# 第 3 章　单跳位置估计

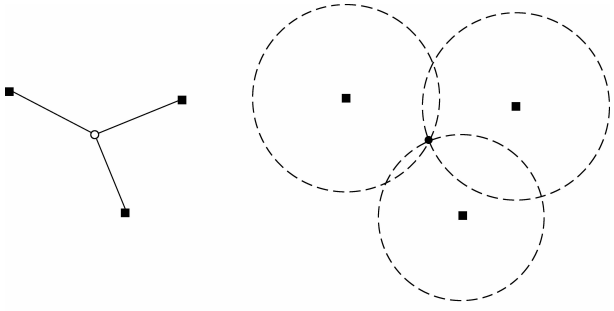
本章讨论如何利用物理测量所获得的信息来计算目标位置。所谓单跳定位是指目标可以直接测量与参考点(已知位置)之间的几何关系,一般的卫星定位系统和室内定位系统都属于单跳定位。与之相对,在多跳定位(网络定位)中,目标只能通过中继点间接测量与参考点之间的几何关系,以无线传感器网络为代表的无线多跳网络定位就属于多跳定位。单跳定位是多跳定位的基础。

单跳位置估计有多种优化方法可以提高估计精度。具体而言,本章专门讨论基于以下几种度量的定位方法:距离、到达时间差(TDoA)、到达角度(AoA)和接收信号强度指纹(RSS fingerprinting)。基于距离的定位方法是通过获得未知节点到多个参考节点的距离来估测未知节点的位置,也就是我们一般所讲的多边测量(multilateration)。TDoA 方法给出了不同参考节点接收同一信号的时间差。给定一个到达时间差 $\Delta t_{ij}$ 和参考节点 $i$ 和 $j$ 的坐标,可以得到以 $i$ 和 $j$ 为焦点的双曲线的一条分支,则未知节点必然位于这条分支上。AoA 方法则给出目标到参考节点的方向角信息。通过结合多个参考节点的 AoA 值,可以得到未知节点的位置估计。基于接收信号强度指纹的方法直接使用 RSS 测量值来进行位置估计。该方法的合理性在于无线信号强度在空间中的分布相对稳定,因此在同一个位置上的 RSS 测量值相对稳定且与其他位置上的 RSS 测量值有所区别。我们把在某个位置上的多个信号的 RSS 形成一个 RSS 向量称为该位置的信号指纹。

## 3.1　基于距离的定位方法

多边测量(multilateration)是一种根据距离定位物体的过程。注意“多边测量”这个词在不同的情形中有不同的意思,在本书中统一表示为基于距离的定位方法。图 3-1 给出了一个三边测量(使用 3 个参考节点的多边测量)的例子,目标(空心点表示)测量其到 3 个参考节点(实心方形表示)的距离。显然,该物体应当位于以每个参考节点为圆心的 3 个圆的交点。只要这 3 个参考点是非线性的(即不在一条直线上),三边测量的结果就是唯一的。

在实际使用中,距离度量难免存在误差,导致 3 个圆不能相交于单个点。这个问题等同于求解超定线性系统的数值解[8]。假设未知节点位于$(x_0, y_0)$,且其到第 $i$ 个参考节点$(x_i, y_i)$的测量距离为 $d_i'(1 \leqslant i \leqslant n$,其中 $n$ 为参考节点的总数)。设



(a) 实际距离 (b) 测距圆

图 3-1 三边测量

$d_i$ 为从该未知节点到第 $i$ 个参考节点的真实距离，则有

$$d_i = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2}$$

因此测量距离和实际距离的差可以表示为 $\rho_i = d_i' - d_i$。当前已有一些方法用于处理测距噪声，其中最小二乘法则通过最小化 $\sum_{i=1}^{n} \rho_i^2$ 的值来确定$(x_0, y_0)$。

具体而言，每一个距离度量确定一个关于未知节点位置的方程式，因此有

$$d_1^2 = (x_1 - x_0)^2 + (y_1 - y_0)^2$$
$$d_2^2 = (x_2 - x_0)^2 + (y_2 - y_0)^2$$
$$\vdots$$
$$d_n^2 = (x_n - x_0)^2 + (y_n - y_0)^2$$

之后的所有方程式都减去第一个方程式，可以得到

$$d_2^2 - d_1^2 = x_2^2 - x_1^2 - 2(x_2 - x_1)x_0 + y_2^2 - y_1^2 - 2(y_2 - y_1)y_0$$
$$d_3^2 - d_1^2 = x_3^2 - x_1^2 - 2(x_3 - x_1)x_0 + y_3^2 - y_1^2 - 2(y_3 - y_1)y_0$$
$$\vdots$$
$$d_n^2 - d_1^2 = x_n^2 - x_1^2 - 2(x_n - x_1)x_0 + y_n^2 - y_1^2 - 2(y_n - y_1)y_0$$

以上方程式可以写成矩阵形式

$$\begin{bmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \\ \vdots & \vdots \\ x_n - x_1 & y_n - y_1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} x_2^2 + y_2^2 - d_2^2 - (x_1^2 + y_1^2 - d_1^2) \\ x_3^2 + y_3^2 - d_3^2 - (x_1^2 + y_1^2 - d_1^2) \\ \vdots \\ x_n^2 + y_n^2 - d_n^2 - (x_1^2 + y_1^2 - d_1^2) \end{bmatrix}$$

该矩阵可进一步简写为

$$\boldsymbol{Hx} = \boldsymbol{b}$$

其中

$$\boldsymbol{H}=\begin{bmatrix} x_2-x_1 & y_2-y_1 \\ x_3-x_1 & y_3-y_1 \\ \vdots & \vdots \\ x_n-x_1 & y_n-y_1 \end{bmatrix} \quad \boldsymbol{x}=\begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$$

$$\boldsymbol{b}=\frac{1}{2}\begin{bmatrix} x_2^2+y_2^2-d_2^2-(x_1^2+y_1^2-d_1^2) \\ x_3^2+y_3^2-d_3^2-(x_1^2+y_1^2-d_1^2) \\ \vdots \\ x_n^2+y_n^2-d_n^2-(x_1^2+y_1^2-d_1^2) \end{bmatrix}$$

这个方程式的最小方差解为

$$\boldsymbol{x}=(\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H})^{-1}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{b}$$

## 3.2　基于到达时间差的定位方法

到达时间差(TDoA)即同一个信号到达不同参考节点的时间差。到达时间差 $\Delta t_{ij}$ 和参考节点 $i$ 和 $j$ 的坐标共同决定了以 $i$ 和 $j$ 为焦点的双曲线的一条分支，未知节点必然位于这条分支上。因此基于到达时间差的定位也可称为双曲线定位。在二维空间 $\mathbb{R}^2$ 中，使用该方法最少需要 3 个参考节点才能唯一确定未知节点的位置，如图 3-2 所示。

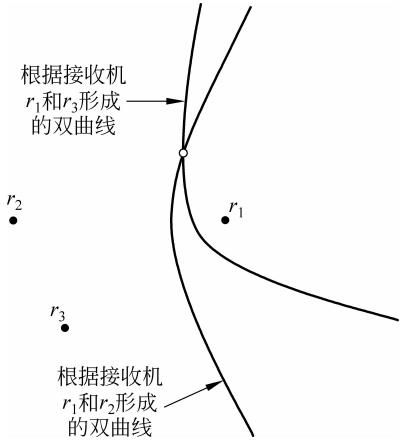



图 3-2　通过到达时间差进行位置计算

我们通过以下方式利用到达时间差的测量值：关于一个参考节点 $i$ 的 TDoA 的值标记为 $\Delta t_i=t_i-t_1$，也就是未知节点发出的信号到达参考节点 $i$ 和参考节点 1 的到达时间差。令 $(x_0,y_0)$ 和 $(x_i,y_i)$ 分别为未知节点和参考节点 $i$ 的坐标，$d_i$ 为该未知节点到参考节点 $i$ 的距离。可以得到如下基本关系式：

$$\begin{aligned} d_1^2&=(x_1-x_0)^2+(y_1-y_0)^2 \\ d_2^2&=(x_2-x_0)^2+(y_2-y_0)^2 \\ &\vdots \\ d_n^2&=(x_n-x_0)^2+(y_n-y_0)^2 \end{aligned}$$

其中，$n$ 为参考节点的总数。令 $\Delta d_i=c\Delta t_i=d_i-d_1$，其中 $c$ 为该未知节点发送信号的传播速度。因此以上方程式可重写为

$$d_1^2 = (x_1 - x_0)^2 (y_1 - y_0)^2$$
$$(d_1 + \Delta d_2)^2 = (x_2 - x_0)^2 (y_2 - y_0)^2$$
$$\vdots$$
$$(d_1 + \Delta d_n)^2 = (x_n - x_0)^2 (y_n - y_0)^2$$

之后所有方程式都减去第一个方程式，可以得到

$$-(x_2 - x_1)x_0 - (y_2 - y_1)y_0 = \Delta d_2 d_1 + \frac{1}{2}(\Delta d_2^2 - x_2^2 - y_2^2 + x_1^2 + y_1^2)$$
$$-(x_3 - x_1)x_0 - (y_3 - y_1)y_0 = \Delta d_3 d_1 + \frac{1}{2}(\Delta d_3^2 - x_3^2 - y_3^2 + x_1^2 + y_1^2)$$
$$\vdots$$
$$-(x_n - x_1)x_0 - (y_n - y_1)y_0 = \Delta d_n d_1 + \frac{1}{2}(\Delta d_n^2 - x_n^2 - y_n^2 + x_1^2 + y_1^2)$$

以矩阵形式重写以上方程式，则有

$$\boldsymbol{Hx} = d_1 \boldsymbol{a} + \boldsymbol{b}$$

其中

$$\boldsymbol{H} = \begin{bmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \\ \vdots & \vdots \\ x_n - x_1 & y_n - y_1 \end{bmatrix} \quad \boldsymbol{x} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \quad \boldsymbol{a} = \begin{bmatrix} -\Delta d_2 \\ -\Delta d_3 \\ \vdots \\ -\Delta d_n \end{bmatrix}$$

$$\boldsymbol{b} = -\frac{1}{2} \begin{bmatrix} \Delta d_2^2 - x_2^2 - y_2^2 + x_1^2 + y_1^2 \\ \Delta d_3^2 - x_3^2 - y_3^2 + x_1^2 + y_1^2 \\ \vdots \\ \Delta d_n^2 - x_n^2 - y_n^2 + x_1^2 + y_1^2 \end{bmatrix}$$

这个方程式的最小方差解为

$$\hat{\boldsymbol{x}} = (\boldsymbol{H}^{\mathrm{T}} \boldsymbol{H})^{-1} \boldsymbol{H}^{\mathrm{T}} (d_1 \boldsymbol{a} + \boldsymbol{b})$$

这个结果中的 $d_1$ 是一个未知量。注意到 $d_1^2 = (x_1 - x_0)^2 + (y_1 - y_0)^2$，将以上中间结果代入这个方程式可以得到关于 $d_1$ 的二次方程式。求解 $d_1$ 并将正实根代回最小方差解可得到 $\boldsymbol{x}$ 的最终解，也就是对未知节点的位置估计。

与一般计算最小方差解的方法不同，研究者们设计了多种方法来处理 TDoA 定位中的非线性方程组[28~30]。例如，准确而鲁棒的泰勒级数法（Taylor-series method）[28]是常用的处理非线性的方法。它是一种避免局部最优的迭代方法，但前提是初始猜测值必须接近真实值。另外，Chan[30]提出一种使用两次最小方差估计的封闭非迭代算法。该算法在 TDoA 测量值误差较小时可以获得很好的效果，而随着误差的增加，其性能快速下降。

## 3.3　基于到达角度的定位方法

如图 3-3 所示，到达角度(AoA)测量值给出了两个节点间的方向角信息。令$(x_0, y_0)$为未知节点的位置，$\alpha_i$ $(1 \leqslant i \leqslant n$，其中 $n$ 为参考节点的总数)为已知的 AoA 度量值。令$(x_i, y_i)$为参考节点 $i$ 的位置，$\theta_i(\boldsymbol{x})$为位于 $\boldsymbol{x}=(x, y)$的节点的方向角。有

$$\tan \theta_i(\boldsymbol{x}) = \frac{y - y_i}{x - x_i}, \quad 1 \leqslant i \leqslant n$$

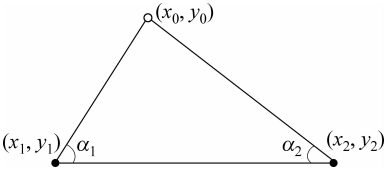



图 3-3　根据 AoA 值进行位置计算

假设参考节点 $i$ 的测量方向角 $\alpha_i$ 被额外的噪声 $\varepsilon_i$ $(1 \leqslant i \leqslant n)$ 干扰。该噪声假定为零均值高斯噪声且其协方差矩阵为 $\sigma_i^2$。于是有

$$\alpha_i = \theta_i(x_0) + \varepsilon_i, \quad 1 \leqslant i \leqslant n$$

当参考节点的构造完全相同且相对于未知节点彼此距离更近时，可以假设各个参考节点方向角度量误差的方差是相等的，即 $\sigma_i^2 = \sigma^2$，$1 \leqslant i \leqslant n$。该未知节点位置的极大似然估计则为

$$\hat{x} = \arg\min \frac{1}{2} \sum_{i=1}^{n} \frac{(\theta_i(\hat{x}) - \alpha_i)^2}{\sigma_i^2}$$

这一非线性最小化问题可通过牛顿-高斯迭代法[31]求解。

另一种方法采用了当 $\varepsilon_i$ 足够小时有 $\varepsilon_i \approx \sin\varepsilon_i$ 这一近似。因此以上的目标函数变换为

$$\frac{1}{2} \sum_{i=1}^{n} \frac{\sin^2(\theta_i(\hat{x}) - \alpha_i)}{\sigma_i^2}$$

由 $d_i = \sqrt{(x_0 - x_i)^2 + (y_0 - y_i)^2}$，且

$$\begin{aligned} \sin(\theta_i(\hat{x}) - \alpha_i) &= \sin\theta_i(\hat{x})\cos\alpha_i - \cos\theta_i(\hat{x})\sin\alpha_i \\ &= \frac{(y_0 - y_i)\cos\alpha_i - (x_0 - x_i)\sin\alpha_i}{d_i} \end{aligned}$$

该目标函数变换为

$$\frac{1}{2} \sum_{i=1}^{n} \frac{[(y_0 - y_i)\cos\alpha_i - (x_0 - x_i)\sin\alpha_i]^2}{\sigma_i^2 d_i^2} = \frac{1}{2}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})^{\mathrm{T}} \boldsymbol{R}^{-1} \boldsymbol{S}^{-1} (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})$$

其中

$$\boldsymbol{A} = \begin{bmatrix} \sin\alpha_1 & -\cos\alpha_1 \\ \vdots & \vdots \\ \sin\alpha_n & -\cos\alpha_n \end{bmatrix}$$

$$\boldsymbol{b} = \begin{bmatrix} x_1 \sin\alpha_1 - y_1 \cos\alpha_1 \\ \vdots \\ x_n \sin\alpha_n - y_n \cos\alpha_n \end{bmatrix}$$

$$\boldsymbol{R} = \mathrm{diag}\{d_1^2, \cdots, d_n^2\}$$

$$\boldsymbol{S} = \mathrm{diag}\{\sigma_1^2, \cdots, \sigma_n^2\}$$

这种方法隐式假设可以获得一个关于 $\boldsymbol{R}$ 的粗略估计。由于该目标函数对 $\boldsymbol{R}$ 的依赖弱，$\boldsymbol{R}$ 的粗略估计对最终解不会有很大影响。在这样的假设下，关于 $\boldsymbol{x}$ 的最小方差解为

$$\hat{\boldsymbol{x}} = (\boldsymbol{A}^{\mathrm{T}}\boldsymbol{R}^{-1}\boldsymbol{S}^{-1}\boldsymbol{A})^{-1}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{R}^{-1}\boldsymbol{S}^{-1}\boldsymbol{b}$$

## 3.4 基于信号指纹的定位方法

基于信号指纹(RSS fingerprinting)的定位方法直接使用接收信号强度(RSS)来进行位置估计。在室内环境中，因为 RSS 容易受到阴影衰落(shadowing)和多径效应(multipath effect)的影响，直接将 RSS 映射为信号传播距离可能引入较大误差。基于信号指纹的方法直接使用 RSS 测量值来进行位置估计。该方法的合理性在于无线信号强度在空间中的分布相对稳定，因此在同一个位置上的 RSS 测量值相对稳定且与其他位置上的 RSS 测量值有所区别。我们把在某个位置上的多个信号的 RSS 形成一个 RSS 向量称为该位置的信号指纹。通过比较信号指纹之间的相似性，可以估计未知节点的位置。根据测量 RSS 的方式，现有的方法可归为如下两类：离线测量与在线测量。

### 3.4.1 离线测量方案

一个典型的离线测量方案称为 RADAR[3]。它通过构建一个信号强度-位置(RSS-location)数据库来定位一个未知节点。RADAR 包含两个步骤：数据库创建和节点定位。在数据库创建阶段，RADAR 收集所有 WiFi 接入点(Access Points)的 RSS 空间分布信息，形成信号强度-位置数据库。具体而言，定位系统管理员现场勘测每个位置的 RSS 指纹。由于 RSS 指纹随着用户身体朝向的变化而显著变化(最大可达 5dBm)，RADAR 将用户朝向考虑进来，并在每一次采样时记录这样的四元组：$(t, x, y, d)$，其中 $t$ 表示测量时间戳，$(x, y)$ 表示测量点的位置，$d$ 为人的朝向。

在构建 RSS-location 数据库完毕之后，RADAR 开始提供定位服务。一个未知节点测量其所在位置的 RSS 指纹，并上传到定位服务器查询该指纹的位置。定位服务器利用最近邻(nearest neighbor)算法将查询指纹与数据库中的指纹进

行匹配，找到与之最相似的指纹，将该指纹对应的位置作为查询指纹的估计位置。

RADAR 仅利用 RSS 信息，无须额外的硬件支持，因此成本低廉、部署简单。但是它容易受到动态环境的影响。环境的变化使得在数据库创建阶段收集的信号指纹与位置的关联关系失效，在这种情况下，重新进行现场勘测并创建数据库是必要的，尽管会带来额外的人力物力耗费。

### 3.4.2　在线测量方案

为了应对环境动态性，LANDMARC[32] 采用在线测量的方式创建信号指纹数据库。LANDMARC 基于射频识别(Radio Frequency Identification，RFID)技术，该技术通过传输电磁波对一个射频兼容的集成电路上进行数据的存储与检索操作。一个 RFID 系统包含若干基本单元，例如 RFID 阅读器和 RFID 标签，其中 RFID 阅读器能从 RFID 标签中读取数据。它们使用预先约定的无线电频率和协议来传输与接收数据。RFID 标签分为被动式和主动式。被动 RFID 标签无须电池供电即可工作，而主动 RFID 标签包含射频收发器和为其供电的电池单元。主动标签的有效通信范围比被动标签大。

LANDMARC 增加了固定位置的参考标签(reference tag)作为定位系统中的参考点(正如日常生活中的地标)来辅助进行位置估计。参考标签的 RSS 共同构成了一个可以在线更新的 RSS-location 数据库。如图 3-4 所示，预先部署的参考标签很好地覆盖了目标区域，并提供了一致的样本数据来定位需要追踪的标签。因此，在参考标签的帮助下，LANDMARC 不需要预先获取 RSS 地图，可以应对环境动态性对离线数据库造成的影响。这样设计的另一个优点是充分利用 RFID 标签代替阅读器来提高定位的精度，因为 RFID 阅读器远比标签昂贵。

追踪标签(tracking tag)的位置计算方法如下。假设有 $n$ 个阅读器、$m$ 个参考标签和 $u$ 个追踪标签。阅读器都配置为连续模式(连续报告在指定范围内的标签)和 1～8 的检测范围(从范围 1～8m 进行扫描并以每个范围 30 秒的周期重复)。定义追踪标签的信号强度向量为 $\boldsymbol{S}=(S_1,S_2,\cdots,S_n)$，其中 $S_i$ 表示阅读器 $i$ 感知到的该追踪标签的信号强度。同样对参考标签而言，设 $\theta=(\theta_1,\theta_2,\cdots,\theta_n)$ 为其对应的信号强度向量。LANDMARC 采用信号强度空间上的欧氏距离作为度量。具体而言，定义

$$E_j=\sqrt{\sum_{i=1}^{n}(\theta_i-S_i)^2},\quad j\in(1,m)$$

为一个追踪标签和参考标签 $r_j$ 在信号强度空间上的欧氏距离。当存在 $m$ 个参考标签时，追踪标签的 $\boldsymbol{E}$ 向量表示为 $\boldsymbol{E}=(E_1,E_2,\cdots,E_m)$。

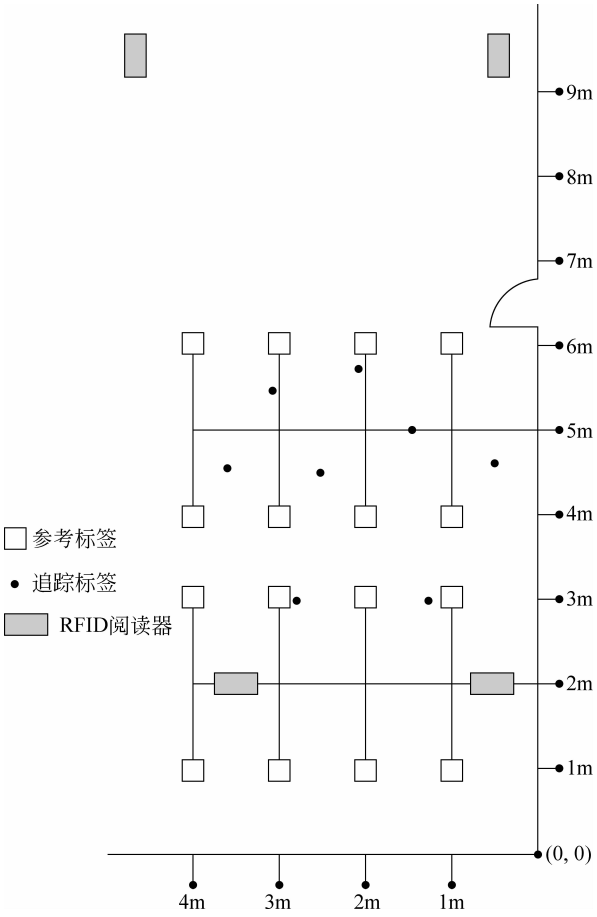



图 3-4 LANDMARC 部署

追踪标签的位置通过对前 $k$ 个最近邻居的位置计算加权平均而得到

$$(x,y)=\sum_{i=1}^{k}w_i(x_i,y_i)$$

其中，$(x_i,y_i)$为第 $i$ 个参考标签的位置，而 $w_i$ 为第 $i$ 个参考标签的加权系数，其计算表达式为

$$w_i=\frac{1/E_i^2}{\sum_{i=1}^{k}1/E_i^2}$$

通常来说，基于接收信号强度特征的定位方案能达到几米的平均误差。例如，RADAR 能有 50% 的概率定位误差不超过 3m，而 LANDMARC 的平均误差为 1m。

## 3.5 小　　结

单跳位置估计的目的是将物理测量数据转化为位置信息。本章具体介绍了如何由距离、到达时间差、到达角度和信号指纹等获得位置信息的方法。在这些方法中，基于距离的定位最容易理解，计算起来也简单，有标准的解析解。而基于到达时间差和到达角度的定位方法在计算上要复杂一些，其物理测量可能需要特殊硬件支持，例如时间同步或天线阵列。基于信号指纹的定位实质上就是一个搜索过程，从指纹数据库中查找出最相似的指纹，其原理容易理解，计算也不复杂。特别是在用信号强度表示信号指纹时，信号指纹易于获取。

在以上的对比中，我们没有提及定位精度，因为定位精度还取决于物理测量的精度以及其他因素。当然，精度归根结底还是取决于应用的需求，根据应用的需求选择位置计算方法和物理测量方法，共同实现定位。就一般情况而言，基于信号指纹的定位方法在精度上较其他 3 种方法低，这主要是因为信号指纹在空间上的区分度比较差。

# 第 4 章　基于测距的多跳定位

随着以传感网和物联网为代表的无线自组织网络技术的发展，定位技术也相应地从基站定位（单跳定位）升级为网络定位（多跳定位）的方式：网络中少量节点（一般称为信标节点或者锚节点）通过手工设置或 GPS 拥有全局位置信息，其余大量节点先获取与邻居节点的相对位置关系，再通过无线网络交换数据相互配合进而获得全局位置。这种定位方式还有合作定位、迭代定位等其他名称，这些名称的共性是都强调当被定位的终端不能与信标直接通信时，需要其他终端的配合确定与信标的相对位置。

与单跳定位相比，多跳定位计算组织形式更复杂，计算量和通信量更大，还会遇到节点密度、锚节点数量要求、误差传播等单跳定位不会遇到的难题。本章以及接下来的两章分别讨论多跳网络中基于测距的定位、基于非测距的定位以及移动连续定位。

## 4.1　计算组织方式

依据算法的执行和组织方式，可以将基于测距的多跳定位方法分为两类：集中式算法和分布式算法。

集中式算法（centralized algorithms）主要在计算资源充足的中央服务器上执行，而网络节点负责物理测量，并将测量结果传回基站进行数据分析。集中式算法突破了单个网络节点运算能力的限制，但其代价是由数据回传引入的大量网络通信。对大多数硬件平台而言，网络通信往往比本地计算更加耗能。

相反，分布式算法（distributed algorithms）在全网执行，通过大规模节点的并行处理和节点间的通信，弥补了由于缺乏集中式计算而导致计算能力不足的缺陷，同时也有效减少了节点和中心服务器通信的高昂开销。与集中式算法一次性使用全部测量数据同时定位全部网络节点的计算方式不同，分布式算法通常每次只利用一个测量数据子集逐步定位网络节点，从而取得和相应的集中式算法近似的定位结果。分布式网络定位方法中有两类重要分支。第一类是基于锚节点（beacon/anchor）的分布式算法，即从锚节点及其邻近节点开始，逐步向周围节点扩展的定位算法。在这类分布式算法中，网络节点通过测量其与若干锚节点之间的距离以确定其位置。在一些算法中，这些新近被定位的节点也可作为锚节点，以辅助后续网络定位过程中其余节点的定位。位置信息通过反复迭代的方式从锚节点逐步扩

散到网络边缘，因此，这类方法可以看作一种自上而下的方式。而第二类分布式算法则采用了一种自下而上的方式，每组节点首先确定其在局部地图中的相对坐标，然后逐步拼接这些局部地图，直至最终实现全网的全局坐标定位。

## 4.2　集中式定位算法

### 4.2.1　多维标度(MDS)

多维标度(Multi-Dimensional Scaling，MDS)[33]来源于计量心理学。假设空间中有 $n$ 个位置未知的点，已知每两点之间的距离，MDS 则利用这些成对距离(pairwise distances)通过余弦定理和线性代数重构出这 $n$ 个点的相对坐标。其时间复杂度为 $O(n^3)$。多维标度分为 3 个步骤：

(1) 生成一个 $n\times n$ 矩阵 $\boldsymbol{M}$，其中处于$(i,j)$的元素表示节点 $i$ 和 $j$ 之间估算的相对距离(可利用 Floyd 全局最短路径算法求得)。

(2) 对矩阵执行经典的度量多维标度算法(Metric-MDS)，确定所有节点的相对坐标。

(3) 根据锚节点的坐标将所有节点的相对坐标变换为全局坐标。

度量多维标度的目标是找到一组多维空间点的配置，使得两点之间的距离与经由某种变换(如线性变化)后的邻近性(proximities)相关联。度量多维标度算法的计算流程如下：

令 $p_{ij}$ 为物体 $i$ 和 $j$ 之间的邻近性测度(proximity measure)，则 $m$ 维空间中的两点 $X_i=(x_{i1},x_{i2},\cdots,x_{im})$ 和 $X_j=(x_{j1},x_{j2},\cdots,x_{jm})$ 的欧氏距离为

$$d_{ij}=\sqrt{\sum_{k=1}^{m}(x_{ik}-x_{jk})^2}$$

若一个几何模型(点的坐标)与给定的邻近度数据(proximity data)$M$ 相匹配，则相应的欧氏距离 $d_{ij}$ 与邻近度 $p_{ij}$ 存在 $d_{ij}=f(p_{ij})$ 的关系。经典的度量多维标度中采用的是线性变化模型，即 $d_{ij}=a+bp_{ij}$。

距离矩阵 $\boldsymbol{D}$ 是在最小二乘意义下最接近邻近性矩阵 $\boldsymbol{P}$ 的一组距离参数。因此，可定义 $\boldsymbol{I}(\boldsymbol{P})=\boldsymbol{D}+\boldsymbol{E}$，其中 $\boldsymbol{I}(\boldsymbol{P})$ 是邻近度的一个线性变换，而 $\boldsymbol{E}$ 是残差(residuals)矩阵。由于距离矩阵 $\boldsymbol{D}$ 是坐标矩阵 $\boldsymbol{X}$ 的函数，因此，经典度量多维标度的目标是计算一组坐标 $\boldsymbol{X}$，在合适的归一化坐标 $\boldsymbol{X}$ 条件下，使得残差 $\boldsymbol{E}$ 的平方和最小。在经典度量多维标度中，矩阵 $\boldsymbol{P}$ 被平移到中心，而坐标 $\boldsymbol{X}$ 可由双中心矩阵(double-centered matrix)$\boldsymbol{P}$ 中通过奇异值分解(SVD)计算得到。对包含了 $n$ 个 $m$ 维点的 $n\times n$ 邻近度矩阵 $\boldsymbol{P}$，有

$$-\frac{1}{2}\left(p_{ij}^2-\frac{1}{n}\sum_{j=1}^{n}p_{ij}^2-\frac{1}{n}\sum_{i=1}^{n}p_{ij}^2+\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}p_{ij}^2\right)=\sum_{k=1}^{m}x_{ik}x_{jk}$$

等式左侧的双中心矩阵(用 $\boldsymbol{B}$ 表示)是对称且正定的，因此对 $\boldsymbol{B}$ 进行奇异值分解可得 $\boldsymbol{B}=\boldsymbol{V}\boldsymbol{A}\boldsymbol{V}$。则坐标矩阵 $\boldsymbol{X}$ 可写为 $\boldsymbol{X}=\boldsymbol{V}\boldsymbol{A}^{1/2}$。

若保留前 $r$ 个最大的特征值以及相应的特征向量($r<m$)，则得到了一组低维空间的解。这意味着，若在上述公式中将对 $k$ 的求和从 1 到 $m$ 改为 1 到 $r$，则得到一组最小二乘意义下的最优低秩近似解。例如，对二维网络，可选择前两个最大的特征值及相应的特征向量来重构最优二维近似解，而可以选择前 3 个最大的特征值及相应的特征向量作为三维网络的最优三维近似解。

对 RSS 数据应用多维标度算法性能优异，当邻节点数量 $n_{\text{local}}$ 大于 12 时，可达到量级为节点测距距离一半的定位精度[34]。然而，多维标度的主要问题是其渐进性能较差，在步骤(1)和步骤(2)中的计算复杂度均为 $O(n^3)$。

此外，经典多维标度方法还存在以下两个主要缺陷。首先，多维标度本质上是一种集中式算法，这极大地制约了多维标度的可扩展性。其次，在形状不规则的网络中，两点之间的最短路径距离与其实际的欧氏距离偏差较大，因此，这种距离估计误差将在定位结果中导致极大的误差。为了解决上述问题，研究者提出了一种基于多维标度的分布式算法，即 MDS-MAP(P)[35]。其主要思想是：对每个节点的直接邻近节点(immediate vicinity)构建一个局部地图，然后将这些局部地图合并为一个全局地图。具体而言，MDS-MAP(P)包括以下 5 个主要步骤。

(1) 设置局部地图的范围 $R_{\text{lm}}$。对每个节点，对在 $R_{\text{lm}}$ 跳之内的邻节点构建其局部地图。$R_{\text{lm}}$ 的选择影响到算法的计算量和定位效果。通常情况下，设置 $R_{\text{lm}}=2$ 即可获得较好的定位效果。计算每个局部地图的复杂度为 $O(k^3)$，其中 $k$ 是平均邻节点数。因此，计算 $n$ 个节点的局部地图的复杂度是 $O(k^3n)$。

(2) 计算每个节点的局部地图。对每个节点执行以下操作：

① 计算在局部地图范围 $R_{\text{lm}}$ 内所有节点对之间的最短路径，用于构建多维标度的距离矩阵。

② 对距离矩阵应用多维标度算法，并保留前 2 个(或 3 个)最大的特征值以及相应的特征向量，用于构建一个二维(或三维)局部地图。

③ 完善局部地图。把用多维标度求得的解作为初始点坐标，对其进行最小二乘化，使得邻近节点之间的距离尽可能与实测值相匹配。

(3) 拼合局部地图。合并操作既可串行进行也可并行进行。首先，随机选取一个节点，并将其局部地图作为核心地图(core map)。然后，通过合并核心地图的邻节点的局部地图逐步扩展核心地图。每次选择与核心地图具有最多共同节点的邻居的局部地图进行合并操作。最终核心地图将覆盖整个网络得到全局地图。若合并操作选择得当，则该步骤的复杂度是 $O(k^3n)$，其中 $k$ 是平均邻节点数，$n$ 是节点数量。

(4) 优化全局地图(可选)。将全局地图中的节点坐标作为初始值，应用最小

二乘最小化使得邻节点之间的距离与实测值尽可能匹配。该步骤的计算复杂度是 $O(n^3)$。

(5) 若有足够多的锚节点(二维网络至少 3 个锚节点,三维网络至少 4 个锚节点),基于锚节点的绝对坐标将全局地图变换为绝对地图。对个 $r$ 个锚节点,该步骤的复杂度为 $O(r^3+n)$。

总而言之,MDS-MAP(P)是利用局部信息计算相对较小的局部地图,而不是利用任意两点间的距离直接构建全局地图。与集中式多维标度算法相比,该算法降低了计算复杂度,且能够处理非凸性的网络拓扑情况,从而提高了定位精度。

### 4.2.2　半定规划(SDP)

半定规划(Semidefinite Programming,SDP)由 Doherty[14] 率先提出。在半定规划算法中,节点间的几何约束被表示为线性矩阵不等式(Linear Matrix Inequalities,LMI)。一旦网络中的所有几何约束条件均用该形式表达,这组 LMI 即可组合成一个单一的半定规划问题,并可解得对每个节点的一个限定区域(bounding region)。半定规划的优势在于其简洁的问题描述、明确的模型表示以及优雅的求解过程。

半定规划的数学表达形式如下:假设有 $m$ 个锚节点 $a_k \in \mathbb{R}^2$,其中 $k=1,\cdots,m$;以及 $n$ 个位置未知的节点 $x_j \in \mathbb{R}^2$,其中 $j=1,\cdots,n$。对于在 $N_e$ 中的两个节点,分别用 $d_{kj}$ 表示锚节点 $a_k$ 与未知节点 $x_j$ 之间的欧氏距离,用 $d_{ij}$ 表示未知节点对 $x_i$ 和 $x_j$ 之间的欧氏距离。对于在 $N_l$ 中的两个节点,锚节点 $a_k$ 与未知节点 $x_j$ 之间的距离下界为 $\underline{r}_{kj}$,未知节点对 $x_i$ 和 $x_j$ 之间的距离下界为 $\underline{r}_{ij}$。而对于在 $N_u$ 中的两个节点,锚节点 $a_k$ 与未知节点 $x_j$ 之间的距离上界为 $\bar{r}_{kj}$,未知节点对 $x_i$ 和 $x_j$ 之间的距离上界为 $\bar{r}_{ij}$。则定位问题即可表述为求解满足以下不等式关系的 $x_j$:

$$\begin{aligned}
&\| x_i - x_j \|^2 = d_{ij}^2, \quad \| a_k - x_j \|^2 = d_{kj}^2, \quad \forall (i,j),(k,j) \in N_e \\
&\| x_i - x_j \|^2 \geqslant \underline{r}_{ij}^2, \quad \| a_k - x_j \|^2 \geqslant \underline{r}_{kj}^2, \quad \forall (i,j),(k,j) \in N_l \\
&\| x_i - x_j \|^2 \leqslant \bar{r}_{ij}^2, \quad \| a_k - x_j \|^2 \leqslant \bar{r}_{kj}^2, \quad \forall (i,j),(k,j) \in N_u
\end{aligned}$$

由于上式中距离及其上下界的测量值通常含有误差,因此要求解使得这些误差之和最小化的 $x_j$:

$$\begin{aligned}
\min \quad & \sum_{i,j\in N_e, i<j} | \| x_i - x_j \|^2 - d_{ij}^2 | + \sum_{k,j\in N_e} | \| a_k - x_j \|^2 - d_{kj}^2 | \\
& + \sum_{i,j\in N_l, i<j} ( \| x_i - x_j \|^2 - \underline{r}_{ij}^2 )_- + \sum_{k,j\in N_l} ( \| a_k - x_j \|^2 - \underline{r}_{kj}^2 )_- \\
& + \sum_{i,j\in N_u, i<j} ( \| x_i - x_j \|^2 - \bar{r}_{ij}^2 )_+ + \sum_{k,j\in N_u} ( \| a_k - x_j \|^2 - \bar{r}_{kj}^2 )_+
\end{aligned}$$

其中,$(u)_-$ 和 $(u)_+$ 分别定义为 $(u)_- = \max\{0,-u\}$ 和 $(u)_+ = \max\{0,u\}$。

令 $\boldsymbol{X}=[x_1 \quad x_2 \quad \cdots \quad x_n]$为待求的 $2\times n$ 维矩阵，则

$\| x_i - x_j \|^2 = \boldsymbol{e}_{ij}^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{e}_{ij}$，　$\| a_i - x_j \|^2 = (a_i;\boldsymbol{e}_j)^{\mathrm{T}}[\boldsymbol{I} \quad \boldsymbol{X}]^{\mathrm{T}}[\boldsymbol{I} \quad \boldsymbol{X}](a_i;\boldsymbol{e}_j)$

其中，$\boldsymbol{e}_{ij}$ 为 $n \times 1$ 维向量，其第 $i$ 个分量为 1，第 $j$ 个分量为 $-1$，其余分量为零。$\boldsymbol{e}_j$ 为除第 $j$ 个分量为 $-1$ 的全零 $n\times 1$ 维向量。记 $\boldsymbol{Y}=\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$。引入松弛变量 $\alpha$ 和 $\beta$，该问题可以改写为一个更宽松的误差最小化问题：

$$
\begin{aligned}
\min \quad & \sum_{i,j\in N_e,i<j}(\alpha_{ij}^{+}+\alpha_{ij}^{-})+\sum_{k,j\in N_e}(\alpha_{kj}^{+}+\alpha_{kj}^{-}) \\
& +\sum_{i,j\in N_l,i<j}\beta_{ij}^{-}+\sum_{k,j\in N_l}\beta_{kj}^{-} \\
& +\sum_{i,j\in N_u,i<j}\beta_{ij}^{+}+\sum_{k,j\in N_u}\beta_{ij}^{+} \\
\text{s.t.} \quad & \boldsymbol{e}_{ij}^{\mathrm{T}}\boldsymbol{Y}\boldsymbol{e}_{ij}-d_{ij}^{2}=\alpha_{ij}^{+}-\alpha_{ij}^{-}, \quad \forall i,j\in N_e, \quad i<j \\
& (a_k;\boldsymbol{e}_j)^{\mathrm{T}}\begin{pmatrix}\boldsymbol{I} & \boldsymbol{X}\\ \boldsymbol{X}^{\mathrm{T}} & \boldsymbol{Y}\end{pmatrix}(a_k;\boldsymbol{e}_j)-d_{kj}^{2}=\alpha_{kj}^{+}-\alpha_{kj}^{-}, \quad \forall k,j\in N_e \\
& \boldsymbol{e}_{ij}^{\mathrm{T}}\boldsymbol{Y}\boldsymbol{e}_{ij}-\overline{r}_{ij}^{\,2}\geqslant-\beta_{ij}^{-}, \quad \forall i,j\in N_l,i<j \\
& (a_k;\boldsymbol{e}_j)^{\mathrm{T}}\begin{pmatrix}\boldsymbol{I} & \boldsymbol{X}\\ \boldsymbol{X}^{\mathrm{T}} & \boldsymbol{Y}\end{pmatrix}(a_k;\boldsymbol{e}_j)-\underline{r}_{kj}^{\,2}\geqslant-\beta_{kj}^{-}, \quad \forall k,j\in N_l \\
& \boldsymbol{e}_{ij}^{\mathrm{T}}\boldsymbol{Y}\boldsymbol{e}_{ij}-\overline{r}_{ij}^{\,2}\leqslant\beta_{ij}^{+}, \quad \forall i,j\in N_u,i<j \\
& (a_k;\boldsymbol{e}_j)^{\mathrm{T}}\begin{pmatrix}\boldsymbol{I} & \boldsymbol{X}\\ \boldsymbol{X}^{\mathrm{T}} & \boldsymbol{Y}\end{pmatrix}(a_k;\boldsymbol{e}_j)-\overline{r}_{kj}^{\,2}\leqslant\beta_{kj}^{+}, \quad \forall k,j\in N_u \\
& \alpha_{ij}^{+},\alpha_{ij}^{-},\alpha_{kj}^{+},\alpha_{kj}^{-},\beta_{ij}^{-},\beta_{kj}^{-},\beta_{ij}^{+},\beta_{kj}^{+}\geqslant 0 \\
& \boldsymbol{Y}=\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}
\end{aligned}
$$

然而上述问题并不是一个凸优化问题。通过把 $\boldsymbol{Y}=\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ 放宽为 $\boldsymbol{Y}\geq\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$，将该问题转化为一个半定规划问题。其中表达式 $\boldsymbol{Y}\geq\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ 等价于

$$
Z:=\begin{pmatrix}\boldsymbol{I} & \boldsymbol{X}\\ \boldsymbol{X}^{\mathrm{T}} & \boldsymbol{Y}\end{pmatrix}\geq 0
$$

然后，该问题即可写为一个标准的半定规划问题：

$$
\begin{aligned}
\min \quad & \sum_{i,j\in N_e,i<j}(\alpha_{ij}^{+}+\alpha_{ij}^{-})+\sum_{k,j\in N_e}(\alpha_{kj}^{+}+\alpha_{kj}^{-}) \\
& +\sum_{i,j\in N_l,i<j}\beta_{ij}^{-}+\sum_{k,j\in N_l}\beta_{kj}^{-} \\
& +\sum_{i,j\in N_u,i<j}\beta_{ij}^{+}+\sum_{k,j\in N_u}\beta_{ij}^{+} \\
\text{s.t.} \quad & (1;0;0)^{\mathrm{T}}\boldsymbol{Z}(1;0;0)=1 \\
& (0;1;0)^{\mathrm{T}}\boldsymbol{Z}(0;1;0)=1 \\
& (1;1;0)^{\mathrm{T}}\boldsymbol{Z}(1;1;0)=2
\end{aligned}
$$

$$
\begin{aligned}
&(0;\boldsymbol{e}_{ij})^{\mathrm{T}}\boldsymbol{Z}(0;\boldsymbol{e}_{ij})-\alpha_{ij}^{+}+\alpha_{ij}^{-}=d_{ij}^{2},\quad \forall i,j\in N_{\mathrm{e}},i<j\\
&(a_k;\boldsymbol{e}_{ij})^{\mathrm{T}}\boldsymbol{Z}(a_k;\boldsymbol{e}_j)-\alpha_{kj}^{+}+\alpha_{kj}^{-}=d_{kj}^{2},\quad \forall k,j\in N_{\mathrm{e}}\\
&(0;\boldsymbol{e}_{ij})^{\mathrm{T}}\boldsymbol{Z}(0;\boldsymbol{e}_{ij})+\beta_{ij}^{-}\geqslant \underline{r}_{ij}^{2},\quad \forall i,j\in N_{\mathrm{l}},i<j\\
&(a_k;\boldsymbol{e}_j)^{\mathrm{T}}\boldsymbol{Z}(a_k;\boldsymbol{e}_j)+\beta_{kj}^{-}\geqslant \underline{r}_{kj}^{2},\quad \forall k,j\in N_{\mathrm{l}}\\
&(0;\boldsymbol{e}_{ij})^{\mathrm{T}}\boldsymbol{Z}(0;\boldsymbol{e}_{ij})-\beta_{ij}^{+}\leqslant \overline{r}_{ij}^{2},\quad \forall i,j\in N_{\mathrm{u}},i<j\\
&(a_k;\boldsymbol{e}_j)^{\mathrm{T}}\boldsymbol{Z}(a_k;\boldsymbol{e}_j)-\beta_{kj}^{+}\leqslant \overline{r}_{kj}^{2},\quad \forall k,j\in N_{\mathrm{u}}\\
&\alpha_{ij}^{+},\alpha_{ij}^{-},\alpha_{kj}^{+},\alpha_{kj}^{-},\beta_{ij}^{-},\beta_{kj}^{-},\beta_{ij}^{+},\beta_{kj}^{+}\geqslant 0\\
&\boldsymbol{Z}\succeq 0
\end{aligned}
$$

若采用集中式方法求解线性或半定规划问题，对于参数为到达角(angle of arrival)的情况而言，其时间复杂度为 $O(k^2)$，而当数据中包含了径向信息(如跳数)时，其时间复杂度则为 $O(k^3)$，其中 $k$ 是一个网络所需的凸约束条件数。因此，半定规划算法较高的复杂度很可能阻碍其在实际问题中的广泛应用。

## 4.3　分布式定位方法

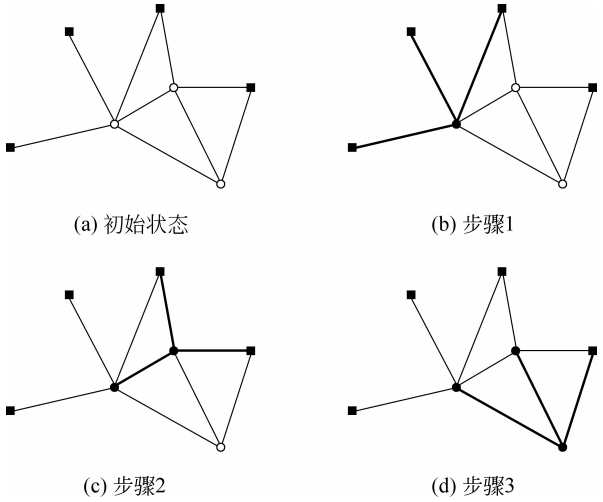
### 4.3.1　基于锚节点的网络定位

**1. 迭代三边定位法**

基于锚节点的定位方法利用了节点与锚节点之间的距离来进行全网定位。未知节点与锚节点之间的距离可以通过基本的距离向量技术(distance-vector technique)[36],[37]估算。由于在该方法中位置信息仍然从锚节点逐步传播到整个网络，因此这种方法也被看作一种自上而下的定位方式。

上述方法的一种变形是间接利用锚节点。初始时，任意一个未知节点试图基于其邻节点的位置通过多边定位(multilateration)或其他定位技术来确定自己的位置。确定其位置后，该节点即可成为一个参考节点，辅助后续定位其他未知节点。这一过程反复执行，直到所有未知节点逐步变为已知节点，从而实现全网定位。图 4-1 示意了这种迭代三边定位(trilateration)过程，其中实心方形代表锚节点，空心圆圈代表未知节点，实心圆点代表位置已知节点。

这种方法的优势是在定位一个节点时仅涉及其局部信息(即邻节点的信息)，因此提高了通信的效率。然而，由于被定位的节点又被当做参考节点来定位其他未知节点，这种方法会引入大量的累积误差，尤其是对距离锚节点较远的网络节点而言，误差累积效应可能会非常严重。因此一些研究工作[38],[39]着手分析了多跳定位方法的误差传播特性，从而尽量控制误差累积。第 7 章将详细讨论误差控制技术。



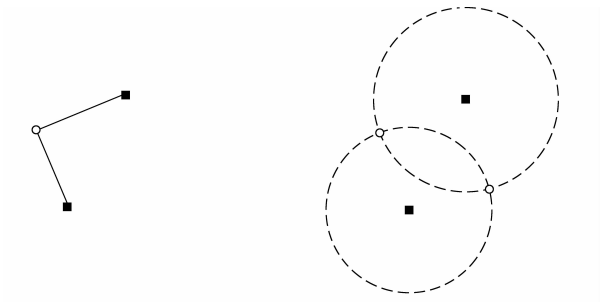
(a) 初始状态 (b) 步骤1

(c) 步骤2 (d) 步骤3

图 4-1 迭代三边定位

**2. 基于双边定位的有限定位法**

实验表明，迭代三边定位法在平均节点连接数超过 10 时才能正确定位网络中的大部分节点[40]。当平均节点连接数低于 8 时，迭代三边定位法在大部分情况下会失效。为了适应稀疏网络的定位需求，Sweeps 算法[41]一定程度上放松了在迭代方法中所需的节点之间的依赖关系。与传统的“唯一定位算法”(unique position computation)不同，Sweeps 算法提出了一种有限定位方法(finite localization)，给出一个目标节点的候选位置集合(candidate positions)，且该候选集中包含目标节点的真实位置。此外，Sweeps 算法运用了一种称为双边定位(bilateration)的方案，仅利用一个未知节点离两个参考节点的距离来确定其候选位置。如图 4-2 所示，利用双边定位为一个未知节点产生了两个候选位置(空心圆圈所示)，其中左边的候选位置是该节点的真实位置。与多边定位法类似，被有限定位了的节点



(a) 真实位置信息 (b) 双边定位法生成两个可能的候选位置

图 4-2 双边定位

(finitely localized node)称为 Swept 节点,也可作为后续过程中定位其他节点时的参考节点。唯一不同的是,Sweeps 算法在计算目标节点的位置时需要枚举所有 Swept 节点的候选位置。此外,在每次双边定位之后,Sweeps 算法需要检查候选位置之间的一致性,并删除互相不兼容的候选位置。在这种机制下,Sweeps 算法能够定位一个网络中大部分理论上可定位的节点。然而,Sweeps 算法在最坏情况下的计算复杂度随节点数量指数增长。

下面通过一个典型的网络拓扑详细介绍 Sweeps 算法的执行流程,如图 4-3(a)所示,其中实心圆圈表示参考节点,空心圆圈表示未知节点。显然,利用传统的三边定位法不能定位图中的任一未知节点。相反,Sweeps 算法能根据未知节点 $v_4$ 与 Swept 节点 $v_1$ 和 $v_3$ 之间的距离测量值生成两个 $v_4$ 的候选位置,分别如图 4-3(b1)和图 4-3(b2)所示。由于 $v_4$ 被有限定位,$v_4$ 也成为了 Swept 节点。然后,由于节点 $v_5$ 已知其到两个 Swept 节点 $v_1$ 和 $v_4$ 的距离,$v_5$ 也可被有限定位。注意到在计算节点 $v_5$ 的候选位置时,必须枚举节点 $v_4$ 的所有候选位置。由于节点 $v_4$ 有 2 个候选位置,则节点 $v_5$ 有 4 个候选位置。图 4-3 中的箭头显示

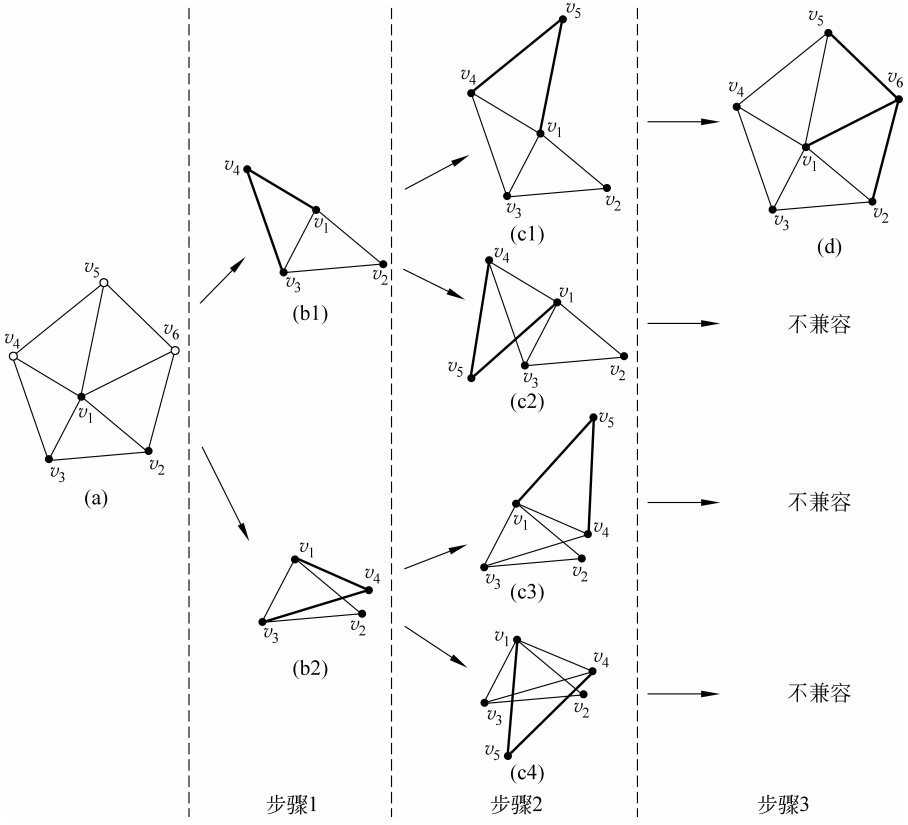



图 4-3　Sweeps 算法执行过程

了候选位置之间的依赖关系。如图 4-3(c1)～(c4)所示，节点 $v_5$ 的所有候选位置与参考节点的距离都要与实际测量值相一致。最后，通过验证节点 $v_6$ 与 3 个 Swept 节点 $v_1$、$v_2$ 和 $v_5$ 的距离测量值进行一致性检查，可以剔除节点 $v_5$ 所有不兼容的候选情况。同时节点 $v_4$ 相应的候选位置(根据依赖树确定)也被剔除，从而正确地定位网络中的所有节点。

Sweeps 算法可以定位一种称为可双边定位网络(localizable bilateration network)的网络，其定义如下：称一个网络有一个关于锚节点 $v_1$、$v_2$ 和 $v_3$ 的双边定位序(bilateration ordering)，若其节点可以被排序为 $v_1$，$v_2$，…，$v_n$，使得 $v_1$ 和 $v_2$ 相邻，并且任意节点 $v_i$(其中 $i>2$)至少与两个节点 $v_j$(其中 $j<i$)相邻。若一个网络有一个双边定位序，则该网络称为双边定位网络。显然，图 4-3(a)所示的轮图网络(wheel network)是一种特殊的双边定位网络。

从上述轮图网络的例子可以看出，在最坏情况下候选位置的数量为 $O(2^n)$，其中 $n$ 是网络节点总数。因此，Sweeps 算法的缺点之一就是其计算复杂度高。Sweeps 算法引入了两种机制来缓解这一问题。首先，Sweeps 算法在每次双边定位后就立即进行一次一致性检查，以减少候选位置的数量。其次，Sweeps 算法通过选择特定的扫描定位顺序，称为 Shell Sweeps，来减缓候选位置集合的增长。Shell Sweeps 是一种广度优先的 Sweeps 算法。在每一阶段，若已知某节点到至少两个 Swept 节点的距离测量值，则将该节点排在定位顺序的较前位置。然而，这些机制都无法降低该算法的最坏计算复杂度，对轮图网络而言，其计算复杂度仍为 $O(2^n)$。虽然 Sweeps 算法在最坏情况下是非多项式的复杂度，但其在随机部署的网络中有较合理的平均算法执行开销。

除了有较高的计算复杂度外，Sweeps 算法也会受到距离测量误差的影响。当测量值有错误时，一致性检查可能会剔除所有候选位置。此外，每一步的位置误差累积下来会导致若干步后定位结果严重偏差。虽然有扩展版本的 Sweeps 算法来处理距离测量值含有误差的情况，但这种修订后的版本要求网络符合一种几何约束，即单位圆盘图(Unit Disk Graph，UDG)模型，并且无法保证其定位结果的正确性。因此，Sweeps 算法的误差控制仍然是一个需要解决的问题。

### 4.3.2 坐标系拼接

#### 1. 局部地图拼接

坐标系拼接也是一种定位方法，在近期受到了许多关注[36],[40],[42]。坐标系拼接是一种自下而上的方式，通过逐步合并局部地图最终实现全局坐标中的全网定位，如图 4-4 所示。

坐标系拼接的工作原理如下：

(1) 将网络分割为相互重叠的子区域，通常每个子区域仅为一个单一的节点

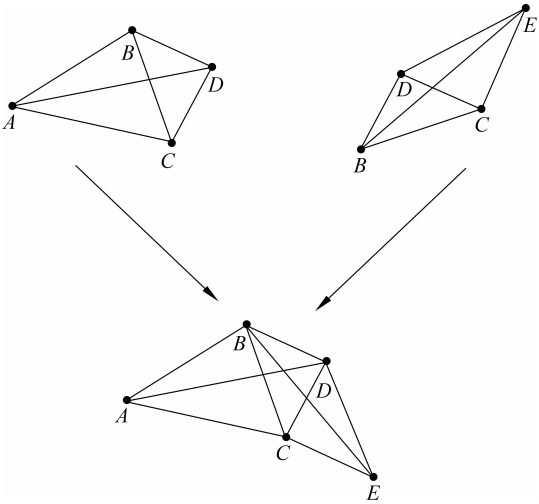



图 4-4　坐标系拼接

及其单跳邻节点。

(2) 对于每个子区域,计算其局部地图,这实质上是把子区域的节点嵌入到一个相对坐标系中。

(3) 最后,利用一个坐标系登记流程(registration procedure)将子区域合并。坐标系登记是通过寻找一个刚性变换把一个坐标系中的点映射到另一个坐标系中。这样,步骤(3)把所有子区域放置到一个统一的全局坐标系中。许多实现这一步的算法都只是次优的,因为登记坐标系需要采用一种闭式、快速和基于最小二乘的优化算法。

Moore 等[62]提出了一种生成鲁棒的局部地图的方法。该方法没有使用 3 个任意节点,而是利用了鲁棒四边形(robust quads)来定义局部地图。如图 4-5 所示,一个鲁棒四边形含有 4 个子三角形($\triangle ABC$,$\triangle ADC$,$\triangle ABD$,$\triangle BCD$)组成,其中:

$$b \times \sin^2\theta > d_{\min}$$

其中,$b$ 是最短边的长度,$\theta$ 是最小的夹角,$d_{\min}$是根据测量误差水平预先设定的参数。这种方法的想法是,一个鲁棒四边形的顶点相互之间可以被正确放置(即没有翻转)。Moore 等证明了在测量误差服从零均值高斯分布的情况下,可以通过设

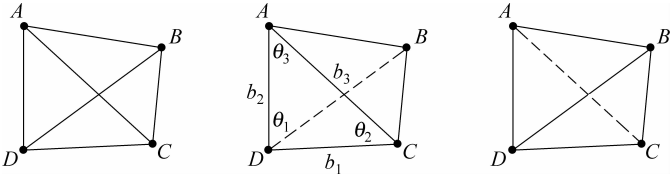



图 4-5　鲁棒四边形

置 $d_{\min}$ 来约束一个鲁棒四边形发生内部翻转的概率，从而可以滤除有过多不确定位置的四边形。过滤的程度由测距误差的水平而定。但是，坐标系拼接会受到局部地图拼接所产生的误差传播的影响。Moore 等计算了他们的算法建立正确的局部地图的概率，并给出了局部地图位置发生错误的下界。此外，他们的算法在许多情形下无法定位孤立节点，因为这些孤立点要么无法被加入到局部地图，要么它们所属的局部地图未能充分与周围的局部地图重叠。Moore 等声称这是可以接受的，因为孤立点最有可能包含很大的错误。此外，对于许多应用而言，缺少某一组已知节点的定位信息比获取一组未知节点的错误信息要更好。

坐标系拼接技术颇引人注目。因为它们本质上是分布式的。此外，由于子区域和局部地图的形成是一个网络中经常出现的情形，因此拼接过程可以很容易地以一种自组织的方式进行。

**2. 组件拼接**

一种更加一般化的坐标系拼接是基于组件的定位方法（component based localization）[43]。一个组件（component）是一组具有刚性结构的节点（每个节点在局部坐标系中拥有有限个候选位置）。把全局刚性组件（每个节点在局部坐标系中都有唯一的位置）作为基本单元，基于组件的定位算法，利用组件之间的距离测量值和锚节点的距离测量值来定位组件。

如图 4-6 所示，组件 A 和 B 之间的 3 个距离测量值约束了它们的相对几何关系。虽然 A 和 B 均分别与两个锚节点相邻，但从每个单一节点的角度看，由于没有节点有足够的邻居锚节点（不超过 2），因此 A 和 B 均不能被有限定位。而传统的基于局部地图的定位方法也无法定位这种网络，因为局部地图（即组件 A 和 B）不包含锚节点用于坐标系转换。然而，从组件的角度看，组件 A 和组件 B 可被合并为一个组件，从而可利用与 4 个锚节点的距离值对其进行定位。最后，两个组件中的所有节点都可被定位。

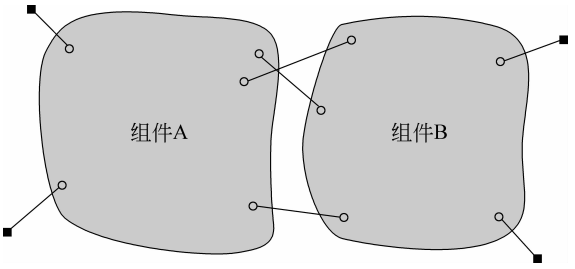



图 4-6 基于组件的定位

组件及其相关概念的正式定义如下：对一个给定的网络，构造距离图 $G=(V, E)$，其中图的顶点表示网络中的节点，若节点 $i$、$j$ 可以测量它们之间的相互距离，