二维平面上的一种线性 SLAM 后端优化算法

First A. Author*. Second B. Author, Jr.**
Third C. Author***

**National Institute of Standards and Technology, Boulder, CO 80305 USA (Tel: 303-555-5555; e-mail: author@ boulder.nist.gov).*
***Colorado State University, Fort Collins, CO 80523 USA (e-mail: author@lamar. colostate.edu)*
**** Electrical Engineering Department, Seoul National University, Seoul, Korea, (e-mail: author@snu.ac.kr)}*

Abstract: xx.

Keywords: Registration, Scan-matching, Lidar, Transformation.

1. INTRODUCTION

results.

2. 构建线性方程

2.1 使用位姿变换构建线性方程

SLAM 算法中可以使用多种方式获得两个关键帧之间的位姿变换（二维情况下，旋转平移矩阵$T \in R^{3\times3}$）；下面表格列出使用相机传感器得到两个关键帧之间的不同类型特征点，可以使用对应的算法计算位姿变换；

表 1，相机的位姿求解算法

特征点类型	位姿变换算法
2d-2d	对极几何
2d-2d（平面）	
2d-3d	DLT算法、EPnP算法
3d-3d	icp算法

如果使用激光雷达传感器获得两帧点云，那么可以使用多种点云配准算法得到位姿变换；此外我们也可以通过 IMU 传感器、GPS 传感器、轮速计和回环检测等方法得到两帧之间的位姿变换；

不管使用什么类型传感器，我们得到的初始信息都是两个关键帧（节点）之间的位姿变换；后端优化的目的是使用这些位姿变换信息估计出精确的全局地图；下面我们使用节点$i \in V$来指代关键帧，每个节点 i 都有自己的局部坐标系Σ_i；节点 i 的局部坐标系Σ_i与全局坐标系Σ_g之间存在未知角度α_i；设节点 i 在二维平面上的坐标为$p_i = [x_i, y_i]^T$；然后使用无向图的边$(i,j) \in E$表示节点 i 和节点 j 之间的长度；那么所有节点和不同节点之间的长度约束就可以使用无向图$G = [V, E]$描述；

我们使用传感器获得的初始信息是任意两个节点（节点 i、节点 j）之间的位姿变换T_{ij}；设所有位姿变换T_{ij}的集合为$E_T = \{T_{ij}, i, j \in V\}$；我们可以定义另外一个位姿约束的图$G_T = [V, E_T]$，其中 V 表示所有节点（关键帧）的集合，$E_T$表示所有位姿变换的集合；长度约束的图$G = [V, E]$可以由位姿约束的图$G_T = \{V, E_T\}$得到，大致方式如下；任意三个节点$i, j, k \in V$之间如果存在两个位姿变换$T_{ij}, T_{ik}$，那么三角形 ijk 的大小形状是确定的，我们可以得到三角形 ijk 的三边长度，作为图$G = [V, E]$中边长集合 E 的元素；其中长度约束的$G = [V, E]$用于计算节点的位置，位姿约束的图$G_T = [V, E_T]$用于离群点去除、局部优化等部分；

下面我们使用线性方程来表示平面上三个节点之间的位姿变换；该方式与论文[1]和论文[2]中相同；设节点 i 有两个邻居节点，分别是节点 j 和节点 k；我们通过 SLAM 算法的前端模块得到从节点 i 到节点 j 的位姿变换为$T_{ji} \in R^{3\times3}$，从节点 i 到节点 k 的位姿变换是$T_{ik} \in R^{3\times3}$；

$$T_{ji} = \begin{bmatrix} R_{ji} & t_{ji} \\ 1 & 0 \end{bmatrix} \in R^{3\times3}$$

$$R_{ji} = \begin{bmatrix} \cos(\theta_{ji}) & -\sin(\theta_{ji}) \\ \sin(\theta_{ji}) & \cos(\theta_{ji}) \end{bmatrix}$$

$$t_{ji} = [x_{ji}, y_{ji}]^T$$

$$T_{ik} = \begin{bmatrix} R_{ik} & t_{ik} \\ 1 & 0 \end{bmatrix} \in R^{3\times3}$$

$$R_{ik} = \begin{bmatrix} \cos(\theta_{ik}) & -\sin(\theta_{ik}) \\ \sin(\theta_{ik}) & \cos(\theta_{ik}) \end{bmatrix}$$

$$t_{ik} = [x_{ik}, y_{ik}]^T$$

从几何上来说，节点 i、节点 j 和节点 k 之间的位姿变换约束的是三角形 ijk 的形状；因为局部坐标系$\Sigma_i, \Sigma_j, \Sigma_k$与全局坐标系$\Sigma_g$之间的角度未知，三角形 ijk 可以整体在二维空间中旋转平移；因为有T_{ji}和T_{ik}的约束，三角形 ijk 的形状不会发生变化；因此T_{ji}和T_{ik}的约束可以转化为全等

三角形约束，比如下面的这组条件$\rho_{ij} = \left|\left|p_i - p_j\right|\right|$、$\rho_{ik} = \left|\left|p_i - p_k\right|\right|$和$\theta_{jik}$；

已知节点 i 到节点 k 的位姿变换是T_{ik}，其中平移量是$t_{ik} = [x_{ik}, y_{ik}]^T$，因此$\rho_{ij}$如下；

$$\rho_{ij} = \left|\left|p_i - p_k\right|\right| = \left|\left|t_{ik}\right|\right| = \sqrt{x_{ik}^2 + y_{ik}^2}$$

已知节点 i 到节点 j 的位姿变换为T_{ji}，那么节点 j 到节点 i 的位姿变换是$T_{ji} = T_{ji}^{-1} \in R^{3\times3}$；我们设$t_{ij} = \left[x_{ij}, y_{ij}\right]^T = \left[T_{ji}[1,3], T_{ji}[1,2]\right]^T$；那么$\rho_{ik}$如下；

$$\rho_{ik} = \left|\left|p_i - p_j\right|\right| = \left|\left|t_{ij}\right|\right| = \sqrt{x_{ij}^2 + y_{ij}^2}$$

节点 i 的局部坐标系Σ_i中，节点 j 的相对坐标是t_{ij}，节点 k 的相对坐标是t_{ik}；那么θ_{ijk}如下；

$$\cos(\theta_{jik}) = \frac{t_{ij}t_{ik}}{\left|\left|t_{ij}\right|\right|\left|\left|t_{ik}\right|\right|}$$

$$= \frac{x_{ij}x_{ik} + y_{ij}y_{ik}}{\sqrt{(x_{ij}^2 + y_{ij}^2)(x_{ik}^2 + y_{ik}^2)}}$$

$$\theta_{jik} = \arccos\left(\frac{t_{ij}t_{ik}}{\left|\left|t_{ij}\right|\right|\left|\left|t_{ik}\right|\right|}\right)$$

我们将节点 i、节点 j 和节点 k 之间的位姿变换约束T_{ji}, T_{ik}转化为全等三角形约束$\rho_{ij}, \rho_{ik}, \theta_{jik}$；为了使用线性方程表示节点之间的位姿约束，我们将全等三角形约束进行放松，变成相似三角形约束$\rho_{jik} = \rho_{ik}/\rho_{ij}, \theta_{jik}$；这样我们只能保证三角形 ijk 的形状不变，但是三角形整体可以进行缩放；此时可以使用如下线性方程组表示这组相似三角形约束；

$$p_k - p_i = \rho_{jik}R_{jik}(p_j - p_i)$$

$$R_{jik} = \begin{bmatrix}\cos(\theta_{jik}) & -\sin(\theta_{jik}) \\ \sin(\theta_{jik}) & \cos(\theta_{jik})\end{bmatrix}$$

其中，p_i, p_j和 pk 分别是节点 i、节点 j 和节点 k 的坐标，属于需要约束的变量；$R_{jik}(p_j - p_i)$表示对向量 ij 旋转θ_{jik}角度，得到与向量 ik 同方向的向量；然后$R_{jik}(p_j - p_i)$与向量 ik 之间存在$\rho_{jik} = \rho_{ik}/\rho_{ij}$的比例；此处$\rho_{jik}$和矩阵$R_{jik}$都可以使用位姿变换$T_{ji}, T_{ik}$计算得到，属于已知的常数；该线性方程组的形式就是论文[1]中对平面上节点之间约束的方程；在图$G = [V, E]$中，任意三个节点（相邻节点或者不相邻节点）之间只要存在 2 个位姿变换，那么这三个节点组成的三角形就是确定的；我们可以使用线性方程组来表示这三个节点的相似三角形约束；

在论文[2]中使用复数的线性方程给出了上述方程的等效形式；设节点$i \in V$的复数坐标为$p_i^c = x_i + y_i i$，其中x_i是节点 i 坐标的 x 轴分量，y_i是节点 i 坐标的 y 轴分量；那么上述线性方程组可以使用如下的复数线性方程表示；

$$p_k^c - p_i^c = \omega_{jik}(p_j^c - p_i^c)$$

$$\omega_{jik} = \rho_{jik}e^{i\theta_{jik}}$$

我们设p_{ik}^c是向量 ik 的复数坐标形式，θ_{ik}是向量 ik 在局部坐标系Σ_i中的角度；设p_{ij}^c是向量 ij 的复数坐标形式，θ_{ij}是向量 ij 在局部坐标系Σ_i中的角度；

$$p_{ik}^c = p_k^c - p_i^c = \rho_{ik}e^{i\theta_{ik}}$$

$$p_{ij}^c = p_j^c - p_i^c = \rho_{ij}e^{i\theta_{ij}}$$

我们将p_{ik}^c和p_{ij}^c替换上述方程中的p_k^c, p_i^c和p_j^c，得到如下方程；

$$\rho_{ik}e^{i\theta_{ik}} = \omega_{jik}\rho_{ij}e^{i\theta_{ij}}$$

$$\rho_{ik}e^{i\theta_{ik}} = \rho_{jik}\rho_{ij}e^{i\theta_{ij}+\theta_{jik}}$$

上述方程可以转化为模长部分的约束和角度部分的约束如下；

$$\rho_{jik} = \rho_{ik}/\rho_{ij}$$

$$\theta_{ij} + \theta_{jik} = \theta_{ik}$$

这些约束刚好与节点 i、节点 j 和节点 k 的相似三角形约束相同；因此上述复数的线性方程同样可以等效的表示三角形 jik 的形状约束；无论使用线性方程还是复数的线性方程，都可以等效的表示三角形 jik 的约束；那么在后续文本中，我们使用复数的线性方程来表示三个节点之间的位姿约束；因为复数的线性方程相对来说稍微简洁，只有一个方程；那么任意节点 i 的坐标使用复数$p_i^c = x_i + y_i i$表示；

2.2 存在多个邻居节点的情况

如果图$G = [V, E]$中所有节点成一条直线分布（比如$1 - 2 - 3 - \ldots - n$），每个节点刚好只有 2 个邻居节点，那么刚好可以使用线性方程组依次表示任意三个相邻点之间的相似三角形约束；但是实际上图$G = [V, E]$中节点之间可能存在更加一般的情况，节点 i 存在多个邻居节点；

设节点 i 的邻居节点为$j \in N_i$；如果节点 i 需要和任意两个邻居节点$j_1 \in N_i$和$j_2 \in N_i$组成一个线性方程组约束，那么节点 i 对应的线性方程组约束个数为$C_2^{|N_i|}$；因此我们可以使用一种更加简洁的方式来选择不同的邻居节点$j \in N_i$与节点 i 构成线性方程组约束，而不是与任意两个邻居节点构建约束；这种选择方式需要保证构建的约束数量尽量少，但是同时尽量的公平的用到每个邻居节点$j \in N_i$；

如果节点 i 存在 2 个邻居节点，分别是节点 j 和节点 k；那么我们可以使用节点 i 与节点 j 之间的位姿变换T_{ij}和节点 i 与节点 k 之间的位姿变换T_{ik}构建一个线性方程；

$$p_k^c - p_i^c = \omega_{jik}(p_j^c - p_i^c)$$

如果节点 i 存在多个邻居节点$j \in N_i$，处理方式如下；在 SLAM 算法中，节点表示每个关键帧；然后关键帧之间存在先后关系，并且从第 1 个关键帧到第 n 个关键帧构成一条连线（$1 - 2 - 3 - \ldots - n$）；节点 i 的多个邻居节点$j \in N_i$中，有两个节点分别是节点 i 的前驱节点j_{prev}和

后继节点j_{next}（可能还没有）；那么我们使用节点 i 的前驱节点j_{prev}构建一个虚拟节点v_i，该节点的坐标是p_{vi}^c；

$$p_{vi}^c - p_i^c = (p_{prev}^c - p_i^c)/\rho_{ij}^{prev}$$

其中p_{prev}^c是节点 i 的前驱节点j_{prev}的复数坐标；ρ_{ij}是节点 i 和节点j_{prev}之间的长度，我们使用ρ_{ij}^{prev}对向量 ij 归一化；因此虚拟节点$v_i \in N_i$的方向与向量ij_{prev}相同，并且模长$||p_{vi} - p_i|| = 1$；设节点 i 到节点j_{prev}的位姿变换是$T_{ij}^{prev} \in R^{3\times3}$，其中平移向量是$t_{ij}^{prev} = \left[x_{ij}^{prev}, y_{ij}^{prev}\right]^T$；那么长度$\rho_{ij}^{prev}$如下；

$$\rho_{ij}^{prev} = \left|\left|t_{ij}^{prev}\right|\right| = \sqrt{x_{ij}^{prev2} + y_{ij}^{prev2}}$$

设节点 i 到其他邻居节点 j 的位姿变换矩阵是T_{ij}，其中平移向量是$t_{ij} = \left[x_{ij}, y_{ij}\right]^T$；在设定虚拟节点$p_{vi}$之后，将节点 i 的所有邻居节点$j \in N_i \text{ and } j \neq j_{prev}$都依次构建“虚拟节点$p_{vi}$-节点 i-节点 j”的线性方程约束如下；对于虚拟节点p_{vi}和邻居节点j_{prev}，使用公式[x]作为线性方程；

$$p_j^c - p_i^c = \omega_{vij}(p_{vi}^c - p_i^c), j \in N_i \text{ and } j \neq j_{prev}$$

$$\omega_{vij} = \rho_{vij}e^{i\theta_{vij}}$$

其中常数$\rho_{vij}, \theta_{vij}, j \in N_i \text{ and } j \neq j_{prev}$使用如下公式计算；

$$\rho_{vij} = \rho_{ij}/\rho_{iv}$$

$$\rho_{ij} = \left|\left|t_{ij}\right|\right| = \sqrt{x_{ij}^2 + y_{ij}^2}$$

$$\rho_{iv} = \left|\left|t_{iv}\right|\right| = \sqrt{x_{iv}^2 + y_{iv}^2}$$

$$\cos(\theta_{vij}) = \frac{t_{ij}t_{iv}}{\left|\left|t_{ij}\right|\right|\left|\left|t_{iv}\right|\right|}$$

$$= \frac{x_{ij}x_{iv} + y_{ij}y_{iv}}{\sqrt{(x_{ij}^2 + y_{ij}^2)(x_{iv}^2 + y_{iv}^2)}}$$

$$\theta_{vij} = \arccos\left(\frac{t_{ij}t_{iv}}{\left|\left|t_{ij}\right|\right|\left|\left|t_{iv}\right|\right|}\right)$$

已知节点 i 的邻居节点为$j \in N_i$，我们采用每个邻居节点 j 依次与虚拟节点v_i构建线性方程，使得各节点之间的互相影响更加公平；构建的线性方程组最后都需要使用最小二乘法计算，那么每个线性方程都对应一个误差项；此时如果某个邻居节点$j_1 \in N_i$的位姿变换发生变化，那么这个变化量需要通过对应的误差项首先影响虚拟节点v_i，然后再通过虚拟节点v_i与其他邻居节点$j \in N_i$的误差项均匀的影响其他邻居节点的位置；如果我们采用每个邻居节点$j \in N_i \text{ and } j \neq j_{prev}$依次与邻居节点$j_{prev}$构建线性方程；那么节点$j_{prev}$就显得比较特殊，与其他邻居节点之间的误差传递有区别；

在 SLAM 算法中，如果某个节点 i 的邻居节点为 2，那么不采用虚拟节点；如果某个节点的邻居节点大于 2，采用虚拟节点；这样的方式会使得线性方程组的形式不确定，从而增加处理的复杂性；所以我们对所有节点都添加虚拟节点，具体如下；节点 i 的虚拟节点v_i的作用是用于和节点 i 的邻居节点$j \in N_i$构建约束；然后节点 i 的虚拟节点通常使用节点 i 的前驱节点$j_{prev} \in N_i$构建，公式为[x]；在 SLAM 算法中，我们通过关键帧获得一组连续的节点（$1-2-3-\ldots-n$）；对于第 1 个节点，没有前驱节点，只有后继节点（节点 2）；所以对于第 1 个节点，我们采用后继节点（节点 2）构建虚拟节点的坐标p_{v1}^c，具体公式如下；

$$p_{v1}^c - p_1^c = (p_2^c - p_1^c)/\rho_{12}$$

我们设图$G = [V, E]$中共有 n 个节点，n 个节点对应的位置向量是$p = [p_1^c, p_{v1}^c, \ldots, p_n^c, p_{vn}^c]^T \in C^{2n}$；位置向量 p 中每个节点 i 的坐标$p_i^c$都有对应的虚拟节点坐标$p_{vi}^c$；那么我们可以将上述所有节点线性方程组合，得到如下的线性方程组；

$$Lp = 0$$

其中矩阵 L 是所有线性方程组合对应的矩阵；在论文[2]中，节点 i 的所有邻居节点$j \in N_i$构建的线性方程都组成一个线性方程，每个节点对应一个线性方程，那么矩阵 L 是方阵；该文献中这样处理的目的是为了实现每个节点的分布式计算；但是在 SLAM 问题中，节点代表关键帧，并不是某个机器人；所以我们因为分布式计算的需求，从而舍弃节点 i 与邻居节点$j \in N_i$之间的完整约束关系；并且该论文中，节点 i 的邻居节点$j \in N_i$的数量是$|N_i|$，那么节点 i 对应的线性方程个数是$|N_i|$；

2.3 节点重合的情况

在图$G = [V, E]$中，如果两个节点之间距离非常近（接近重合），会使得线性方程约束的误差增加；这种情况通常发生在机器人只做纯旋转没有平移的情况；下面我们给出节点重合情况的具体例子；如下图所示，节点$1-4$成一条线排列；其中节点 2 和节点 3 之间的距离非常近；我们可以对“节点 1-节点 2-节点 3”的相似三角形使用线性方程表示；但我们同样也可以等效的使用原始的相似三角形约束表示为$\rho_{123} = \rho_{23}/\rho_{21}$（非常小数值）；对于“节点 2-节点 3-节点 4”的相似三角形可以使用线性方程表示，同样可以使用等效的相似三角形约束表示为$\rho_{234} = \rho_{34}/\rho_{32}$（非常大数值）；

现在假设我们已知节点 1 到节点 2 的长度为ρ_{12}，需要计算节点 3 到节点 4 的长度ρ_{34}；那么使用相似三角形约束计算如下；

$$\rho_{23} = \rho_{123}\rho_{12}$$

$$\rho_{34} = \rho_{234}\rho_{23}$$

在这个计算过程中，因为参数ρ_{123}非常小，计算得到的ρ_{23}非常小，会噪声舍入误差；然后参数ρ_{234}非常大，会将舍入误差放大到ρ_{34}；因此如果两个节点重合，那么会使得计算过程中出现舍入误差；此外当两个节点之间距离很近时，这两个节点之间距离的测量会因为传感器的

精度限制出现误差；这个测量误差会随着参数ρ_{123}（非常小）和参数ρ_{234}（非常大）放大；

对此我们给出的处理方案如下；设节点 i 的两个邻居节点是节点 j 和节点 k；然后我们假设节点 i 和节点 k 之间的距离非常近；

$$p_k - p_i = \omega_{jik}(p_j - p_i)$$

$$\omega_{jik} = \rho_{jik} e^{i\theta_{jik}}$$

$$\rho_{jik} = \rho_{ik}/\rho_{ij}$$

因为ρ_{ik}的距离很小，所以参数ρ_{jik}的数值接近 0；那么上述方程退化为$p_k = p_i$；在 SLAM 算法中，我们通过连续的关键帧得到如下连接在一条线的多个节点，$n_1 - m_1 - m_2 - \ldots - m_k - n_2$；其中$m_1$到$m_k$之间的 k 个节点重合在一起；我们总体的处理方式是在位置计算时，将重合的节点$m_1 - m_k$当作同一个节点 m 处理；在后续的角度计算中，再分别计算节点$m_1 - m_k$对应的 k 个局部坐标系的对应角度；对于上述节点重合的情况，我们具体分析如下；

当 SLAM 算法增加一个新的节点m_1时，检测到节点m_1和前驱节点n_1之间的距离$\rho_{m1,n1} > \epsilon_{thres}$，那么节点$m_1$和节点$n_1$不是重合节点；其中参数$\epsilon_{thres}$用于判断节点是否重合的阈值；我们使用节点$m_1$和节点$n_1$计算虚拟节点位置$p^c_{v,m1}$；

当算法增加一个新的节点m_2时，检测到节点m_2和前驱节点m_1之间的距离$\rho_{m2,m1} < \epsilon_{thres}$，那么节点$m_2$和节点$m_1$属于重合节点；此时我们将节点$m_2$和节点$m_1$当作同一个节点；因此不会增加新的节点$m_2$和对应的虚拟节点；依此类推，当算法检测到重合节点集合$S = \{m_i, i \in [1, k]\}$中$m_i, i \in [2, k]$的任意节点，都不会增加新的节点；

当算法增加一个新的节点n_2时，检测到节点n_2和前驱节点m_k之间的距离$\rho_{n2,mk} > \epsilon_{thres}$；那么节点$n_2$和节点$m_k$不是重合节点；此时我们使用节点$m_1$（因为重合集合 S 中所有节点都使用第一个节点$m_1$代替）和节点$n_2$计算虚拟节点位置$p^c_{v,n2}$；

在图$G = [V, E]$中共存在 n 个节点，这些节点对应关键帧；第 1 个关键帧到第 n 个关键帧可以构建一条连线（$1 - 2 - 3 - \ldots - n$）；为了将 SLAM 中节点之间位姿变换转换为线性方程组的过程展示清楚，我们使用伪代码的形式给出具体方式；

我们将构建线性方程的过程分为两个部分，分别是增加关键帧（过程 1）和增加位姿约束（过程 2）；这两个过程是同时进行的；在增加关键帧（节点 i）的时候，必然会增加一组节点 i 到前驱节点$i - 1$之间的位姿变换；然后我们除此之外的所有增加位姿变换都归类到“增加位姿约束（过程 2）”部分；

过程 1，增加关键帧

输入，新节点i与前驱节点$i - 1$之间的位姿变换$T_{i,i-1}$

输出，节点位置相邻$p = [p^c_1, p^c_{v1}, \ldots, p^c_n, p^c_{vn}]^T$，矩阵L

If(节点位置向量p为空):

节点位置向量p增加变量p^c_1，$p = [p^c_1]$；

Elseif(节点位置向量p中只有1个变量，$p = [p^c_1]$):

If(新节点i与前驱节点i-1是重合节点，$\rho_{i,i-1} < \epsilon_{thres}$):

设前驱节点$i - 1$所属的重合节点集合为S；

节点i加入集合S；

Else:

节点位置向量中增加三个变量，$p^c_{v1}, p^c_2, p^c_{v2}$；

使用节点2和节点1计算节点1的虚拟节点位置p^c_{v1}；$p^c_{v1} - p^c_1 = (p^c_2 - p^c_1)/\rho_{1,2}$，该方程加入矩阵L；

使用节点1和节点2计算节点2的虚拟节点位置p^c_{v2}；$p^c_{v2} - p^c_2 = (p^c_1 - p^c_2)/\rho_{2,1}$，该方程加入矩阵L；

Else:

If(新节点i与前驱节点$i - 1$是重合节点，$\rho_{i,i-1} < \epsilon_{thres}$):

设前驱节点$i - 1$所属的重合节点集合为S；

节点i加入集合S；

Else:

节点位置向量p增加两个变量p^c_i, p^c_{vi}，$p = [p^T, p^c_i, p^c_{vi}]^T$；

使用节点i和节点$i - 1$构建虚拟节点位置p^c_{vi}，$p^c_{vi} - p^c_i = (p^c_{i-1} - p^c_i)/\rho_{i,i-1}$，该方程加入矩阵L；

过程 2，增加位姿约束

输入，节点i到节点k的位姿变换T_{ik}

输出，节点位置相邻$p = [p^c_1, p^c_{v1}, \ldots, p^c_n, p^c_{vn}]^T$，矩阵L

设节点i的虚拟节点位置是p^c_{vi}，虚拟节点在局部坐标系Σ_i中的位置t_{iv}；

节点i到节点k的位姿变换$T_{ik} = \begin{bmatrix} R_{ik} & t_{ik} \\ 1 & 0 \end{bmatrix}$，其中平移向量是$t_{ik}$；

使用虚拟节点i_v、节点i和节点k构建线性方程，$p^c_k - p^c_i = \omega_{vik}(p^c_{vi} - p^c_i), \omega_{vik} = f(t_{iv}, t_{ik})$；

设节点k的虚拟节点位置是p^c_{vk}，虚拟节点在局部坐标系Σ_k的位置t_{kv}；

节点k到节点i的位姿变换$T_{ki} = (T_{ik})^{-1} = \begin{bmatrix} R_{ki} & t_{ki} \\ 1 & 0 \end{bmatrix}$，其中平移向量是$t_{ki}$；

使用虚拟节点k_v、节点k和节点i构建线性方程，$p^c_i - p^c_k = \omega_{vki}(p^c_{vk} - p^c_k), \omega_{vki} = f(t_{kv}, t_{ki})$；

2.4 节点可定位性

我们设图$G=[V,E]$中共有 n 个节点，n 个节点对应的位置向量是$p=[p_1^c,p_{v1}^c,\ldots,p_n^c,p_{vn}^c]^T\in C^{2n}$；位置向量 p 中每个节点 i 的坐标$p_i^c$都有对应的虚拟节点坐标$p_{vi}^c$；我们可以将上述所有节点线性方程组合，得到如下的线性方程组；

$$Lp=0$$

其中矩阵$L\in C^{m\times 2n}$是线性方程的系数矩阵；该线性方程组对应图$G=[V,E]$中多个节点之间的相似三角形约束；所以在该线性方程组约束下，整个图可以进行缩放；同时因为没有确定任意点$i\in V$的位置的全局坐标，整个图可以在二维空间中旋转和平移；由于图$G=[V,E]$中的 n 个节点是通过连续的关键帧构建的，所有的节点可以连成一条线（$1-2-3-\ldots-n$）；那么我们设节点 1 的全局坐标系Σ_g中的位置是$p_1^c=0+0\times i$，设节点 2 的坐标是$p_2^c=\rho_{12}+0\times i$；这样整个图被两个锚节点（节点 1、节点 2）固定下来，不能发生旋转平移；同时整个图的尺度大小被节点 1 和节点 2 之间的距离ρ_{12}固定下来；

设定节点 1 和节点 2 的全局坐标后，我们需要确定线性方程$Lp=0$是否具有唯一的非零解；该问题是节点的可定位性问题；如果线性方程不具有唯一解，那么存在某些节点的位置是任意的；下面我们从两个角度证明该问题具有唯一的非零解；

首先我们可以使用文献[2]中的网络可定位性条件进行证明；图$G=[V,E]$可自定位的条件如下；

(NS-1) 网络中存在两个锚节点；

(NS-2) 对于每个自由节点，存在两条不相交的路径可以到达两个锚节点；

我们已经满足条件 1，网络中存在两个锚节点；对于条件 2，我们考虑图$G=[V,E]$的子图$G_{sub}=[V,E_{sub}]$；该子图G_{sub}中，只保留节点 1 到节点 n 前后两个节点之间位姿变换约束对应的连线；设节点 i 有 2 个邻居节点，分别是节点 j 和节点 k；我们已知节点 i 到节点 j 的位姿变换T_{ij}和节点 i 到节点 k 的位姿变换T_{ik}；将上述位姿约束转换为相似三角形约束，我们得到节点 i、节点 j 和节点 k 互相之间存在连线，构成一个三角形；那么我们设 4 个节点之间的位姿变换构成的连线如下；

在图中，节点 1 和节点 2 是锚节点，节点 3 和节点 4 是需要定位的自由节点；实际上这个图可以继续延伸下去，得到节点 1-节点 n 的连线构成的图；对于节点 3，我们可以直接找到两条不相交的边到达两个锚节点；对于节点 4，第一条边是$4-3-1$，第二条边是$4-2$；那么同样的方式，对于节点 n 我们可以找这样交错的两条路到达两个锚节点；因此对于每个自由节点，存在两条不相交的路径可以到达两个锚节点；因为子图$G_{sub}=[V,E_{sub}]$是可自定位的，那么边更多的图$G=[V,E]$可以自定位；自定位的含义是每个节点可以通过分布式的方法只使用邻居节点的信息计算自己的全局坐标；本文中我们采用集中式方案，能够获得更多的信息；因此如果图$G=[V,E]$是可自定位的，那么必然是可定位的；

然后我们使用几何变换的方式证明每个节点都是可定位的；在图$G=[V,E]$中，存在 n 个节点，依次记为$1,2,\ldots,n$；设节点 1 的全局坐标是$p_1\in R^2$，节点 1 到节点 2 的位姿变换是T_{12}，那么节点 2 的全局坐标如下；

$$p_2=T_{12}p_1$$

以此类推，已知节点 i 的全局坐标时$p_i\in R^2$，节点 i 到节点$i+1$的位姿变换是$T_{i,i+1}$，那么节点$i+1$的全局坐标如下；

$$p_{i+1}=T_{i,i+1}p_i$$

因此已知节点p_1的全局坐标和任意两个相邻节点p_i,p_{i+1}之间的位姿变换$T_{i,i+1}$。我们可以顺序的推导得到所有节点的全局坐标；从几何上任意节点的全局坐标是可定位的；然后几何上任意三个节点i,j,k之间如果存在两个位姿变换T_{ij},T_{ik}，那么等效为三角形 ijk 的形状和大小是确定的；此处我们设定节点 1 和节点 2 为锚节点，并且 n 个节点可以按顺序连接成一条线；那么任意节点$i\in V$都可以和前面两个节点（$i-1,i-2$）构成三角形约束；然后我们将三角形的相似约束转换为等效的线性方程组，所以该线性方程组具有唯一解；

3. 地图尺度计算

我们设图$G=[V,E]$中共有 n 个节点，n 个节点对应的位置向量是$p=[p_1^c,p_{v1}^c,\ldots,p_n^c,p_{vn}^c]^T\in C^{2n}$；位置向量 p 中每个节点 i 的坐标$p_i^c$都有对应的虚拟节点坐标$p_{vi}^c$；我们可以将上述所有节点线性方程组合，得到如下的线性方程组；

$$Lp=0$$

该线性方程组对应图$G=[V,E]$中多个节点之间的相似三角形约束；所以在该线性方程组约束下，整个图可以进行缩放；同时因为没有确定任意点$i\in V$的位置的全局坐标，整个图可以在二维空间中旋转和平移；由于图$G=[V,E]$中的 n 个节点是通过连续的关键帧构建的，所有的节点可以连成一条线（$1-2-3-\ldots-n$）；那么我们设节点 1 的全局坐标系Σ_g中的位置是$p_1^c=0+0\times i$，设节点 2 的坐标是$p_2^c=\rho_{12}+0\times i$；这样整个图被两个锚节点（节点 1、节点 2）固定下来，不能发生旋转平移；同时整个图的尺度大小被节点 1 和节点 2 之间的距离ρ_{12}固定下来；

然而上述计算地图尺度的过程中，我们只是使用节点 1 和节点 2 之间的距离ρ_{12}来计算整个地图的大小；如果ρ_{12}存在误差，那么整个地图也会随着缩放；对于图$G=[V,E]$在线性方程约束下可以缩放的问题，是由于我们将位姿变换约束放松为相似三角形约束造成的；对此我们可以采用如下方法确定更为精确的地图尺度；

设节点 1 的坐标为$p_1^c=0+i\times 0$，节点 2 的坐标为$p_2^c=x+i\times 0$；然后我们将 n 个节点的位置向量$p\in R^{2n}$分为两个部分$p_a=[p_1^c,p_2^c]$^T 和

$p_s = [p_{v1}^c, p_{v2}^c, p_3^c, p_{v3}^c, \ldots, p_n^c, p_{vn}^c]^T$；将锚节点$p_a \in R^2$当作常数（虽然存在未知量 x），自由节点$p_s \in R^{2n-2}$作为未知量；那么线性方程$Lp = 0$可以写成如下；

$$Bp_a + Hp_s = 0$$

该方程的最小二乘解如下；

$$p_s = (B^T B)^{-1} B^T (-Hp_a)$$
$$= \Phi p_a$$

我们将节点坐标$p_s \in R^{2n-2}$中的虚拟节点去除，剩余的坐标为$ps_{sub} = [p_3^c, p_4^c, \ldots, p_n^c]^T \in R^{n-1}$；然后设节点坐标$p_{sub} = [p_1^c, p_2^c, ps_{sub}^T]^T \in R^n$；节点坐标向量$p_{sub}$是图$G = [V, E]$中 n 个节点的坐标；由于矩阵$\Phi$是常数，$p_a$中只有一个未知量 x；那么节点坐标向量$p_{sub}$中每个节点的坐标都是 x 的一次函数；比如$p_1^c = (a_1 x + b_1) + i \times (c_1 x + d_1), \ldots, p_n^c = (a_n x + b_n) + i \times (c_n x + d_n)$；

设图$G = [V, E]$中共有位姿变换约束集合为$Q = \{T_{ij} = \begin{bmatrix} R_{ij} & t_{ij} \\ 1 & 0 \end{bmatrix}, i, j \in V\}$；$T_{ij}$表示从节点 i 到节点 j 的位姿变换，其中$t_{ij} = [x_{ij}, y_{ij}]^T$是平移向量；那么从观测值上节点 i 和节点 j 之间的距离如下；

$$\rho_{ij} = ||t_{ij}|| = \sqrt{x_{ij}^2 + y_{ij}^2}$$

从p_{sub}的坐标向量中，我们可以得到节点 i 和节点 j 的距离如下；

$$\rho'_{ij} = ||p_i^c - p_j^c||$$
$$= \sqrt{(a_i x + b_i - a_j x - b_j)^2 + (c_i x + d_i - c_j x - d_j)^2}$$

图$G = [V, E]$中共有$|Q|$个位姿变换，那么我们可以构建如下的误差函数；

$$J = \sum_{T_{ij} \in Q} (\rho'^2_{ij} - \rho_{ij}^2)^2$$
$$= \sum_{T_{ij} \in Q} \Big(\big((a_i - a_j)^2 + (c_i - c_j)^2\big) x^2$$
$$- \big(2(a_i - a_j)(b_i - b_j) + 2(c_i - c_j)(d_i - d_j)\big) x$$
$$+ \big((b_i - b_j)^2 + (d_i - d_j)^2 - \rho_{ij}^2\big)\Big)^2$$
$$= Ax^4 + Bx^3 + Cx^2 + Dx + E$$

为了使该误差函数最小，我们对 J 计算导数；

$$\frac{dJ}{dx} = 4Ax^3 + 3Bx^2 + 2Cx + D = 0$$

然后使用三次方程的公式计算三个根x_1, x_2, x_3；舍弃三个根中的复数根之后，对于剩余的根使用 J 的二次导数判断；

$$\frac{dJ^2}{dx^2} = 12Ax^2 + 6Bx + 2C$$

如果根x_i的二次导数为正，那么根x_i是函数 J 的最小值；如果根x_i的二次导数为负，那么根x_i是函数 J 的局部最大值；得到优化的地图尺度 x 之后，我们可以使用如下方程计算每个节点的全局坐标；

$$p_s = (B^T B)^{-1} B^T (-Hp_a)$$
$$p_a = [0, x + 0 \times i]^T$$

该全局坐标p_s使用第一和第二个节点的坐标$p_a = [0, x + 0 \times i]^T$作为锚节点；我们只需要计算得到每个节点在某个锚节点下的坐标，这样整个地图的形状就固定下来；如果需要以其他节点为锚节点，可以将整个地图对应的旋转平移；

4. 不同传感器的位姿变换

本节中我们讨论不同类型的传感器获得的数据，怎样转化为不同节点之间的位姿变换；我们可以通过相机、激光雷达、IMU 传感器、GPS 传感器、轮速计和回环检测得到两帧之间的位姿变换；下面分别讨论各种传感器获得的位姿变换；

4.1 非 GPS 传感器

我们首先讨论相机和激光雷达传感器的情况；这两种传感器可以获得确定的两个节点（如节点 i 和节点 j）之间的位姿变换T_{ij}；通常节点 i 与节点 j 是相邻的两个关键帧，或者距离非常接近的两个关键帧；

然后对于 IMU 传感器和轮速计，这两种传感器通常获得的是相邻的两个关键帧之间位姿变换$T_{i-1,i}$；

回环检测部分同样可以获得两个节点（节点 i 和节点 j）之间的位姿变换T_{ij}；并且通常节点 i 与节点 j 是机器人绕了一圈后回到原点找到的两个相邻节点；节点 i 与节点 j 在机器人运动时间上相隔较远，但是在距离上较近；

4.2 GPS 传感器

GPS 传感器的情况较为特殊，GPS 传感器获得的数据是经度和纬度；我们可以使用公式将经纬度转换为具体的距离值；那么 GPS 传感器每次检测的位置都是在固定的经纬度坐标系的，或者说 GPS 传感器具有一个固定的全局坐标系；其他类型的传感器（如相机和激光雷达）只能获得具体的两个节点（节点 i、节点 j）之间的位姿变换T_{ij}；如果需要获得距离很远的两个节点（节点 i、节点 $i + n$）之间的位姿变换，只能通过不断累加得到，$T_{i,i+n} = T_{i,i+1} T_{i+1,i+2} \ldots T_{i+n-1,i+n}$；这样获得累加位姿变换$T_{i,i+n}$的误差会逐渐累加；对于 GPS 传感器，因为具有固定的经纬度坐标系，对于距离很远的两个节点（节点 i、节点$i + n$）之间的平移量$t_{i,i+n}$的仍然具有非常高的精度；因为 GPS 传感器的数据对于大型地图在保持整体地图形状的准确性上具有很大的作用；

GPS 传感器的另一个特殊情况是 GPS 数据只有位移，没有旋转信息；假设我们获得了节点 i 的 GPS 数据（经纬度坐标系中的位置），然后获得了节点 j 的 GPS 数据；那么我们只能得到在经纬度坐标系中，节点 i 到节点 j 的

平移量$t_{ij} \in R^2$，没有旋转信息；因此 GPS 传感器获得的平移向量$t_{ij} \in R^2$不能和其他传感器获得位姿变换$T_{ij} \in R^{3\times3}$组合，构建线性方程约束；虽然我们可以构建非线性的位姿图约束，但是这个和我们希望构建线性方程的优化算法不同；不过由于所有的 gps 数据都在同一个坐标系（经纬度坐标系）中测量，那么任意三个节点（节点 i、节点 j 和节点 k）的 GPS 数据可以构建线性方程；

在图$G = [V, E]$中共有 n 个节点（关键帧）；GPS 数据的更新频率通常远低于相机传感器或者 IMU 传感器；所以不是每个节点都有对应的 GPS 数据，我们设拥有 GPS 数据的节点序号集合为$G_{index} = \{g_i, i \in [1, m]\}$；序号集合中每个序号值$g_i \in [1, n]$，然后该集合中共有 m 个元素，对应 m 个 GPS 经纬度数据；设 m 个 GPS 经纬度数据的集合为$G = \{pos_{g(i)} = [lon_{g(i)}, lat_{g(i)}]^T, g(i) \in [1, n], i \in [1, m]\}$；

设第 i 个 GPS 经纬度数据为$pos_{g(i)} = [lon_{g(i)}, lat_{g(i)}]^T$，第 j 个 GPS 经纬度数据为$pos_{g(j)} = [lon_{g(j)}, lat_{g(j)}]^T$；我们可以通过球面距离公式或者 UTM 平面坐标，将节点g_i和节点g_j的经纬度数据转化为相对平移向量；

$$t_{g(i),g(j)} = f(pos_{g(i)}, pos_{g(j)})$$

其中，函数 f 将两个节点的经纬度$pos_{g(i)}, pos_{g(j)}$转化为平移向量$t_{g(i),g(j)}$；

获得 GPS 传感器的数据后，我们会构建两种类型的线性方程，分别是相邻 GPS 数据的线性方程和跨节点 GPS 数据的线性方程；下面首先介绍相邻 GPS 数据的线性方程；

已知第 i 个 GPS 数据是$pos_{g(i)}$，第$i+1$个 GPS 数据是$pos_{g(i+1)}$，第$i+2$个 GPS 数据是$pos_{g(i+2)}$；使用节点g_{i+1}作为中间节点，那么节点g_{i+1}到节点g_i的相对位置是$t_{g(i+1),g(i)} = f(pos_{g(i+1)}, pos_{g(i)})$；节点$g_{i+1}$到节点$g_{i+2}$的相对位置是$t_{g(i+1),g(i+2)} = f(pos_{g(i+1)}, pos_{g(i+2)})$；节点$g_{i+1}$到节点$g_{i+2}$的相对位置是$t_{g(i+1),g(i+2)} = f(pos_{g(i+2)}, pos_{g(i+1)})$；首先需要保证这三个节点没有重合的节点，判断方式如下；

$$\rho_{g(i+1),g(i)} = \left|\left|t_{g(i+1),g(i)}\right|\right| > \epsilon_{thres}$$

$$\rho_{g(i+1),g(i+2)} = \left|\left|t_{g(i+1),g(i+2)}\right|\right| > \epsilon_{thres}$$

$$\rho_{g(i),g(i+2)} = \left|\left|t_{g(i),g(i+2)}\right|\right| > \epsilon_{thres}$$

在保证没有重合节点后，我们可以构建如下线性方程；

$$p^c_{g(i+2)} - p^c_{g(i+1)} = \omega_{i+1}(p^c_{g(i)} - p^c_{g(i+1)})$$

$$\omega_{i+1} = \rho_{i+1}e^{i\theta_{i+1}}$$

其中参数ρ_{i+1}和θ_{i+1}可以如下计算；

$$\rho_{i+1} = \frac{\rho_{g(i+1),g(i+2)}}{\rho_{g(i+1),g(i)}}$$

$$\cos(\theta_{i+1}) = \frac{t_{g(i+1),g(i)}t_{g(i+1),g(i+2)}}{\left|\left|t_{g(i+1),g(i)}\right|\right|\left|\left|t_{g(i+1),g(i+2)}\right|\right|}$$

此外我们还可以构建跨节点 GPS 数据的线性方程；任意两个相邻的 GPS 数据结合计算平移向量$t_{g(i),g(i+1)}$并不能完整的使用 GPS 提供的信息；因为 GPS 数据的优势在于相距很远的两个节点之间的相对位置测量仍然能保持很好的精度；所以除了构建上述线性方程，我们还需要对距离很远的节点使用 GPS 数据构建线性方程；

已知获得的一组 GPS 经纬度数据的集合为$G = \{pos_{g(i)} = [lon_{g(i)}, lat_{g(i)}]^T, g_i \in [1, n], i \in [1, m]\}$；那么我们设想是否可以将第一个 GPS 数据$pos_{g(1)}$和其他任意 GPS 数据$pos_{g(i)}, i \in [2, m]$组合，计算$t_{g(1),g(i)}, i \in [2, m]$；但是这个方案存在如下问题；如果第一个 GPS 数据$pos_{g(1)}$存在很大的误差，那么所有计算得到的平移向量$t_{g(1),g(i)}$都存在很大的误差；因此在计算位姿变换时，不能完全信任某一个 GPS 数据，而是应该大致均匀的使用所有的 GPS 数据；

然后我们设想是否可以随机的在 GPS 数据集合中选取两个数据点（$pos_{g(i)}, pos_{g(j)}$）计算平移向量$t_{g(i),g(j)}$；然后使用节点g_i的相邻节点g_{i+1}的 GPS 数据$pos_{g(i+1)}$和$pos_{g(i)}$计算平移向量$t_{g(i),g(i+1)}$；使用节点g_i, g_j, g_{i+1}构建线性方程；

$$p^c_{g(j)} - p^c_{g(i)} = \omega_{g(i+1),g(i),g(j)}(p^c_{g(i+1)} - p^c_{g(i)})$$

$$\omega_{g(i+1),g(i),g(j)} = h(t_{g(i),g(i+1)}, t_{g(i),g(j)})$$

该方案存在如下问题；节点g_i和节点g_j是随机选择的，那么这两个数据点$pos_{g(i)}$和$pos_{g(j)}$之间的距离通常是很大的（因为是在整个地图上随机选择的两个点）；平移向量$t_{g(i),g(j)}$的模长会远大于平移向量$t_{g(i),g(i+1)}$的模长，$\rho_{g(i),g(j)} \gg \rho_{g(i),g(i+1)}$；上述线性方程对应的等效的相似三角形约束如下；

$$\rho_{g(i),g(j)} = \rho_{g(i+1),g(i),g(j)}\rho_{g(i),g(i+1)}$$

$$\theta_{g(i),g(j)} = \theta_{g(i+1)g(i),g(j)} + \theta_{g(i),g(i+1)}$$

因为$\rho_{g(i),g(j)} \gg \rho_{g(i),g(i+1)}$，那么$\rho_{g(i+1),g(i),g(j)}$的数值很大；对线性方程组$Lp = 0$使用最小二乘法计算时，上述线性方程对应一个误差项；为了使得该误差项减小，会倾向于修改节点g_{i+1}的坐标，因为节点g_{i+1}相乘的系数$\rho_{g(i+1),g(i),g(j)}$很大，稍微的修改就可以使得整个误差项下降很多；而不会倾斜修改节点g_j，因为节点g_j对应的系数 1 远小于$\rho_{g(i+1),g(i),g(j)}$，需要使节点g_j发生很大的移动，才能减小误差；

从几何上看，节点g_{i+1}、节点g_i和节点g_j组成相似三角形约束；因为节点g_j到节点g_(i)的距离（$\rho_{g(i),g(j)}$）远大于节点g_i到节点g_{i+1}的距离（$\rho_{g(i),g(i+1)}$）；那么为了使得这三个节点尽量满足相似三角形的性质，会更倾向于略微修改节点g_{i+1}的位置；这样造成的结果是，虽然距离很远的节点g_i和节点g_j构建了线性方程约束；但这个线性方程约束在几何上的实际效果却是在节点g_i的附近，局部微

调了节点g_{i+1}的位置；那么这样的线性方程没有实现对地图整体形状的约束；

对此我们需要在 GPS 数据集合 G 中寻找这样的三个节点（节点g_i、节点g_j和节点g_k）；这三个节点中没有重合节点；

$$\rho_{g(i),g(j)} > \epsilon_{thres}$$

$$\rho_{g(i),g(k)} > \epsilon_{thres}$$

$$\rho_{g(j),g(k)} > \epsilon_{thres}$$

并且这三个节点互相之间的距离近似，或者我们使用如下公式描述；

$$\frac{\rho_{g(i),g(j)}}{\rho_{g(i),g(k)}} < p_{thres}, \text{if } \rho_{g(i),g(j)} \geq \rho_{g(i),g(k)}$$

$$\frac{\rho_{g(i),g(k)}}{\rho_{g(i),g(j)}} < p_{thres}, \text{if } \rho_{g(i),g(j)} < \rho_{g(i),g(k)}$$

$$\frac{\rho_{g(j),g(i)}}{\rho_{g(j),g(k)}} < p_{thres}, \text{if } \rho_{g(j),g(i)} \geq \rho_{g(j),g(k)}$$

$$\frac{\rho_{g(j),g(k)}}{\rho_{g(j),g(i)}} < p_{thres}, \text{if } \rho_{g(j),g(i)} < \rho_{g(j),g(k)}$$

$$\frac{\rho_{g(k),g(i)}}{\rho_{g(k),g(j)}} < p_{thres}, \text{if } \rho_{g(k),g(i)} \geq \rho_{g(k),g(j)}$$

$$\frac{\rho_{g(k),g(j)}}{\rho_{g(k),g(i)}} < p_{thres}, \text{if } \rho_{g(k),g(i)} < \rho_{g(k),g(j)}$$

其中p_{thres}表示两边的比例；我们需要保证三角形$g_i g_j g_k$中任意两边的比例小于p_{thres}（长边比短边）；

因为节点g_i、节点g_j和节点g_k在整个地图上随机寻找的，通常三个点之间的距离都较大；那么使用这三个点构成的相似三角形构建线性方程，会对整个地图的形状进行很好的约束；因为为了尽量满足这三个点构成的相似三角形形状，那么需要对节点g_i、节点g_j和节点g_k的位置进行均匀且较大的移动；

对于如何在 GPS 数据集合 G 中寻找尽量多的这样的三角形组合；我们采用如下方案；在 GPS 数据集合中随机寻找三个点，判断是否满足上述边的比例条件，如果不满足则重新寻找；我们可以设定寻找的最大次数f_{max}和所需的三角形个数Δ_{req}；

使用回环检测同样可以获得两个点（节点 i 和节点 j）之间的位姿变换T_{ij}，并且这两个点在机器人运动的时间上相隔很远，是否也会出现上述 GPS 数据的问题；对此应该不存在的，因为回环检测得到的两个点虽然在运动时间上间隔很远，但是在空间距离上间隔很近；设位姿变换T_{ij}的平移向量t_{ij}的模长为ρ_{ij}；设节点 i 到虚拟节点v_i的距离是ρ_{iv}；那么ρ_{ij}和ρ_{iv}的长度应该相差不是很大，所以不会遇到上述相似三角形形状的问题；

得到满足要求的三个节点g_i, g_j, g_k后，我们可以构建如下线性方程；

$$p^c_{g(k)} - p^c_{g(i)} = \omega_{g(j),g(i),g(k)}(p^c_{g(j)} - p^c_{g(i)})$$

$$\omega_{g(j),g(i),g(k)} = h(t_{g(i),g(j)}, t_{g(i),g(k)})$$

对于 GPS 数据集合 G，我们会构建两种类型的线性方程，分别是相邻 GPS 数据的线性方程和跨节点 GPS 数据的线性方程；下面首先介绍相邻 GPS 数据的线性方程；下面分别使用伪代码展示具体方法；

过程 3，相邻 GPS 数据构建线性方程

输入，GPS数据集合$G = \{pos_{g(i)} = [lon_{g(i)}, lat_{g(i)}]^T, g_i \in [1, n], i \in [1, m]\}$，重合节点距离$\epsilon_{thres}$

输出，节点位置相邻$p = [p^c_1, p^c_{v1}, \ldots, p^c_n, p^c_{vn}]^T$，矩阵L

For i in $[1, m-2]$:

节点g_i, g_{i+1}的相对平移$t_{g(i),g(i+1)} = f(pos_{g(i)}, pos_{g(i+1)})$，距离为$\rho_{g(i),g(i+1)}$；

节点g_{i+1}, g_{i+2}的相对平移$t_{g(i+1),g(i+2)} = f(pos_{g(i+1)}, pos_{g(i+2)})$，距离为$\rho_{g(i+1),g(i+2)}$；

节点g_i, g_{i+2}的相对平移$t_{g(i),g(i+2)} = f(pos_{g(i)}, pos_{g(i+2)})$，距离为$\rho_{g(i),g(i+2)}$；

If($\rho_{g(i+1),g(i)} > \epsilon_{thres}$):

break

If($\rho_{g(i+1),g(i+2)} > \epsilon_{thres}$):

break

If($\rho_{g(i),g(i+2)} > \epsilon_{thres}$):

break

构建线性方程，$p^c_{g(i+2)} - p^c_{g(i+1)} = \omega_{i+1}(p^c_{g(i)} - p^c_{g(i+1)})$，加入矩阵L；

过程 4，跨节点 GPS 数据构建线性方程

输入，GPS数据集合$G = \{pos_{g(i)} = [lon_{g(i)}, lat_{g(i)}]^T, g_i \in [1, n], i \in [1, m]\}$，边比例$k_{thres}$，最大寻找次数$f_{max}$，所需三角形个数$\Delta_{req}$

输出，节点位置相邻$p = [p^c_1, p^c_{v1}, \ldots, p^c_n, p^c_{vn}]^T$，矩阵L

迭代次数$iter_{num} = 0$，找到三角形个数$\Delta_{num} = 0$

For $iter_{num} < f_{max}$ and $\Delta_{num} < \Delta_{req}$

$iter_{num} = iter_{num} + 1$；

GPS数据集合G中，随机选择三个点g_i, g_j, g_k；

节点g_i, g_j的相对平移$t_{g(i),g(j)} = f(pos_{g(i)}, pos_{g(j)})$，距离为$\rho_{g(i),g(j)}$；

节点g_i, g_k的相对平移$t_{g(i),g(k)} = f(pos_{g(i)}, pos_{g(k)})$，距离为$\rho_{g(i),g(k)}$；

节点g_j, g_k的相对平移$t_{g(j),g(k)} = f(pos_{g(j)}, pos_{g(k)})$，距

离为$\rho_{g(j),g(k)}$；

If($\rho_{g(i),g(j)} < \epsilon_{thres}$):

break

If($\rho_{g(i),g(k)} < \epsilon_{thres}$):

break

If($\rho_{g(j),g(k)} < \epsilon_{thres}$):

break

If$\left(\frac{\rho_{g(i),g(j)}}{\rho_{g(i),g(k)}} > p_{thres} \text{ and } \rho_{g(i),g(j)} \geq \rho_{g(i),g(k)}\right)$:

break

If$\left(\frac{\rho_{g(i),g(k)}}{\rho_{g(i),g(j)}} > p_{thres} \text{ and } \rho_{g(i),g(j)} < \rho_{g(i),g(k)}\right)$:

break

If$\left(\frac{\rho_{g(j),g(i)}}{\rho_{g(j),g(k)}} > p_{thres} \text{ and } \rho_{g(j),g(i)} \geq \rho_{g(j),g(k)}\right)$:

break

If$\left(\frac{\rho_{g(j),g(k)}}{\rho_{g(j),g(i)}} > p_{thres} \text{ and } \rho_{g(j),g(i)} < \rho_{g(j),g(k)}\right)$:

break

If$\left(\frac{\rho_{g(k),g(i)}}{\rho_{g(k),g(j)}} < p_{thres} \text{ and } \rho_{g(k),g(i)} \geq \rho_{g(k),g(j)}\right)$:

break

If$\left(\frac{\rho_{g(k),g(j)}}{\rho_{g(k),g(i)}} < p_{thres} \text{ and } \rho_{g(k),g(i)} < \rho_{g(k),g(j)}\right)$:

break

使用节点g_i、节点g_j和节点g_k构建线性方程，$p^c_{g(k)} - p^c_{g(i)} = \omega_{g(j),g(i),g(k)}(p^c_{g(j)} - p^c_{g(i)})$，该方程加入矩阵L；

$\Delta_{num} = \Delta_{num} + 1$；

4.3 位姿变换的方差

我们可以通过相机、激光雷达、IMU 传感器、GPS 传感器、轮速计和回环检测得到两帧之间的位姿变换；通常不同传感器获得位姿变换具有不同的方差，我们可以根据传感器自身的特点预设设定每种传感器的方差；或者根据 SLAM 算法的前端模块中，计算位姿变换的过程中得到位姿T_{ij}的方差；二维平面上的位姿变换矩阵T_{ij}是3×3矩阵，为了方便起见，我们使用单个方差σ^2_{ij}当作整个位姿变换矩阵的方差；

使用节点 i、节点 j 和节点 k 构建线性方程时，需要使用节点 i 到节点的位姿变换T_{ij}（设方差为σ^2_{ij}）和节点 i 到节点 k 的位姿变换T_{ik}（设方差为σ^2_{ik}）；我们设构建的线性方程带有一个误差变量$v \sim N(0, \sigma^2_{jik})$；

$$p^c_k - p^c_i = \omega_{jik}(p^c_j - p^c_i) + v_{jik}$$

为了方便起见，我们设误差变量v_{jik}的方差$\sigma_{jik}\text{^}2 = \sigma^2_{ij} + \sigma^2_{ik}$；那么组成的线性方程组如下；

$$Bp_s + Hp_a = 0 + v$$

其中误差向量$v \sim N(0, \Sigma)$；我们忽略各个线性方程之间的联系，那么协方差矩阵Σ是对角矩阵；该方程的最小二乘解应该如下，使用该方程可以计算更加精确的节点坐标；

$$p_s = (B\Sigma^{-1}B^T)^{-1}B^T\Sigma^{-1}(-Hp_a)$$

5. 离群点去除

我们设图$G = [V, E]$中共有 n 个节点，n 个节点对应的位置向量是$p = [p^c_1, p^c_{v1}, \ldots, p^c_n, p^c_{vn}]^T \in C^{2n}$；位置向量 p 中每个节点 i 的坐标$p^c_i$都有对应的虚拟节点坐标$p^c_{vi}$；我们可以将上述所有节点线性方程组合，得到如下的线性方程组；

$$Lp = 0$$

其中每个线性方程都是通过节点之间的位姿变换T_{ij}构建的；如果图$G_T = [V, E_T]$中的位姿变换集合$E_T = \{T_{ij}, i, j \in V\}$中存在离群点，对应的线性方程的系数也会出现离群点；线性方程组在使用最小二乘法计算时，每个线性方程都对应一个误差项；离群点会使得方程的解出现很大的偏移；

在非线性优化的 SLAM 方案中，每个位姿变换T_{ij}都会使用观测函数构建误差函数$h(T_{ij})$；然后由这些误差函数组成的误差项$e(T_{ij}) = h(T_{ij})^2$，通过将这些误差项相加得到误差函数 J；但是通常我们会在误差项外侧套一个鲁棒函数如下；

$\rho(x) = x^2, \text{if } -1 < x < 1$

$x, \text{if } x \geq 1$

$-x, \text{if } x \leq -1$

鲁棒函数ρ在 0 附近的函数值上述很快；在超过某个范围后，鲁棒函数的函数值上述很慢；增加鲁棒函数后的误差函数 J 如下；

$$J = \sum_{T_{ij} \in E_T} \rho\left(h(T_{ij})^2\right)$$

因为鲁棒函数的存在，即使某个位姿变换T_{ij}属于离群点，对应的误差项的数值大小也不会很大；因此离群点对整个误差函数的最小值的影响不大；并且原始的误差项$e(T_{ij}) = h(T_{ij})^2$是非线性函数，增加鲁棒函数后的误差项$\rho\left(h(T_{ij})^2\right)$仍然是非线性函数；我们仍然使用数值计算方法求解误差函数的最小值；

然而对于线性方程$Lp = 0$，我们不能采用鲁棒函数的方案；如果在每个线性方程构成的误差项外侧增加鲁棒函数，那么就变成非线性函数；我们将位姿变换构建成线性方程的目的是为了减少计算量，如果使用非线性方程，就没法达到所需的目的；

论文[1]给出了一种在位姿变换中去除离群点的方法；该方法的具体步骤较多，因此我们简要介绍大致的方法；该算法输入图$G_T = [V, E_T]$中的所有位姿变换集合$E_T = \{T_{ij}, i, j \in V\}$，输出去除离群点的位姿变换集合$E'_T = \{T_{ij}, i, j \in V\}$；我们使用集合$E'_T$中的所有位姿变换构建线性方程，就能避免离群点对方程的影响；

该算法需要对输入的位姿变换集合$E_T = \{T_{ij}, i, j \in V\}$中的每个位姿变换$T_{ij}$预设一个属于离群点的概率$p_{ij}$；这个概率并不是需要绝对精确的数值，我们可以使用位姿变换T_{ij}对应的方差σ_{ij}^2大致取值；比如可以采用如下的方案；

$$p_{ij} = 0.1, \text{if } \sigma_{ij}^2 < \sigma_{thres1}^2$$

$$0.2, \text{if } \sigma_{thres1}^2 \le \sigma_{ij}^2 < \sigma_{thres2}^2$$

$$0.3, \text{if } \sigma_{thres2}^2 \le \sigma_{ij}^2$$

概率p_{ij}在离群点去除算法中用于寻找不同节点（节点 i、节点 j）之间的路线；位姿变换T_{ij}表示路径中的边；那么算法会尽量寻找一条离群点少的边组成的路径实现从节点 i 到达节点 j；

该离群点去除算法的大致方法如下；设$G_T = [V, E_T]$中存在两个节点$p_1, p_2 \in V$；然后节点p_1, p_2之间存在三个旋转平移矩阵$T_{12}^1, T_{12}^2, T_{12}^3$；

同时节点p_1, p_2经过另外一个节点（如p_3, p_4, p_5）也能构成连线；那么p_1到p_2的位姿变换可以如下计算；

$$T_{12}^4 = T_{13}T_{32}$$

$$T_{12}^5 = T_{14}T_{42}$$

$$T_{12}^6 = T_{15}T_{52}$$

然后节点p_1, p_2经过另外两个节点（如p_6, p_7或p_8, p_9或p_{10}, p_{11}）可以构成连线；那么p_1到p_2的位姿变换计算如下；

$$T_{12}^7 = T_{16}T_{67}T_{72}$$

$$T_{12}^8 = T_{18}T_{89}T_{92}$$

$$T_{12}^9 = T_{1,10}T_{10,11}T_{11,2}$$

那么我们共得到 9 组节点 1 到节点 2 之间的位姿变换，$T_{12}^i, i \in [1,9]$；如果使用的所有位姿变换（集合E_T）没有离群点，那么这些位姿变换T_{12}^i应该是非常近似的；如果使用的某个位姿变换（集合E_T）是离群点，那么对应的位姿变换T_{12}^i和其他位姿变换相比，差别较大；我们可以将位姿变换T_{12}^i转化为平移向量和旋转向量，然后对每个分量都使用基于统计的 IQR 算法去除离群点；

6. 局部坐标系求解

6.1 非重合节点

我们已经计算得到图$G = [V, E]$中每个节点在全局坐标系Σ_g中的坐标$p_{sub} = [p_1^c, p_2^c, \dots, p_n^c]^T$；设节点 i 的邻居节点为$j \in N_i$，其中节点 j 的坐标为$p_j^c = x_j + iy_i$；使用向量形式表示节点 j 的坐标为$p_j = [x_j, y_j]^T$；节点 j 相对节点 i 的坐标是$p_j^r = [x_j - x_i, y_j - y_i]^T$；这些邻居节点在全局坐标系中的相对坐标向量为$P_i^{neig} = [p_{j1}^r, \dots, p_{jn}^r] \in R^{2|N_i|}, j_1, \dots, j_n \in N_i$；设节点 i 到邻居节点$j \in N_i$的位姿变换是$T_{ij} = \begin{bmatrix} R_{ij} & t_{ij} \\ 1 & 0 \end{bmatrix} \in R^{3\times3}, j \in N_i$，其中平移向量是$t_{ij} = [x_{ij}, y_{ij}]^T \in R^2, j \in N_i$；那么节点 i 的邻居节点$j \in N_i$在局部坐标系$\Sigma_i$中的相对位置向量是$Q_i^{neig} = [t_{ij1}, \dots, t_{ijn}] = [q_{ij1}, \dots, q_{ijn}] \in R^{2|N_i|}, j_1, \dots, j_n \in N_i$；

已知一组点$j \in N_i$在全局坐标系Σ_g中的位置向量$P_i^{neig} \in R^{2|N_i|}$，并且已知点$j \in N_i$在局部坐标系$\Sigma_i$中的位置向量$Q_i^{neig} \in R^{2|N_i|}$；我们需要寻找一个旋转矩阵$R_i$，构建如下误差函数；

$$E(R_i) = \sum_{j \in N_i} \left|\left| R_i p_{ij}^r - q_{ij} \right|\right|^2$$

我们需要使误差函数$E(R_i)$最小的旋转矩阵$R_i \in R^{2\times2}$；该问题属于点云配准问题，我们可以使用 ICP 算法的奇异值分解计算矩阵R_i；矩阵R_i表示将点集从全局坐标系Σ_g中变换到局部坐标系Σ_i的旋转矩阵；坐标系本身的旋转和点集的坐标旋转相反，那么局部坐标系Σ_i相对于全局坐标系Σ_g的旋转矩阵是$(R_i)^{-1}$；旋转矩阵$(R_i)^{-1}$转换为角度θ_i，θ_i表示局部坐标系Σ_i在全局坐标系Σ_g中的角度；

根据 ICP 算法中的奇异值分解方法，使得误差函数$E(R_i)$取得最小值的旋转矩阵如下；

$$H = P_i^{neig} {Q_i^{neig}}^T$$

$$= U\Sigma V^T$$

$$R_i = VU^T$$

对矩阵$H \in R^{2\times2}$计算奇异值分解，得到正交矩阵U, V；旋转矩阵$R_i = VR^T$，然而如果点集P_i^{neig}或者Q_i^{neig}退化为一条直线或者一个点，那么计算得到的矩阵R_i可能是反射矩阵（$|R_i| = -1$）；我们可以通过如下步骤将反射矩阵R_i调整为旋转矩阵；

$$R_i = V\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}U^T$$

那么完整的旋转矩阵计算公式如下；

$$R_i = V\begin{bmatrix} 1 & 0 \\ 0 & |VU^T| \end{bmatrix}U^T$$

过程 4，非重合节点 i 的局部坐标系

输入，节点i的坐标p_i^c，节点i的邻居节点$j \in N_i$，邻居节点$j \in N_i$的坐标p_j^c，邻居节点$j \in N_i$的位姿变换T_{ij}

输出，局部坐标系Σ_i在全局坐标系Σ_g中的角度θ_i

邻居节点$j \in N_i$相对节点i的坐标为$p_j^r = [x_j - x_i, y_j - y_i]^T$；

邻居节点$j \in N_i$在全局坐标系Σ_g中的相对坐标是，

$P_i^{neig} = [p_{j1}^r, \ldots, p_{jn}^r] \in R^{2|N_i|}, j_1, \ldots, j_n \in N_i$；

邻居节点$j \in N_i$在局部坐标系Σ_i中的坐标是，$Q_i^{neig} = [t_{ij1}, \ldots, t_{ijn}] = [q_{ij1}, \ldots, q_{ijn}] \in R^{2|N_i|}, j_1, \ldots, j_n \in N_i$；

使用点集配准计算旋转矩阵R_i，矩阵$H = P_i^{neig} Q_i^{neig\,T}$；

对矩阵H奇异值分解，$H = U\Sigma V^T$；

旋转矩阵为$R_i = V \times diag([1, |VU^T|]) \times U^T$；

局部坐标系Σ_i在全局坐标系Σ_g中的角度是θ_i（旋转矩阵$(R_i)^{-1}$）

6.2 重合节点

如果节点 i 不是重合节点，那么节点 i 只有一个局部坐标系Σ_i；如果节点 i 是重合节点，我们需要对每个重合节点节点计算局部坐标系，具体方式如下；在 SLAM 算法中，我们通过连续的关键帧得到如下连接在一条线的多个节点，$n_1 - m_1 - m_2 - \ldots - m_k - n_2$；其中$m_1$到$m_k$之间的 k 个节点重合在一起，设重合节点集合为$M = \{m_i, i \in [1, k]\}$；在位置计算时，我们将重合的节点$m_1 - m_k$当作同一个节点 m 处理，并且计算得到节点 m 在全局坐标系Σ_g中的位置p_m^c；

重合节点集合$M = \{m_i, i \in [1, k]\}$具有特殊性；因为节点$m_i$与很多邻居节点的$j \in N_{mi}$的距离是 0，不能使用点集配准的方式计算局部坐标系的角度；我们可以采用非线性优化的方案，使用每个重合节点的位姿约束建立误差函数；然后通过对误差函数计算最小值，得到优化的每个节点的局部坐标系；但是该论文中，我们使用线性方程求解的目的是为了避免使用非线性函数，减少计算量；因此我们选择舍弃部分位姿约束条件，使得计算节点m_i的局部坐标系Σ_i的角度时仍然可以使用线性方程的方式计算；我们总体的处理方式如下；对于重合节点集合中端点的节点m_1和节点m_k，仍然可以使用点集配准的方式计算旋转矩阵；对于非端点节点$m_i, i \in [2, k-1]$，使用重合节点互相之间的角度约束计算局部坐标系；

设第一个重合节点m_1的邻居节点为$j \in N_{m1}$；邻居节点$j \in N_{m1}$中可能包括两类节点；第一类节点是重合节点集合$M = \{m_i, i \in [1, k]\}$中的节点；因为重合节点$m_1$与重合节点集合 M 中的节点的距离是 0，不能通过点集配准获得局部坐标系Σ_{m1}的约束信息；所以我们忽略第一类邻居节点；第二类邻居节点是$j \in N_{m1}$ and $j \notin M$（最少情况下只有 1 个邻居节点，节点n_1）；节点m_1与第二类邻居节点的距离不是 0，可以使用点集配准的方式获得局部坐标系Σ_{m1}的角度；设第二类邻居节点集合为$N_{m1}^{part2} = \{j \in N_{m1}$ and $j \notin M\}$；

下面我们使用节点m_1和邻居节点集合N_{m1}^{part2}计算局部坐标系Σ_{m1}的位姿；设重合节点集合$M = \{m_i, i \in [1, k]\}$在全局坐标系中的坐标是$p_m^c = x_m + iy_m$；邻居节点$j \in N_{m1}^{part2}$在全局坐标系$\Sigma_g$中的坐标是$p_j^c = x_j + iy_i$；那么邻居节点$j \in N_{m1}^{part2}$在全局坐标系$\Sigma_g$中的相对坐标是$p_j^r = [x_j - x_m, y_j - y_m]^T$；那么邻居节点$j \in N_{m1}^{part2}$在全局坐标系$\Sigma_g$中的相对坐标向量$P_{m1}^{neig} \in R^{2|N_{m1}^{part2}|}$；节点$m_1$的邻居节点$j \in N_{m1}^{part2}$在局部坐标系$\Sigma_{m1}$中的坐标向量$Q_{m1}^{neig} \in R^{2|N_{m1}^{part2}|}$；然后使用奇异值分解得到坐标向量$P_{m1}^{neig}$到坐标向量$Q_{m1}^{neig}$的旋转矩阵$R_{m1}$；那么全局坐标系$\Sigma_g$到局部坐标系$\Sigma_{m1}$的旋转矩阵为$(R_{m1})^{-1}$；

设最后一个重合节点m_k的邻居节点为$j \in N_{mk}$；邻居节点$j \in N_{mk}$中可能包括两类节点；第一类节点是重合节点集合$M = \{m_i, i \in [1, k]\}$中的节点；因为重合节点$m_k$与重合节点集合 M 中的节点的距离是 0，不能通过点集配准获得局部坐标系Σ_{mk}的约束信息；所以我们忽略第一类邻居节点；第二类邻居节点是$j \in N_{mk}$ and $j \notin M$（最少情况下只有 1 个邻居节点，节点n_2）；节点m_k与第二类邻居节点的距离不是 0，可以使用点集配准的方式获得局部坐标系Σ_{mk}的角度；设第二类邻居节点集合为$N_{mk}^{part2} = \{j \in N_{mk}$ and $j \notin M\}$；

我们使用节点m_k和邻居节点集合N_{mk}^{part2}计算局部坐标系Σ_{mk}的位姿；设重合节点集合$M = \{m_i, i \in [1, k]\}$在全局坐标系中的坐标是$p_m^c = x_m + iy_m$；邻居节点$j \in N_{mk}^{part2}$在全局坐标系$\Sigma_g$中的坐标是$p_j^c = x_j + iy_i$；那么邻居节点$j \in N_{mk}^{part2}$在全局坐标系$\Sigma_g$中的相对坐标是$p_j^r = [x_j - x_m, y_j - y_m]^T$；那么邻居节点$j \in N_{mk}^{part2}$在全局坐标系中的相对坐标向量$P_{mk}^{neig} \in R^{2|N_{mk}^{part2}|}$；节点$m_k$的邻居节点$j \in N_{mk}^{part2}$在局部坐标系$\Sigma_{mk}$中的坐标向量是$Q_{mk}^{neig} \in R^{2|N_{mk}^{part2}|}$；然后使用奇异值分解得到坐标向量$P_{mk}^{neig}$到坐标向量$Q_{mk}^{neig}$的旋转矩阵$R^{mk}$；那么全局坐标系$\Sigma_g$到局部坐标系$\Sigma_{mk}$的旋转矩阵为$(R_{mk})^{-1}$；

下面我们需要计算剩余的重合节点$M_{mid} = \{m_i, i \in [2, k-1]\}$对应的局部坐标系；重合节点集合$M_{mid}$中每个节点$m_i$的邻居节点$j \in N_{mi}$可以分为两类；第一类邻居节点是集合$M = \{m_i, i \in [1, k]\}$的节点，设为$N_{mi}^{part1} = \{j \in N_{mi}$ and $j \in M\}$；通常任意两个相邻的重合节点（节点m_i, m_{i+1}）存在位姿约束$R_{i,i+1}$；然后也可能存在任意两个重合节点（节点$m_i, m_j, i, j \in [1, k]$）存在位姿约束$R_{i,j}$；第一类邻居节点$N_{mi}^{part1}$在后续计算每个节点$m_i$的局部坐标系$\Sigma_{mi}$时会使用；第二类邻居节点集合是$N_{mi}^{part2} = \{j \in N_i$ and $j \notin M\}$；后续计算M_{mid}中每个节点m_i的局部坐标系Σ_i时，我们忽略这类邻居节点；

已知重合节点集合M_{mid}中节点m_i的第一类邻居节点是$j \in N_{mi}^{part1}$；因为节点m_i与节点 j 是重合节点，那么这两个节点之间的位姿约束只有旋转R_{ij}；通过前面步骤已经得到节点m_1的局部坐标系Σ_{m1}在全局坐标系Σ_g中的角度时$(R_{m1})^{-1}$，转化为角度记为θ_{m1}；节点m_k的局部坐标系Σ_{mk}在全局坐标系Σ_g中的角度是$(R_{mk})^{-1}$，转化为角度记为θ_{mk}；设重合节点集合M_{mid}中节点m_i的局部坐标系Σ_{mi}在全局坐标系Σ_g中的角度是θ_{mi}；

下面我们使用节点m_i的第一类邻居节点$j \in N_{mi}^{part1}$构建角度的线性方程；因为节点m_i和节点 j 是重合节点，那么节点m_i到节点 j 的位姿变换是旋转矩阵$R_{mi,j}$；我们将旋转矩阵$R_{mi,j}$转化为角度$\theta_{mi,j}$，那么满足如下式子；

$$\theta_{mi} + \theta_{mi,j} = \theta_j, j \in N_{mi}^{part2}$$

重合节点集合M_{mid}中有$k-2$个节点，我们获得$k-2$组线性方程如下；

$$\theta_{m2} + \theta_{m2,j} = \theta_j, j \in N_{m2}^{part2}$$

$$\dots$$

$$\theta_{m(k-1)} + \theta_{m(k-1),j} = \theta_j, j \in N_{m(k-2)}^{part2}$$

节点m_1和节点m_k的角度已知（θ_{m1}和θ_{m2}），可以作为已知量代入上述线性方程组；该方程的未知量是集合M_{mid}中每个节点的角度$\theta_i, i \in [2, k-1]$；我们将上述线性方程组写成矩阵形式如下；

$$A\theta = b$$

$$\theta = [\theta_2, \dots, \theta_{k-1}]^T \in R^{k-2}$$

然后我们分析该线性方程组是否具有唯一解；如果重合节点集合$M = \{m_i, i \in [1, k]\}$中，每个节点$m_i$只和前驱节点$m_{i-1}$和后继节点$m_{i+1}$有位姿约束（约束最少的情况）；那么角度线性方程组中，方程的个数是$k-1$，未知量个数是$k-2$；并且我们可以写出这种情况下线性方程组的系数矩阵 A，得到矩阵$\text{rank}(A) = k-2$；因此在约束最少的情况下，该线性方程组是有唯一的最小二乘解的；那么增加更多的约束后，该线性方程也是有唯一的最小二乘解；该线性方程组的最小二乘解如下；

$$\theta = (A^T A)^{-1} A^T b$$

从图形上看，上述线性方程的约束是在已知节点m_1的角度θ_{m1}和节点m_k的角度θ_{mk}的情况下，对其他节点$m_i \in M_{mid}$的角度进行微调，使得尽量满足方程；如果我们希望人为的控制每个节点$m_i \in M_{mid}$的角度按照比例调整，那么可以增加如下线性方程；

$$\lambda\left(\frac{\theta_{m2} - \theta_{m1}}{\theta_{m3} - \theta_{m2}}\right) = \lambda\left(\frac{\theta_{12}}{\theta_{23}}\right)$$

$$\dots$$

$$\lambda\left(\frac{\theta_{mk} - \theta_{m(k-1)}}{\theta_{m(k-1)} - \theta_{m(k-2)}}\right) = \lambda\left(\frac{\theta_{m(k-1),mk}}{\theta_{m(k-2),m(k-1)}}\right)$$

增加上述线性方程后，使用最小二乘法求解节点$m_i \in M_{mid}$的角度θ_{mi}时，会使得节点m_i的角度θ_{mi}尽量满足各节点之间的比例；参数λ可以用来调整约束的大小；

过程 5，重合节点集合$M = \{m_i, i \in [1, k]\}$的局部坐标系

输入，重合节点集合$M = \{m_i, i \in [1, k]\}$，重合节点集合 M 的坐标$p_m^c$，每个节点$m_i$的邻居节点$j \in N_i$，邻居节点$j \in N_i$的坐标$p_j^c, j \in N_i$，邻居节点$j \in N_i$的位姿变换$T_{ij}, j \in N_i$

输出，每个节点$m_i, i \in [1, k]$的局部坐标系Σ_i在全局坐标系Σ_g中的角度$\theta_i, i \in [1, m]$

#part1，计算节点m_1的局部坐标系

节点m_1的邻居节点为$j \in N_{m1}$，节点m_1的部分邻居节点集合为$N_{m1}^{part2} = \{j \in N_{m1} \text{ and } j \notin M\}$；

邻居节点$j \in N_{m1}^{part2}$在全局坐标系Σ_g的相对坐标是$p_{m1}^r = [x_j - x_m, y_j - y_m]^T$；

邻居节点$j \in N_{m1}^{part2}$在全局坐标是Σ_g的相对坐标向量是$P_{m1}^{neig} = [p_{j1}^r, \dots, p_{jn}^r] \in R^{2|N_{m1}^{part2}|}, j_1, \dots, j_n \in N_{m1}^{part2}$；

邻居节点$j \in N_{m1}^{part2}$在局部坐标系Σ_{m1}的相对坐标向量是$Q_{m1}^{neig} = [t_{ij1}, \dots, t_{ijn}] \in R^{2|N_{m1}^{neig}|}, j_1, \dots, j_n \in N_{m1}^{part2}$；

使用点集配准对P_{m1}^{neig}和Q_{m1}^{neig}配准，得到旋转矩阵R_{m1}；

旋转矩阵$(R_{m1})^-1$转为θ_{m1}；

#part2，计算节点m_k的局部坐标系

节点m_k的邻居节点为$j \in N_{mk}$，节点m_k的部分邻居节点集合为$N_{mk}^{part2} = \{j \in N_{mk} \text{ and } j \notin M\}$；

邻居节点$j \in N_{mk}^{part2}$在全局坐标系Σ_g的相对坐标是$p_{m1}^r = [x_j - x_m, y_j - y_m]^T$；

邻居节点$j \in N_{m1}^{part2}$在全局坐标是Σ_g的相对坐标向量是$P_{mk}^{neig} = [p_{j1}^r, \dots, p_{jn}^r] \in R^{2|N_{mk}^{part2}|}, j_1, \dots, j_n \in N_{mk}^{part2}$；

邻居节点$j \in N_{mk}^{part2}$在局部坐标系Σ_{mk}的相对坐标向量是$Q_{mk}^{neig} = [t_{ij1}, \dots, t_{ijn}] \in R^{2|N_{mk}^{neig}|}, j_1, \dots, j_n \in N_{mk}^{part2}$；

使用点集配准对P_{mk}^{neig}和Q_{mk}^{neig}配准，得到旋转矩阵R_{mk}；

旋转矩阵$(R_{mk})^{-1}$转为θ_{mk}；

#part3，计算剩余节点集合$M_{mid} = \{m_i, i \in [2, k-1]\}$中每个节点$m_i$的局部坐标系

已知节点m_1的角度是θ_{m1}，节点m_k的角度是θ_{mk}；

设节点$m_i \in M_{mid}$的邻居节点为$j \in N_{mi}$，节点m_i的部分邻居节点集合为$N_{mi}^{part1} = \{j \in N_{mi} \text{ and } j \in M\}$；

对于邻居节点$j \in N_{mi}^{part1}$，设节点i到节点j的旋转矩阵为$R_{mi,j}$，转化为角度为$\theta_{mi,j}$；

对邻居节点$j \in N_{mi}^{part1}$，构建线性方程$\theta_{mi} + \theta_{mi,j} = \theta_j, j \in N_{mi}^{part1}$；

对节点集合M_{mid}中每个节点构建线性方程，得到线性方程组$A\theta = b, \theta = [\theta_2, \dots, \theta_{k-1}]^T \in R^{k-2}$；

线性方程组的最小二乘解，$\theta = (A^T A)^{-1} A^T b$；

迭代算法的实现

1. 节点i从内邻居j得到p_j^z的估计值$\hat{p}_j^z$；

2. 节点i从外邻居得到加权辅助变量$\omega_{ki}^z\zeta_k^z(t)$；

3. 节点i使用辅助变量估计自己的位置如下；

$$\zeta_i^z(t+1)=\frac{1}{\sqrt{\sigma_i^z}}\sum_{j\in N_i}\omega_{ij}^z\hat{p}_j^z\ (t)$$

$$\hat{p}_i^z(t+1)=\hat{p}_i^z\ (t)-\frac{\epsilon^z}{\sqrt{\sigma_i^z}}\sum_{j\in N_k}\omega_{ki}^z\zeta_k^z\ (t)$$

4. 节点i发送自己的位置估计$\hat{p}_i^z\ (t+1)$给自己的外邻居；

5. 节点i发送自己的加权辅助变量$\omega_{ki}^z\zeta_i^z\ (t+1)$给自己的内邻居；

REFERENCES

Brown, F., Harris, M.G., and Other, A.N. (1998). Name of paper. In Name(s) of editor(s) (ed.), *Name of book in italics*, page numbers. Publisher, Place of publication.

Smith, S.E. (2004). *Name of book in italics*, page or chapter numbers if relevant. Publisher, Place of publication.

Smith, S.E. and Jones, L.Q. (2008). Name of paper. *Name of journal in italics*, volume (number), page numbers.

【待补充，调整格式后加入】