 随 g 改变，且有限角域并非无限带状区域，故不能由此推出"总应横移固定距离"或"对某个假定源位置成 90° 即全局最优"。本文仅用它优先考虑轴线两侧的候选点，最终仍以后验集合指标评价，交会角只作同分筛选指标。

5.8 圆弧先验的保守离散与读数区间评估

外包切线多边形。直接连接圆周采样点得到的是内接多边形，会错误排除圆盘边缘的真实位置。为复用问题 1 的半平面工具，取 $M\ge 3$ 个均匀法向量 $v_j=u(2\pi j/M)$，定义

$$P_M(o,R)=\{x: v_j\cdot(x-o)\le R,\ j=0,\dots,M-1\}. \tag{18}$$

由柯西不等式，$B(o,R)\subseteq P_M(o,R)$；相邻切线交点到圆心的距离为 $R\sec(\pi/M)$，故单个圆盘外包的最大径向超出不超过 $R[\sec(\pi/M)-1]$。令 $\widetilde{K}_1=P_M(0,1800)\cap W(s_1,\theta_1,\delta)\cap P_M(s_1,1500)$，则 $K_1\subseteq\widetilde{K}_1$，对任何第二读数再求半平面交仍保留真源。靠近切触或

退化情形须加密检查，不能把单圆径向误差未经论证直接当成最终定位半径的误差界。安全选点仍使用解析的 $\mathcal{Q}_\eta$：即使粗糙外包多边形的最小圆偏大，也不能据此推翻解析安全区域的非空结论。若改用任意有界凸外包多边形 $\widetilde{K} = \mathrm{conv}\{v_j\}$，则 $\max_{g\in\widetilde{K}} \|q-g\| = \max_j \|q-v_j\|$（与 5.4 节相同的凸组合论证），其保证接收区域由有限个圆盘约束 $\|q-v_j\| \le 1000$ 给出；离散区域为空时优先加密或回到解析候选盘，不把离散失败误报为问题无解。

读数区间的保守评估。仅枚举少量假定源位置与误差端点，只得到被枚举情形的最大值，可能漏掉真正最坏的读数。将整周按圆周意义划分为若干区间 I_j（中心 β_j、宽度 h_j，要求 $\delta + h_j/2 < 90^\circ$，跨 0° 的区间按圆周距离处理）。

命题 2　对任意 $\beta \in I_j$，有 $W(q,\beta,\delta) \subseteq W(q,\beta_j,\delta+h_j/2)$。

证明　左侧角域内任意非顶点位置，其方向到 β 的圆周距离不超过 δ，而 β 到区间中心的距离不超过 $h_j/2$，由圆周距离的三角不等式，其方向到中心的距离不超过两者之和；顶点在两个闭角域中均被保留。 □

定义 $U_j(q) = \widetilde{K}_1 \cap W(q,\beta_j,\delta+h_j/2)$，则落在该区间的任何真实读数对应的保守后验都含于 $U_j(q)$，从而 $\rho(K_d(q,\beta)) \le \rho(U_j(q))$；`near` 分支取统一上界 $\min\{5,\rho(\widetilde{K}_1)\}$。令 $\overline{J}(q)$ 为所有非空 $U_j(q)$ 的最小包围半径与该上界的最大值，则 $J(q) \le \overline{J}(q)$（精确几何计算下成立；浮点实现还须另行给出包含性与数值误差处理）。二分角区间或增加外包切线都使相应上界单调不增，给出稳定的自适应加密方向。实际可先以少量读数采样粗筛，再对入围点使用完整区间上界：粗筛只决定计算资源分配，最终报告的保守上界必须来自完整覆盖的区间，不能来自粗筛样本最大值。

5.9　选点算法与边界情形

算法 2（安全区域内的有限候选主动测向）

1. 核验首次观测为同一全向源的有效示向度，构造 K_1 或其保守表示；
2. 按 (12) 计算 a、c 与 $b = 1000 - a$，选取满足误差预算的缩小半径 $b-\eta$；
3. 构造有限候选集：圆心 c 及 $q = c + r_\ell(\cos\psi_j\, u + \sin\psi_j\, u_\perp)$（$0 < r_\ell \le b-\eta$、$\psi_j = 2\pi j/N_\psi$、$u_\perp = (-\sin\theta_1, \cos\theta_1)$），全部候选自动满足接收距离保证；
4. 对每个候选点构造外包后验并评估 $\overline{J}(q)$，保存最坏角区间及对应包围圆作为评估依据；
5. 优先取 $\overline{J}$ 较小者，按预先声明的 τ 筛出近优点后比较移动检测成本，必要时以固定顺序消除完全同分；

6. 在计算预算内对入围候选附近加密位置、细分最坏角区间并增加外包切线；预算耗尽时返回当前已验证安全的最好点；

7. 在所选位置测量同一频道：若 `near` 则就地清除；若 `direction` 则用实际读数更新候选集合并复算最小包围圆，半径仍大于 20 米时继续测向，不宣称两次定位已经完成；

8. 若在距离保证条件下意外返回 `no_signal`，核查动作是否被接受、频道是否正确、源是否仍存在及坐标模型是否一致；原因查明前不按正常 `direction` 分支更新候选。

例如可取 3 层半径、每层 16 个方向加圆心共 49 个候选点起步；这只是位置搜索的数值分辨率示例，与问题 3 的 49 点覆盖网格无关，也不是已优化参数。设候选数为 N_q、角区间数为 N_β、每个后验的半平面数为 k，复用问题 1 的边界枚举时主计算量约为 $O(N_q N_\beta k^3)$，另加直径或最小圆计算；选点器自身的运行时间不能忽略。算法输出保证的是：所选点位于已证明安全的区域内，并在给定有限候选与评估精度下按既定规则择优；本文未证明 $q \mapsto J(q)$ 为凸函数，也未证明有限候选的最优点就是整个 $\mathcal{Q}(K_1)$ 中的全局最优点。

边界情形。(i) K_1 为空：报告模型或数据不一致，不构造虚假圆心；(ii) 首次结果已是 `near`：直接清除，不再套用本问流程；(iii) 一般集合满足 $\rho(K) = 1000$：安全区域退化为最小包围圆中心这个单点；(iv) $\rho(K) > 1000$：不存在对该集合全部位置保证接收的点，圆心仅是最小化最坏距离的探索选择，改善交会角若要求偏离该中心还会牺牲最坏距离最优性，两项冲突的要求不能同时称为已保证；(v) 外包圆半径大于 1000：只能说明该证书不足，不能判真实安全区域为空；(vi) 测量轴接近 $0°/360°$：使用向量约束与圆周区间，避免角度断点错误；(vii) 两次测向交会较差：保留完整后验并继续测量，不把小面积当成小直径；(viii) 源类型为定向：距离证书仍成立，但接收还受朝向限制，不能直接沿用本问的接收保证（见问题 4）。【候选区域示意图与选点对比实验待补】

6 问题 3：全向干扰源的搜索定位与清除调度

6.1 虚拟时间与动作成本模型

依据题设动作耗时，完成任务所需虚拟时间

$$T = \frac{L}{5} + N_{\text{switch}} + 5N_{\text{measure}} + 5N_{\text{clear}}^{\text{s}} + 3N_{\text{clear}}^{\text{f}}, \tag{19}$$

其中 L 为移动总路程（含检测与清除引发的移动），频道切换次数只计被接受的检测动作。优化顺序为：先保证全部清除，再减小 T；不能通过少清除源来改善平均时间。虚拟活动时长上限为 100 小时；程序运行时间另受 20 分钟窗口约束，二者独立。

6.2 全域覆盖发现基线

在 s 处对某频道测得 `no_signal`，只能排除以 s 为中心、半径 1000 米圆盘内存在该频道源（有效接收半径可能只有 1000 米），不能一律排除到 1500 米。据此构造覆盖点集

$$S = \{(700i, 700j) : i, j = -3, -2, \ldots, 3\}, \qquad |S| = 49, \tag{20}$$

其覆盖范围 $[-2100, 2100]^2$ 包含 Ω。对任一单元内的源 g，其所在单元的 4 个格点到 g 的距离均不超过 $700\sqrt{2} \approx 989.95 < 1000$ 米，故必存在合格检测点；又 g 位于 4 个格点的凸包内部，经过 g 的任意发射半平面必包含至少一个格点（否则全部格点落入相反开半平面，与 g 在凸包内矛盾）。因此 49 点逐一检测可保证发现该频道的源；源恰在网格线上时按半开区间归入相邻单元处理。49 点 $\times$ 20 频道至多 980 次检测，虚拟耗时上界约 1.32×10^4 秒，在 100 小时上限内有充分设计空间；实际执行中已清除频道跳过、同点共享多频道测量、已发现频道转入定位队列。

6.3 发现—定位—清除闭环

机器狗维护逐频道状态：未知 / 已发现待定位 / 可清除 / 已清除 / 已证明不存在，以及候选集合 K_c。闭环流程为：覆盖扫描发现源 $\to$ 用问题 2 的主动选点缩小 K_c 直至 $\rho(K_c) \le 20 \to$ 移动到 $C_{20}(K_c)$ 内距当前点最近的点清除 $\to$ 更新覆盖记录与队列。多个已发现源按“预期信息改善 / 增量虚拟耗时”排序处理；候选采样点恰返回 `near` 时立即就地清除。

6.4 停止条件与结果统计

停止成功的依据为以下之一：(i) 已成功清除 16 个不同频道的源（数量上界）；(ii) 每个频道都已有清除成功记录或经完整覆盖证明不存在。发现 10 个源、连续若干次无信号、候选采样为空均不是充分停止条件；程序运行时间或窗口不足而退出记为未证明完成，不与成功混记。结果统计按题设给出

$$\text{被清除干扰源个数的比例} = \frac{\text{被清除干扰源的个数}}{\text{干扰源总数}}, \tag{21}$$

$$\text{平均定位清除时间} = \frac{\text{定位清除总时间}}{\text{被清除干扰源的个数}}. \tag{22}$$

【演练测试统计值（清除比例、平均定位清除时间）待演练后填入】

7 问题 4：含定向干扰源的检测与清除

7.1 定向源观测模型

定向源的接收条件是“距离 $\le R_c$ 且检测点位于发射半平面内”。因此 `no_signal` 约束的是位置—半径—朝向的联合假设：若不联合维护朝向可行集合，就不能用它单独删

表 5　问题 3 正式测试结果（按题目表 1 格式）

	测试案例编码	清除干扰源个数	平均定位清除时间	程序运行时间
测试 1	【待填】	【待填】	【待填】	【待填】
测试 2	【待填】	【待填】	【待填】	【待填】
测试 3	【待填】	【待填】	【待填】	【待填】

除空间候选点。可靠基线是对 `no_signal` 只记录“该点覆盖检测已完成”，不做空间删除；加速版本再维护朝向可行集合，且必须验证不误删真源。

7.2　覆盖发现基线对定向源仍然有效

由 7.2 节的凸包论证：源 g 位于所在单元 4 格点的凸包内部，故经过 g 的任意发射半平面必含至少一个格点；该格点同时满足距离不超过 $989.95 < 1000$ 米。因此 49 点逐频道检测对全向与定向源统一保证发现，包括贴近 Ω 边界朝外发射的源（格点允许越出 Ω 访问）。

7.3　失去信号后的后备清除

已发现的定向源在定位过程中可能再次失去信号。对已知有界候选集合 K，构造边长 20 米的全局网格：格中心到格内任意点距离不超过 $10\sqrt{2} \approx 14.14 < 20$ 米，保留与 K 的外包框相交的格子即得无空洞的有限圆盘覆盖，逐点调用清除动作。请求次数上界

$$N \le \Big(\lceil w/20 \rceil + 2\Big)\Big(\lceil h/20 \rceil + 2\Big), \tag{23}$$

其中 w, h 为候选集合外包框的宽与高；单源全失败清除耗时为 $L/5 + 3N$。若 K 外包尺寸较大（如半径 1500 米圆盘的外包框达上万格点），盲扫不满足现实预算，应先继续定位缩小候选集合。现实时间预算按“剩余请求数 $\times$ 实测单请求高分位耗时”估计，不满足预算时转入其他任务或按预算退出。

表 6　问题 4 正式测试结果（按题目表 1 格式）

	测试案例编码	清除干扰源个数	平均定位清除时间	程序运行时间
测试 1	【待填】	【待填】	【待填】	【待填】
测试 2	【待填】	【待填】	【待填】	【待填】
测试 3	【待填】	【待填】	【待填】	【待填】

8 模型评价与推广

优点：（1）四个问题复用同一几何内核（角域半平面交—直径—最小包围圆），模型一致、实现可复用；（2）全程坚持保守几何：有界误差不取平均、无信号只排除 1000 米、`no_signal` 对定向源不删位置、停止条件需覆盖证明，所有“清除完毕”的结论都可验证；（3）虚拟时间公式把调度目标显式化，且区分了虚拟时间与程序运行时间两类约束。

缺点与改进：（1）主动选点采用最坏情形采样，是启发式而非全局最优，可研究对候选集合的更精细离散化；（2）49 点覆盖基线存在冗余，自适应跳点须以保留逐频道覆盖证据为前提；（3）后备清除请求数较大，宜结合朝向可行集合减小候选集合；（4）当前演练验证限于问题 1 的几何内核，问题 3、4 的闭环策略与统计结果有待演练填充。

参考文献

[1] Preparata F P, Shamos M I. Computational Geometry: An Introduction[M]. New York: Springer-Verlag, 1985.

[2] Welzl E. Smallest enclosing disks (balls and ellipsoids)[C]// Maurer H. New Results and New Trends in Computer Science. Lecture Notes in Computer Science, vol 555. Berlin: Springer, 1991: 359–370.

[3] Toussaint G T. Solving geometric problems with the rotating calipers[C]//Proceedings of IEEE MELECON. Athens: IEEE, 1983.

[4] CGAL Project. Bounding Volumes: Min_circle_2[EB/OL]. `https://doc.cgal.org/latest/Bounding_volumes/`, 2026.

[5] McGill School of Computer Science. Rotating Calipers[EB/OL]. `https://www-cgrl.cs.mcgill.ca/~godfried/research/calipers.html`, 2026.

[6] Boyd S, Vandenberghe L. Convex Optimization[M]. Cambridge: Cambridge University Press, 2004.

A　支撑材料文件列表

【提交前更新为实际支撑材料压缩包内容，须与附录 B 源码一致】

表 7　支撑材料文件列表

文件/目录	内容	说明
geometry/	几何内核源代码	角域交、直径、最小包围圆
planning.py	问题 2 安全候选盘计算	first_bearing_candidate_disk
examples/q1/	问题 1 输入样例	含等边三角形反例
output/q1/	问题 1 输出与演练记录	优化报告、可视化

B　程序源代码

以下为支撑模型计算的核心源代码，完整可运行代码以支撑材料压缩包为准。

B.1　src/mcm26b/geometry/localization.py：交会定位几何内核

```
"""Intersect forward bearing wedges without an artificial bounding box.

All calculations are floating point with normalized station coordinates.
The reference algorithm deliberately favors auditability over asymptotic speed.
Near-parallel plane pairs are intersected instead of skipped (only exactly
parallel pairs are discarded), and the feasibility tolerance scales with the
candidate vertex magnitude, so legitimate far-away vertices of long thin
regions are never silently dropped.
"""

from dataclasses import dataclass
from itertools import combinations
from math import cos, sin, radians, hypot, isfinite
from random import Random

Point = tuple[float, float]
EPS = 1e-9
PARALLEL_EPS = 1e-12


@dataclass(frozen=True)
class Measurement:
    x: float
    y: float
    bearing_deg: float
    half_angle_deg: float = 1.0
```

```
27
28      def __post_init__(self):
29          values = (self.x, self.y, self.bearing_deg, self.half_angle_deg)
30          if any(isinstance(v, bool) or not isinstance(v, (int, float))
31                 or not isfinite(v) for v in values):
32              raise ValueError("坐标、示向度和误差半角必须是有限数值")
33          if not 0 <= self.half_angle_deg < 90:
34              raise ValueError("误差半角必须在 [0, 90) 度；题设为1度")
35
36
37  def cross(a: Point, b: Point) -> float:
38      return a[0] * b[1] - a[1] * b[0]
39
40
41  def distance(a: Point, b: Point) -> float:
42      return hypot(a[0] - b[0], a[1] - b[1])
43
44
45  def halfplanes(m: Measurement) -> list[tuple[float, float, float]]:
46      """Return a*x+b*y<=c with unit normals, including the forward constraint."""
47      theta = m.bearing_deg % 360
48      lower, upper = radians(theta - m.half_angle_deg), radians(theta + m.half_angle_deg)
49      middle = radians(theta)
50      normals = [(sin(lower), -cos(lower)), (-sin(upper), cos(upper)),
51                 (-cos(middle), -sin(middle))]
52      return [(a, b, a * m.x + b * m.y) for a, b in normals]
53
54
55  def convex_hull(points: list[Point], eps: float = EPS) -> list[Point]:
56      """Monotone chain, CCW, no repeated final vertex; retain segment endpoints."""
57      ordered = sorted(points)
58      unique = []
59      for p in ordered:
60          if not unique or distance(p, unique[-1]) > eps:
61              unique.append(p)
62      if len(unique) <= 2:
63          return unique
64
65      def chain(seq):
66          out = []
67          for p in seq:
68              while len(out) >= 2:
69                  a = (out[-1][0] - out[-2][0], out[-1][1] - out[-2][1])
70                  b = (p[0] - out[-1][0], p[1] - out[-1][1])
71                  # Distance-to-line tolerance, not an unscaled area tolerance.
72                  if cross(a, b) > eps * max(hypot(*a), hypot(*b), eps):
```

```
                    break
                out.pop()
            out.append(p)
        return out

    return chain(unique)[:-1] + chain(reversed(unique))[:-1]


def diameter(points: list[Point]):
    if not points:
        raise ValueError("空点集没有本程序定义下的直径")
    a = b = points[0]
    value = 0.0
    for p, q in combinations(points, 2):
        d = distance(p, q)
        if d > value:
            value, a, b = d, p, q
    return value, (a, b)


def _circumcircle(a: Point, b: Point, c: Point):
    u = (b[0] - a[0], b[1] - a[1])
    v = (c[0] - a[0], c[1] - a[1])
    det = 2 * cross(u, v)
    if abs(det) <= PARALLEL_EPS * max(hypot(*u) * hypot(*v), 1e-300):
        return None
    uu, vv = u[0] ** 2 + u[1] ** 2, v[0] ** 2 + v[1] ** 2
    center = (a[0] + (uu * v[1] - vv * u[1]) / det,
              a[1] + (u[0] * vv - v[0] * uu) / det)
    return center, distance(center, a)


def _enclosing_circle_reference(points: list[Point], eps: float = EPS):
    """Enumerate all 1/2/3-point supports; O(m^4), appropriate for small m.

    Radius is recomputed against every vertex to preserve numerical containment.
    Coordinates should be normalized when there are very large offsets.
    """
    if not points:
        raise ValueError("空集不计算最小包围圆")
    xs, ys = [p[0] for p in points], [p[1] for p in points]
    best_c = ((min(xs) + max(xs)) / 2, (min(ys) + max(ys)) / 2)
    best_r = max(distance(best_c, p) for p in points)

    def consider(center, radius):
        nonlocal best_c, best_r
```

```
        if radius > best_r + eps:
            return
        actual = max(distance(center, p) for p in points)
        if actual <= radius + eps and actual < best_r:
            best_c, best_r = center, actual

    for p in points:
        consider(p, 0.0)
    for a, b in combinations(points, 2):
        consider(((a[0] + b[0]) / 2, (a[1] + b[1]) / 2), distance(a, b) / 2)
    for a, b, c in combinations(points, 3):
        circle = _circumcircle(a, b, c)
        if circle is not None:
            consider(*circle)
    return best_c, best_r


def smallest_enclosing_circle(points: list[Point], eps: float = EPS):
    """Welzl's randomized incremental algorithm, using an explicit call stack.

    The fixed shuffle makes results reproducible. Expected linear work refers
    to randomized input order, not a worst-case guarantee for this fixed seed.
    Normalize internally so this public helper also handles translated inputs.
    The exhaustive reference is retained for independent regression comparisons.
    """
    if not points:
        raise ValueError("空集不计算最小包围圆")
    if not isfinite(eps) or eps <= 0:
        raise ValueError("eps必须为有限正数")
    if any(len(p) != 2 or any(isinstance(x, bool) or not isinstance(x, (int, float))
                              or not isfinite(x) for x in p) for p in points):
        raise ValueError("点集必须由有限二维坐标组成")
    origin = points[0]
    scale = max(distance(origin, p) for p in points)
    if scale == 0:
        return tuple(origin), 0.0
    local = [((p[0]-origin[0])/scale, (p[1]-origin[1])/scale) for p in points]
    tolerance = min(eps/scale, EPS)
    Random(20260910).shuffle(local)
    # Frame = (remaining prefix length, forced boundary, returning from child).
    stack = [(len(local), (), False)]
    circle = None
    while stack:
        n, boundary, returning = stack.pop()
        if n == 0 or len(boundary) == 3:
            # Forced support points must lie ON the boundary. For three points
```

```
            # use their circumcircle, even when their unconstrained MEC is smaller.
            circle = _circumcircle(*boundary) if len(boundary) == 3 else None
            if circle is None and boundary:
                circle = _enclosing_circle_reference(list(boundary), tolerance)
        elif not returning:
            stack.append((n, boundary, True))
            stack.append((n-1, boundary, False))
        elif circle is None or distance(circle[0], local[n-1]) > circle[1] + tolerance:
            stack.append((n-1, boundary + (local[n-1],), False))
    center = (origin[0]+circle[0][0]*scale, origin[1]+circle[0][1]*scale)
    # Recompute after transforming back: returned circle contains all input points
    # to distance() precision, including any roundoff from the large translation.
    return center, max(distance(center, p) for p in points)


def solve(measurements: list[Measurement], eps: float = EPS) -> dict:
    """Solve Q1's pure angular intersection (no disk/range prior added).

    Nonempty intersections of <180-degree forward wedges are pointed polyhedra,
    so they possess a vertex, even if unbounded. Feasibility follows from vertex
    enumeration; unboundedness follows from a nonzero recession direction.
    Empty diameter is null, unbounded diameter is null with diameter_infinite=true.

    Near-parallel plane pairs are intersected rather than skipped: their
    intersection can be a legitimate vertex of a very long thin region
    (e.g. stations 1cm apart with bearings 1 and 358.99999999999 deg give a
    vertex ~5.7e10 m away). Only exactly-parallel pairs (det == 0) have no
    intersection. The feasibility tolerance scales with the vertex magnitude,
    since absolute residuals grow with coordinates.
    """
    if not isfinite(eps) or eps <= 0:
        raise ValueError("eps必须为有限正数")
    if not measurements:
        return {"status": "unbounded", "vertices": [], "diameter_m": None,
                "diameter_infinite": True, "recession_direction": [1.0, 0.0],
                "message": "没有观测，整个平面均可行"}
    if len(measurements) > 100:
        raise ValueError("参考实现最多接收100个检测点；更大规模应使用半平面交算法")
    ox = sum(m.x for m in measurements) / len(measurements)
    oy = sum(m.y for m in measurements) / len(measurements)
    scale = max(1.0, max(hypot(m.x - ox, m.y - oy) for m in measurements))
    local = [Measurement((m.x - ox) / scale, (m.y - oy) / scale,
                         m.bearing_deg, m.half_angle_deg) for m in measurements]
    planes = [h for m in local for h in halfplanes(m)]

    def inside(p):
```

```
        tol = eps * max(1.0, abs(p[0]), abs(p[1]))
        return all(a * p[0] + b * p[1] <= c + tol for a, b, c in planes)

    feasible_vertices = []
    for (a, b, c), (d, e, f) in combinations(planes, 2):
        det = a * e - b * d
        if det == 0.0:
            continue   # 精确平行：无交点（近平行对的交点是合法远端顶点，保留）
        p = ((c * e - b * f) / det, (a * f - c * d) / det)
        if isfinite(p[0]) and isfinite(p[1]) and inside(p):
            feasible_vertices.append(p)

    if not feasible_vertices:
        return {"status": "empty", "vertices": [], "diameter_m": None,
                "diameter_infinite": False, "tolerance_m": eps * scale,
                "message": "测向角域没有公共点；检查频道、数据及误差界"}

    # A nontrivial 2D recession cone has a ray tangent to an input halfplane.
    # Use a much tighter angular threshold than the positional feasibility epsilon.
    ray = None
    for a, b, _ in planes:
        for direction in ((b, -a), (-b, a)):
            if all(d * direction[0] + e * direction[1] <= 2e-14
                   for d, e, _ in planes):
                ray = direction
                break
        if ray is not None:
            break

    hull = convex_hull(feasible_vertices, eps)

    def world(p):
        return [ox + p[0] * scale, oy + p[1] * scale]

    result = {"status": "unbounded" if ray else
              ("point" if len(hull) == 1 else "segment" if len(hull) == 2 else "polygon"),
              "vertices": [world(p) for p in hull], "tolerance_m": eps * scale,
              "diameter_infinite": ray is not None}
    if ray is not None:
        result.update(diameter_m=None, recession_direction=list(ray),
                      message="存在非零衰退方向，区域无界，直径为无穷大")
        return result

    d, pair = diameter(hull)
    midpoint = ((pair[0][0] + pair[1][0]) / 2, (pair[0][1] + pair[1][1]) / 2)
    distances = [distance(midpoint, p) for p in hull]
```

```
    witness_index = max(range(len(hull)), key=distances.__getitem__)
    max_distance = distances[witness_index]
    center, radius = smallest_enclosing_circle(hull, eps)
    area = abs(sum(cross(hull[i], hull[(i + 1) % len(hull)])
                   for i in range(len(hull)))) / 2
    result.update(diameter_m=d * scale, diameter_pair=[world(p) for p in pair],
                  area_m2=area * scale ** 2,
                  diameter_circle={"center": world(midpoint), "radius_m": d * scale / 2,
                                   "max_vertex_distance_m": max_distance * scale,
                                   "covers": max_distance <= d / 2 + eps * 4,
                                   "coverage_tolerance_m": 4 * eps * scale,
                                   "max_excess_m": (max_distance-d/2) * scale,
                                   "outside_vertex_indices": [i for i, value in enumerate(
    distances)
                                                              if value > d/2 + 4*eps],
                                   "witness_vertex": world(hull[witness_index])
                                       if max_distance > d/2 + 4*eps else None},
                  minimum_enclosing_circle={"center": world(center), "radius_m": radius * scale
    },
                  guaranteed_clearable_at_mec_center=radius * scale <= 20 - eps * scale * 4,
                  clearance_certificate={"model": "angular_intersection_only",
                                         "clear_radius_m": 20.0,
                                         "numerical_margin_m": 4*eps*scale,
                                         "radius_with_margin_m": (radius+4*eps)*scale,
                                         "remaining_margin_m": 20-(radius+4*eps)*scale,
                                         "center": world(center)})
    return result
```

B.2 src/mcm26b/geometry/planning.py：问题 2 安全候选区域

```
"""Distance certificates for Q2; does not change Q1's angular polygon model."""

from math import cos, sin, radians, isfinite
from .localization import Measurement


def first_bearing_candidate_disk(measurement: Measurement,
                                 max_range_m: float = 1500,
                                 guaranteed_range_m: float = 1000) -> dict:
    """Enclose the entire truncated first-bearing sector, then erode its disk.

    For half-angle <=45 degrees the sector MEC has center distance and radius
    a=R/(2*cos(delta)). Intersecting with Omega only shrinks the source set.
    A missing sufficient disk is NOT proof that Q is empty. The certificate
    guarantees distance only: directional transmitters may still be invisible.
    """
```

```
    if any(isinstance(v, bool) or not isinstance(v, (int, float)) or not isfinite(v)
           or v <= 0 for v in (max_range_m, guaranteed_range_m)):
        raise ValueError("接收半径必须为有限正数")
    if measurement.half_angle_deg > 45:
        raise ValueError("此窄角域证书仅支持半角不超过45度；题设为1度")
    delta, theta = radians(measurement.half_angle_deg), radians(measurement.bearing_deg % 360)
    a = max_range_m/(2*cos(delta))
    center = [measurement.x+a*cos(theta), measurement.y+a*sin(theta)]
    available = guaranteed_range_m-a
    return {"source_model": "first_bearing_sector_with_range_upper_bound",
            "sector_enclosing_circle": {"center": center, "radius_m": a},
            "sufficient_candidate_disk": {"center": center, "radius_m": available}
                if available >= 0 else None,
            "candidate_nonempty_certified": available >= 0,
            "distance_only": True,
            "max_range_m": max_range_m, "guaranteed_range_m": guaranteed_range_m}
```