A linear SLAM backend optimization algorithm on 2D space

Zhitao Wu

Hangzhou Dianzi University

Abstract: Common SLAM back-end optimization methods include filtering approaches and nonlinear optimization techniques. Among these, the extended Kalman filter (EKF) is a commonly used filtering method. This approach treats the robot's pose and all landmarks as state variables, which are updated using observational data. However, as the map becomes larger, the dimensionality of the state variables increases rapidly, leading to a significant growth in computational complexity. Nonlinear optimization methods, on the other hand, construct an error term by associating the spatial coordinates of observed feature points with the robot's pose. All error terms are then summed to form an error function, and the robot's pose and feature point positions are estimated by minimizing this error function. Nevertheless, this approach also suffers from high computational demands. This paper proposes an algorithm for backend optimization using systems of linear equations, where linear equations are utilized to represent the pose constraints between different keyframes.

Keywords: SLAM, backend optimizaiton, linear equation.

1. Introduction

Common SLAM back-end optimization methods include filtering approaches and nonlinear optimization techniques. Among these, the extended Kalman filter (EKF) is a commonly used filtering method. This approach treats the robot's pose and all landmarks as state variables, which are updated using observational data. However, as the map becomes larger, the dimensionality of the state variables increases rapidly, leading to a significant growth in computational complexity. Nonlinear optimization methods, on the other hand, construct an error term by associating the spatial coordinates of observed feature points with the robot's pose. All error terms are then summed to form an error function, and the robot's pose and feature point positions are estimated by minimizing this error function. Nevertheless, this approach also suffers from high computational demands. This paper proposes an algorithm for backend optimization using systems of linear equations, where linear equations are utilized to represent the pose constraints between different keyframes.

2. Construct linear equations

2.1 Constructing linear equations using pose transformation

In SLAM algorithms, various methods can be employed to obtain the pose transformation between two keyframes (in the 2D case, the transformation matrix $T \in R^{3\times3}$, representing rotation and translation). The table below lists different types of feature points extracted between two keyframes using camera sensors, along with the corresponding algorithms that can be used to compute the pose transformation:

Table 1. Transformation calculation algorithm

Feature point type	Transformation calculation algorithm
2d-2d	epipolar geometry
2d-2d(planar)	homography matrix
2d-3d	DLT, EPnP
3d-3d	ICP

If two successive point clouds are obtained using a LiDAR sensor, various point cloud registration algorithms can be applied to compute the pose transformation between them. Additionally, pose transformations between frames can also be estimated using other methods such as IMU sensors, GPS sensors, wheel odometry, and loop closure detection.

Regardless of the type of sensor used, the initial information we obtain consists of the pose transformations between two keyframes (nodes). The goal of back-end optimization is to use these pose transformation measurements to estimate an accurate global map. Here, we denote a keyframe as node $i \in V$. Each node i has its own local coordinate system, denoted as Σ_i. There exists an unknown rotation angle α_i between the local coordinate system Σ_i of node i and the global coordinate system Σ_g. Let the coordinates of node i in the 2D plane be denoted as $p_i = [x_i, y_i]^T$. An undirected edge $(i, j) \in E$ is used to represent the distance between nodes i and j. Thus, all nodes and the distance constraints between them can be represented as an undirected graph $G = [V, E]$.

The initial information we obtain from the sensors is the relative pose transformation T_{ij} between any two nodes (node i and node j). Let the set of all pose transformations T_{ij} be denoted as $E_T = \{T_{ij}, i, j \in V\}$. We can define a pose constraint graph $G_T = [V, E_T]$, where V represents the set of all nodes (keyframes), and E_T represents the set of all pose

transformations. The length-constrained graph $G = [V, E]$ can be derived from the pose-constrained graph $G_T = \{V, E_T\}$ using the following general approach: if there exist two pose transformations T_{ij} and T_{ik} for any three nodes $i, j, k \in V$, the size and shape of the triangle ijk are determined. From this, we can compute the three side lengths of the triangle ijk, which are used as elements of the edge length set E in the graph $G = [V, E]$. Here, the length-constrained graph $G = [V, E]$ is used for computing the positions of the nodes, while the pose-constrained graph $G_T = [V, E_T]$ is employed for outlier removal, local optimization, and other purposes.

Next, we use linear equations to represent the pose transformations between three nodes in a 2D plane. This method is consistent with those proposed in [1] and [2]. Assume that node i has two neighbor nodes, namely j and k. Using the front-end module of the SLAM algorithm, we obtain the pose transformation from node i to node j as $T_{ji} \in R^{3\times3}$, and the pose transformation from node i to node k as $T_{ik} \in R^{3\times3}$.

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

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

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

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

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

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

From a geometric perspective, the pose transformation constraints between nodes i, j, and k define the shape of triangle ijk. Since the angles between the local coordinate systems $\Sigma_i, \Sigma_j, \Sigma_k$, and the global coordinate system Σ_g are unknown, triangle ijk can undergo arbitrary rotation and translation in the 2D plane. However, due to the constraints imposed by T_{ji} and T_{ik}, the shape of triangle ijk remains unchanged. Consequently, the constraints T_{ji} and T_{ik} can be reformulated as congruence triangle constraints, such as the following set of conditions: $\rho_{ij} = \left||p_i - p_j|\right|$, $\rho_{ik} = \left||p_i - p_k|\right|$, and the angle θ_{jik}.

Given that the pose transformation from node i to node k is T_{ik}, where the translation component is $t_{ik} = [x_{ik}, y_{ik}]^T$, the value of ρ_{ij} is determined as follows.

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

Given that the pose transformation from node i to node j is T_{ji}, the pose transformation from node j to node i is $T_{ji} = T_{ji}^{-1} \in R^{3\times3}$. We define $t_{ij} = [x_{ij}, y_{ij}]^T = \left[T_{ji}[1,3], T_{ji}[1,2]\right]^T$. Then, ρ_{ik} is defined as follows.

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

In the local coordinate system Σ_i of node i, the relative coordinates of node j are t_{ij}, and the relative coordinates of node k are t_{ik}. Therefore, the angle θ_{ijk} is defined as follows.

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

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

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

We convert the pose transformation constraints T_{ji} and T_{ik} between nodes iii, jjj, and kkk into congruent triangle constraints, represented by ρ_{ij}, ρ_{ik} and θ_{jik}. To express the pose constraints between nodes using linear equations, we relax the congruent triangle constraints into similar triangle constraints, defined as $\rho_{jik} = \rho_{ik}/\rho_{ij}$ and θ_{jik}. In this way, we ensure the shape of triangle ijkijkijk remains unchanged, but the triangle as a whole is allowed to scale. Under this relaxation, the set of similar triangle constraints can be represented by the following system of linear equations.

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

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

Among them, p_i, p_j and p_k represent the coordinates of nodes i, j, and k, respectively, which are variables requiring constraints. $R_{jik}(p_j - p_i)$ denotes the vector obtained by rotating vector ij by an angle θ_{jik}, resulting in a vector that aligns with the direction of vector ik. Furthermore, there exists a proportional relationship $\rho_{jik} = \rho_{ik}/\rho_{ij}$ between $R_{jik}(p_j - p_i)$ and vector ik. Here, both ρ_{jik} and the rotation matrix R_{jik} can be computed using the pose transformations T_{ji} and T_{ik}, which are known constants. The form of this linear system of equations corresponds to the equations describing the constraints between nodes in a plane as presented in [1]. In the graph $G = [V, E]$, any three nodes (whether adjacent or non-adjacent) form a determined triangle as long as there exist two pose transformations between them. The similarity constraints of this triangle can be expressed using a linear system of equations.

In [2], the equivalent form of the aforementioned equations is presented using linear equations in complex numbers. Let the complex coordinate of node $i \in V$ be $p_i^c = x_i + y_i i$, where x_i and y_i represent the x-axis and y-axis components of the coordinate of node i, respectively. Then, the above system of linear equations can be expressed as the following linear equations in complex numbers.

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

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

Let p_{ik}^c represent the complex coordinate form of vector ik, and let θ_{ik} denote the angle of vector ik in the local

coordinate system Σ_i. Similarly, let p_{ij}^c represent the complex coordinate form of vector ij, and let θ_{ij} denote the angle of vector ij in the local coordinate system Σ_i.

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

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

By substituting p_{ik}^c and p_{ij}^c into the above equations in place of p_k^c, p_i^c and p_j^c, we obtain the following equations.

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

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

The equations above can be reformulated into constraints on the magnitude and the angular components as follows.

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

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

The constraints precisely correspond to the similarity triangle constraints among nodes i, j, and k. Therefore, the aforementioned linear equations in the form of complex numbers can equivalently represent the shape constraints of triangle jik. Whether linear equations or linear equations in the complex number form are used, both can equivalently represent the constraints of triangle jik. In the subsequent text, we will use linear equations in the form of complex numbers to represent the pose constraints among three nodes, as the complex number representation is relatively more concise, requiring only a single equation. Thus, the coordinates of any node i are expressed using the complex number notation as $p_i^c = x_i + y_i i$.

2.2 The case with multiple neighboring nodes

If the graph $G = [V, E]$ has all its nodes distributed along a straight line (e.g., $1 - 2 - 3 - \ldots - n$), where each node has exactly two neighboring nodes, it is possible to use a system of linear equations to represent the similar triangle constraints between any three consecutive nodes. However, in practice, the graph $G = [V, E]$ may exhibit a more general structure, where node i can have multiple neighboring nodes.

Let the set of neighboring nodes for node i be $j \in N_i$. If node i is required to form a system of linear equations with any two of its neighboring nodes $j_1 \in N_i$ and $j_2 \in N_i$, then the number of linear constraints associated with node i will be $C_2^{|N_i|}$. Therefore, we can adopt a more concise approach to selecting different neighbors $j \in N_i$ to form linear constraints with node i, rather than constructing constraints with every pair of neighboring nodes. This selection strategy needs to ensure that the total number of constraints is minimized while also fairly utilizing each neighboring node $j \in N_i$ as much as possible.

If node i has two neighboring nodes, denoted as node j and node k, we can construct a linear equation using the pose transformation T_{ij} between node i and node j, and the pose transformation T_{ik} between node i and node k.

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

If node i has multiple neighboring nodes $j \in N_i$, the processing method is as follows: in SLAM algorithms, nodes represent keyframes, and there exists a sequential relationship between keyframes. From the first keyframe to the nnn-th keyframe, a connected trajectory is formed ($1 - 2 - 3 - \ldots - n$). Among the multiple neighboring nodes $j \in N_i$ of node i, two nodes are identified as the predecessor node j_{prev} and the successor node j_{next} (which may not yet exist). A virtual node v_i is then constructed using the predecessor node j_{prev} of node i, and the coordinate of this virtual node is denoted as p_{vi}^c.

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

Let p_{prev}^c represent the complex coordinate of the predecessor node j_{prev} of node i. ρ_{ij} is the distance between node i and its predecessor node j_{prev}, and we use ρ_{ij}^{prev} to normalize the vector ij. Thus, the direction of the virtual node $v_i \in N_i$ is identical to that of the vector ij_{prev}, and its magnitude satisfies $||p_{vi} - p_i|| = 1$. Let the pose transformation from node iii to its predecessor node j_{prev} be $T_{ij}^{prev} \in R^{3\times3}$, where the translation vector is $t_{ij}^{prev} = \left[x_{ij}^{prev}, y_{ij}^{prev}\right]^T$. Then, the length ρ_{ij}^{prev} is calculated as follows.

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

Let the pose transformation matrix from node i to its neighboring node j be denoted as T_{ij}, where the translation vector is $t_{ij} = \left[x_{ij}, y_{ij}\right]^T$. After introducing a virtual node p_{vi}, a linear equation constraint is formulated for each neighboring node $j \in N_i$ and $j \neq j_{prev}$, by sequentially constructing the "virtual node p_{vi} - node i - node j" relationship as follows. For the virtual node p_{vi} and the neighboring node j_{prev}, the equation is formulated using the expression [x] as the linear equation.

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

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

The constants $\rho_{vij}, \theta_{vij}, j \in N_i$ and $j \neq j_{prev}$ are computed using the following equations.

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

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

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

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

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

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

Let the neighbors of node i be denoted as $j \in N_i$. We construct a linear equation for each neighboring node j in conjunction with a virtual node v_i, ensuring a more equitable mutual influence among all nodes. The resulting system of linear equations is ultimately solved using the least squares method, with each linear equation corresponding to an error term. If the pose transformation of a specific neighboring node $j_1 \in N_i$ changes, this change is first propagated to the virtual node v_i through the associated error term. Subsequently, the virtual node v_i evenly distributes this influence to the positions of the other neighboring nodes $j \in N_i$ via their respective error terms. Alternatively, if we construct linear equations by pairing each neighboring node $j \in N_i$ and $j \neq j_{prev}$ sequentially with another neighboring node j_{prev}, the node j_{prev} becomes particularly distinct. In this case, the error propagation between j_{prev} and other neighboring nodes differs from the rest, leading to an asymmetry in how errors are transmitted.

In SLAM algorithms, if a node i has 2 neighboring nodes, virtual nodes are not used. However, if a node has more than 2 neighboring nodes, virtual nodes are introduced. This approach can result in an indeterminate form of the linear equation system, thereby increasing the computational complexity. To address this, we add virtual nodes to all nodes, as detailed below. The virtual node v_i for node i is used to establish constraints with the neighboring nodes $j \in N_i$. Typically, the virtual node of node i is constructed using its predecessor node $j_{prev} \in N_i$, as expressed by the following formula: [x]. In SLAM algorithms, we obtain a sequence of consecutive nodes $(1-2-3-\ldots-n)$ through keyframes. For the first node, there is no predecessor node, only a successor node (node 2). Therefore, for the first node, we use the successor node (node 2) to construct the coordinates p_{v1}^c of its virtual node. The specific formula is as follows: [x].

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

We define a graph $G = [V, E]$ with nnn nodes, where the position vectors of the nnn nodes are represented as $p = [p_1^c, p_{v1}^c, \ldots, p_n^c, p_{vn}^c]^T \in C^{2n}$. In the position vector ppp, each node iii has a corresponding coordinate p_i^c, which is associated with a virtual node coordinate p_{vi}^c. Based on this, we can construct a set of linear equations by combining the linear relationships of all the nodes, resulting in the following system of linear equations.

$$Lp = 0$$

The matrix L represents the matrix corresponding to all linear equation combinations. In [2], all the neighboring nodes $j \in N_i$ of node i form a set of linear equations, each node corresponding to a single linear equation. As a result, the matrix L is a square matrix. The purpose of this approach in the referenced paper is to enable distributed computation at each node. However, in the SLAM (Simultaneous Localization and Mapping) problem, the nodes represent keyframes rather than individual robots. Therefore, due to the requirements of distributed computation, the complete constraint relationship between node i and its neighboring nodes $j \in N_i$ is discarded. Furthermore, in this paper, the number of neighboring nodes $j \in N_i$ for node i is $|N_i|$, resulting in $|N_i|$ linear equations corresponding to node i.

2.3 The case of node overlap

In the graph $G = [V, E]$, if the distance between two nodes is extremely small (almost overlapping), it increases the error in the constraints of the linear equations. This situation often occurs when the robot undergoes pure rotation without translation. Below, we provide a specific example of node overlapping. As shown in the figure, nodes $1-4$ are arranged in a straight line, where the distance between node 2 and node 3 is extremely small. We can represent the similar triangle formed by "Node 1–Node 2–Node 3" using linear equations. Alternatively, we can equivalently represent the constraint of the original similar triangle as $\rho_{123} = \rho_{23}/\rho_{21}$ (a very small value). Similarly, for the similar triangle formed by "Node 2–Node 3–Node 4," it can also be expressed using linear equations, or equivalently as $\rho_{234} = \rho_{34}/\rho_{32}$ (a very large value).

Now, assuming the length between Node 1 and Node 2 (ρ_{12}) is known, we need to calculate the length between Node 3 and Node 4 (ρ_{34}). The calculation based on the similar triangle constraints is as follows.

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

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

In this calculation process, due to the fact that the parameter ρ_{123} is extremely small, the computed value of ρ_{23} also becomes very small, which can result in rounding errors caused by noise. Meanwhile, the parameter ρ_{234} is extremely large, which can amplify these rounding errors into ρ_{34}. Consequently, if two nodes coincide, rounding errors may occur during the computation. Furthermore, when the distance between two nodes is very small, the measurement of this distance may be affected by the precision limitations of the sensor, leading to measurement errors. These measurement errors can be further amplified by the extremely small parameter ρ_{123} and the extremely large parameter ρ_{234}.

To address this issue, we propose the following solution: Suppose node i has two neighboring nodes, j and k. We assume that the distance between node i and node k is extremely small.

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

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

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

Since the distance ρ_{ik} is very small, the parameter ρ_{jik} approaches zero. Thus, the above equation degenerates to $p_k = p_i$. In SLAM algorithms, we obtain a sequence of consecutive keyframes that form multiple nodes connected in a line as follows: $n_1 - m_1 - m_2 - \ldots - m_k - n_2$. Among these nodes, the k nodes from m_1 to m_k overlap. Our general approach is to treat the overlapping nodes m_1 to m_k as a single node m during position calculations. In subsequent

angle calculations, the corresponding angles for the k local coordinate systems of nodes m1 to mk are computed separately. For the case of overlapping nodes mentioned above, we analyze the situation in detail as follows.

When the SLAM algorithm adds a new node m_1, it detects that the distance between the node m_1 and its predecessor node n_1, $\rho_{m1,n1} > \epsilon_{thres}$. Therefore, nodes m_1 and n_1 are not overlapping. Here, the parameter ϵ_{thres} serves as the threshold for determining whether nodes overlap. We then compute the virtual node position $p^c_{v,m1}$ using nodes m_1 and n_1.

When the algorithm adds a new node m_2, it detects that the distance between node m_2 and its predecessor node m_1, denoted as $\rho_{m2,m1}$, satisfies $\rho_{m2,m1} < \epsilon_{thres}$. In this case, nodes m_2 and m_1 are considered overlapping nodes. At this point, m_2 and m_1 are treated as the same node. Therefore, no new node m_2 or its corresponding virtual node will be added. Similarly, if the algorithm detects any overlapping node in the set $S = \{m_i, i \in [1, k]\}$, where $m_i, i \in [2, k]$ is included, no new node will be added.

When the algorithm adds a new node n_2, it detects that the distance between node n_2 and its predecessor node m_k, denoted as $\rho_{n2,mk}$, satisfies $\rho_{n2,mk} > \epsilon_{thres}$. In this case, nodes n_2 and m_k are not overlapping nodes. At this point, the algorithm uses node m_1 (since all nodes in the overlapping set S are replaced by the first node m_1) and node n_2 to calculate the position of the virtual node $p^c_{v,n2}$.

In the graph $G = [V, E]$, there exist n nodes, each corresponding to a keyframe. A trajectory connecting the first keyframe to the nth keyframe can be constructed as a sequence $(1 - 2 - 3 - \ldots - n)$. To clearly illustrate the process of transforming the pose relationships between nodes in SLAM into a system of linear equations, we present the specific methodology in the form of pseudocode.

The process of constructing the system of linear equations is divided into two parts: adding keyframes (Process 1) and adding pose constraints (Process 2). These two processes are carried out simultaneously. When adding a keyframe (node i), a set of pose transformations between node i and its predecessor node i-1 will necessarily be added. All other pose transformations, apart from these, are categorized under "adding pose constraints (Process 2)".

Process 1. add keyframes

Input: The pose transformation $T_{i,i-1}$ between the new node i and its predecessor node $i - 1$.

Output: The node position vector $p = [p^c_1, p^c_{v1}, \ldots, p^c_n, p^c_{vn}]^T$, and the matrix L.

If(the node position vector p is empty):

Add the variable p^c_1 to the position vector: $p = [p^c_1]$.

Elseif(the node position vector p contains only one variable, $p = [p^c_1]$):

If(the new node i and its predecessor node i−1 are coincident, $\rho_{i,i-1} < \epsilon_{thres}$):

Let the set of coincident nodes containing the predecessor node i−1 be S.

Add node i to the set S.

Else:

Add three variables $p^c_{v1}, p^c_2, p^c_{v2}$ to the position vector p.

Compute the virtual node position p^c_{v1} using nodes 2 and 1: $p^c_{v1} - p^c_1 = (p^c_2 - p^c_1)/\rho_{1,2}$, add this equation to the matrix L.

Compute the virtual node position p^c_{v2} using nodes 1 and 2: $p^c_{v2} - p^c_2 = (p^c_1 - p^c_2)/\rho_{2,1}$, add this equation to the matrix L.

Else:

If(the new node i and its predecessor node i−1 are coincident, $\rho_{i,i-1} < \epsilon_{thres}$):

Let the set of coincident nodes containing the predecessor node i−1 be S.

Add node i to the set S.

Else:

Add two variables p^c_i, p^c_{vi} to the position vector p: $p = [p^T, p^c_i, p^c_{vi}]^T$.

Construct the virtual node position p^c_{vi} using nodes i and i−1: $p^c_{vi} - p^c_i = (p^c_{i-1} - p^c_i)/\rho_{i,i-1}$, add this equation to the matrix L.

Process 2. add pose constraints

Input: The pose transformation from node i to node k, T_{ik}.

Output: The adjacent positions of nodes $p = [p^c_1, p^c_{v1}, \ldots, p^c_n, p^c_{vn}]^T$, and the matrix L.

Assume the virtual node position of node i is p^c_{vi}, and the position of the virtual node in the local coordinate system Σ_i is t_{iv}.

The pose transformation from node i to node k is represented as: $T_{ik} = \begin{bmatrix} R_{ik} & t_{ik} \\ 1 & 0 \end{bmatrix}$, where the translation vector is t_{ik}.

Using the virtual node i_v, node i, and node k, a linear equation is constructed as: $p^c_k - p^c_i = \omega_{vik}(p^c_{vi} - p^c_i)$, where$\omega_{vik} = f(t_{iv}, t_{ik})$.

Assume the virtual node position of node k is p^c_{vk}, and the position of the virtual node in the local coordinate system Σ_k is t_{kv}.

The pose transformation from node k to node i is represented as: $T_{ki} = (T_{ik})^{-1} = \begin{bmatrix} R_{ki} & t_{ki} \\ 1 & 0 \end{bmatrix}$, where the translation vector is t_{ki}.

Using the virtual node k_v, node k, and node i, another linear equation is constructed as: $p_i^c - p_k^c = \omega_{vki}(p_{vk}^c - p_k^c)$, where $\omega_{vki} = f(t_{kv}, t_{ki})$.

2.4 The localizability of nodes

We define a graph $G = [V, E]$ with nnn nodes, where the position vector corresponding to the nnn nodes is given as $p = [p_1^c, p_{v1}^c, \ldots, p_n^c, p_{vn}^c]^T \in C^{2n}$. In the position vector p, each node i has a coordinate p_i^c, which is associated with a corresponding virtual node coordinate p_{vi}^c. By combining the linear equations for all the nodes as described above, we can construct the following system of linear equations.

$$Lp = 0$$

Where matrix $L \in C^{m \times 2n}$ is the coefficient matrix of the linear equations. The linear system of equations corresponds to the similarity triangle constraints among multiple nodes in the graph $G = [V, E]$. Under these constraints, the entire graph can undergo scaling transformations. Additionally, since the global coordinates of any point $i \in V$ are not determined, the entire graph can rotate and translate freely in 2D space. Considering that the n nodes in the graph $G = [V, E]$ are constructed from continuous keyframes, all nodes can be connected sequentially in a linear order $(1 - 2 - 3 - \ldots - n)$. Now, we assume that the position of node 1 in the global coordinate system Σ_g is $p_1^c = 0 + 0 \times i$, and the position of node 2 is $p_2^c = \rho_{12} + 0 \times i$. In this way, the entire graph is anchored by two nodes (node 1 and node 2), preventing rotation and translation. Simultaneously, the scale of the entire graph is fixed by the distance ρ_{12} between node 1 and node 2.

After establishing the global coordinates of Node 1 and Node 2, we need to determine whether the linear equation $Lp = 0$ has a unique non-zero solution. This question pertains to the localizability of the nodes. If the linear equation does not have a unique solution, the positions of certain nodes could be arbitrary. Below, we will prove that this problem has a unique non-zero solution from two perspectives.

First, we can use the network localizability conditions presented in [2] for the proof. The conditions for a graph $G = [V, E]$ to be self-localizable are as follows:

(NS-1) The network contains at least two anchor nodes.

(NS-2) For each free node, there exist two disjoint paths connecting it to the two anchor nodes.

We have satisfied Condition 1, where the network contains two anchor nodes. For Condition 2, we consider the subgraph $G_{sub} = [V, E_{sub}]$. In this subgraph G_{sub}, only the edges corresponding to the pose transformation constraints between two consecutive nodes from node 1 to node n are retained. Assume that node i has two neighboring nodes, denoted as node j and node k. We are given the pose transformations T_{ij} from node i to node j, and T_{ik} from node i to node k. By converting the aforementioned pose constraints into similar triangle constraints, we observe that there exist edges between nodes i, j, and k, forming a triangle. Based on this, we define the edges corresponding to the pose transformations among the four nodes as follows.

In the figure, Node 1 and Node 2 are anchor nodes, while Node 3 and Node 4 are free nodes that need to be localized. In fact, this figure can be further extended to form a graph composed of connections between Node 1 through Node n. For Node 3, we can directly identify two disjoint paths leading to the two anchor nodes. For Node 4, the first path is $4 - 3 - 1$, and the second path is $4 - 2$. Similarly, for Node n, we can find two such interleaved paths leading to the two anchor nodes. Therefore, for each free node, there exist two disjoint paths that connect it to the two anchor nodes. Since the subgraph $G_{sub} = [V, E_{sub}]$ is localizable, a graph $G = [V, E]$ with more edges is also localizable.

We then use geometric transformations to prove that each node is localizable. In the graph $G = [V, E]$, there are nnn nodes, sequentially labeled as $1, 2, \ldots, n$. Let the global coordinates of node 1 be $p_1 \in R^2$. If the pose transformation from node 1 to node 2 is T_{12}, then the global coordinates of node 2 are given by:

$$p_2 = T_{12} p_1$$

By analogy, if the global coordinates of node i are known as $p_i \in R^2$, and the pose transformation from node i to node $i + 1$ is $T_{i,i+1}$, then the global coordinates of node $i + 1$ can be expressed as:

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

Given the global coordinates of node p_1 and the relative pose transformations $T_{i,i+1}$ between any two adjacent nodes p_i, p_{i+1}, we can sequentially derive the global coordinates of all nodes. Geometrically, the global coordinates of any node can be determined. Furthermore, for any three nodes i, j, k, if there exist two relative pose transformations T_{ij} and T_{ik}, then the shape and size of the triangle ijk are uniquely determined. Here, we designate nodes 1 and 2 as anchor nodes, and assume that the n nodes are sequentially connected into a linear structure. Under this assumption, any node $i \in V$ can form a triangular constraint with its two preceding nodes $(i - 1, i - 2)$. These triangular similarity constraints are then converted into an equivalent system of linear equations. This system of equations has a unique solution.

3. Calculation of map scale

We define a graph $G = [V, E]$ with nnn nodes, where the position vector corresponding to the nnn nodes is represented as $p = [p_1^c, p_{v1}^c, \ldots, p_n^c, p_{vn}^c]^T \in C^{2n}$. In the position vector p, each node i has a corresponding coordinate p_i^c, and there is an associated virtual node coordinate p_{vi}^c. By linearly combining all nodes as described above, we obtain the following system of linear equations.

$$Lp = 0$$

The system of linear equations corresponds to the constraints of similar triangles among multiple nodes in the graph $G = [V, E]$. Therefore, under the constraints of this system, the entire graph can undergo scaling. Additionally, since the global coordinates of any point $i \in V$ are not fixed, the entire

graph can rotate and translate freely in the two-dimensional space. Because the nnn nodes in $G = [V, E]$ are constructed through a sequence of consecutive keyframes, all the nodes can be connected in a linear sequence $(1 - 2 - 3 - \ldots - n)$. Let us define the position of node 1 in the global coordinate system Σ_g as $p_1^c = 0 + 0 \times i$, and the position of node 2 as $p_2^c = \rho_{12} + 0 \times i$. In this way, the entire graph is fixed in place by two anchor nodes (node 1 and node 2), preventing it from undergoing rotation and translation. At the same time, the scale of the entire graph is determined by the distance ρ_{12} between nodes 1 and 2.

However, in the process of calculating the map scale described above, we only use the distance ρ_{12} between node 1 and node 2 to determine the overall size of the map. If there is an error in ρ_{12}, the entire map will scale accordingly. The issue of scaling the graph $G = [V, E]$ under linear equation constraints arises because we have relaxed the pose transformation constraints into similarity triangle constraints. To address this, we can adopt the following method to determine a more accurate map scale.

Let the coordinates of node 1 be denoted as $p_1^c = 0 + i \times 0$, and the coordinates of node 2 as $p_2^c = x + i \times 0$. Then, we divide the position vector of the n nodes, $p \in R^{2n}$, into two parts: $p_a = [p_1^c, p_2^c]^T$ and $p_s = [p_{v1}^c, p_{v2}^c, p_3^c, p_{v3}^c, \ldots, p_n^c, p_{vn}^c]^T$. Here, the anchor nodes $p_a \in R^2$ are treated as constants (even though there is an unknown variable x), while the free nodes $p_s \in R^{2n-2}$ are considered as unknowns. Thus, the linear equation $Lp = 0$ can be written as follows.

$$Bp_a + Hp_s = 0$$

The least squares solution of the equation is as follows.

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

We remove the virtual nodes from the node coordinates $p_s \in R^{2n-2}$, leaving the remaining coordinates as $ps_{sub} = [p_3^c, p_4^c, \ldots, p_n^c]^T \in R^{n-1}$. Then, we define the node coordinates as $p_{sub} = \left[p_1^c, p_2^c, ps_{sub}^T\right]^T \in R^n$. The vector p_{sub} represents the coordinates of the n nodes in the graph $G = [V, E]$. Since the matrix Φ is constant and p_a contains only one unknown variable x, the coordinates of each node in p_{sub} are linear functions of x. For example, $p_1^c = (a_1 x + b_1) + i \times (c_1 x + d_1), \ldots, p_n^c = (a_n x + b_n) + i \times (c_n x + d_n)$.

In the graph $G = [V, E]$, the set of pose transformation constraints is defined as $Q = \{T_{ij} = \begin{bmatrix} R_{ij} & t_{ij} \\ 1 & 0 \end{bmatrix}, i, j \in V\}$, where T_{ij} represents the pose transformation from node i to node j. Here, $t_{ij} = \left[x_{ij}, y_{ij}\right]^T$ is the translation vector. Based on the observed values, the distance between nodes i and j is given as follows.

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

From the coordinate vector p_{sub}, the distance between nodes iii and jjj can be expressed as:

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

Given that the graph $G = [V, E]$ contains $|Q|$ pose transformations, we can construct the following error function.

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

To minimize the error function J, we compute its derivative.

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

Subsequently, the three roots x_1, x_2, x_3 are determined using the cubic equation formula. After discarding any complex roots among them, the second derivative of J is used to evaluate the remaining roots.

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

If the second derivative of root x_i is positive, then x_i is the minimum of the function J; if the second derivative of root x_i is negative, then x_i is a local maximum of the function J. Once the optimized map scale x is obtained, the global coordinates of each node can be calculated using the following equation.

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

The global coordinate system p_s uses the coordinates of the first and second nodes $p_a = [0, x + 0 \times i]^T$ as anchor nodes. We only need to calculate the coordinates of each node relative to a specific anchor node to determine the fixed shape of the entire map. If it is necessary to use other nodes as anchor nodes, the entire map can be adjusted through corresponding rotation and translation transformations.

4. Transformation of different sensors

In this section, we discuss how data obtained from different types of sensors can be converted into pose transformations between various nodes. Pose transformations between two frames can be derived using cameras, LiDAR, IMU sensors, GPS sensors, wheel odometers, and loop closure detection. The following subsections will discuss the pose transformations obtained from each type of sensor in detail.

4.1 Non-GPS sensors

We first discuss the case of the camera and LiDAR sensors. These two types of sensors can obtain a determined pose

transformation T_{ij} between two specific nodes (e.g., nodes i and j). Typically, nodes iii and jjj correspond to two adjacent keyframes or two keyframes that are very close to each other.

For IMU sensors and wheel odometers, these sensors usually provide the pose transformation $T_{i-1,i}$ between two consecutive keyframes.

Similarly, in the loop closure detection module, the pose transformation T_{ij} between two nodes (nodes i and j) can also be obtained. In this case, nodes i and j are typically two adjacent nodes found after the robot completes a loop and returns to the starting point. Although nodes i and j are separated by a long time interval in the robot's motion, they are spatially close to each other.

4.2 GPS sensors

The GPS sensor presents a unique case as it provides data in the form of latitude and longitude coordinates. Using specific formulas, these coordinates can be converted into precise distance values. Each position detected by the GPS sensor is defined within a fixed latitude-longitude coordinate system, or in other words, the GPS sensor operates within a fixed global coordinate system. In contrast, other types of sensors, such as cameras and LiDAR, can only obtain the relative pose transformation T_{ij} between two specific nodes (e.g., node i and node j). To calculate the pose transformation between two distant nodes (e.g., node i and node $i+n$), it is necessary to accumulate the transformations step by step, as follows: $T_{i,i+n} = T_{i,i+1}T_{i+1,i+2}\dots T_{i+n-1,i+n}$. However, this cumulative process introduces a compounding error into the resulting transformation $T_{i,i+n}$. For GPS sensors, since they operate within a fixed latitude-longitude coordinate system, the translational component $t_{i,i+n}$ between two distant nodes (e.g., node i and node $i+n$) remains highly accurate. This high level of accuracy in GPS data plays a crucial role in maintaining the overall shape and structure of large-scale maps, making GPS sensors particularly valuable for ensuring the global consistency of large maps.

Another special case of GPS sensors is that GPS data only provides displacement without any rotational information. Suppose we obtain GPS data for node i (its position in the latitude-longitude coordinate system) and then acquire GPS data for node j. In this case, we can only determine the translation vector $t_{ij} \in R^2$ from node i to node j in the latitude-longitude coordinate system, without any rotational information. Consequently, the translation vector $t_{ij} \in R^2$ obtained from the GPS sensor cannot be combined with the pose transformation $T_{ij} \in R^{3\times3}$ obtained from other sensors to construct linear equation constraints. Although it is possible to establish nonlinear pose graph constraints using the GPS data, this approach differs from the linear equation-based optimization algorithm we aim to develop. However, since all GPS data is measured in the same coordinate system (latitude-longitude coordinate system), the GPS data of any three nodes (nodes i, j, and k) can be used to construct a linear equation.

In the graph $G = [V, E]$, there are nnn nodes (keyframes). The update frequency of GPS data is typically much lower than that of camera sensors or IMU sensors. Therefore, not every node has corresponding GPS data. We define the index set of nodes with available GPS data as $G_{index} = \{g_i, i \in [1, m]\}$. Each index value $g_i \in [1, n]$ within this set corresponds to a node, and the set contains mmm elements, representing mmm GPS latitude and longitude data points. Let the set of these mmm GPS latitude and longitude data points be $G = \{pos_{g(i)} = [lon_{g(i)}, lat_{g(i)}]^T, g(i) \in [1, n], i \in [1, m]\}$.

Let the GPS latitude and longitude data of the i-th point be represented as $pos_{g(i)} = [lon_{g(i)}, lat_{g(i)}]^T$, and that of the j-th point as $pos_{g(j)} = [lon_{g(j)}, lat_{g(j)}]^T$. Using either the spherical distance formula or the UTM planar coordinate system, the latitude and longitude data of nodes g_i and g_j can be transformed into a relative translation vector.

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

Specifically, the function f converts the latitude and longitude of two nodes, $pos_{g(i)}$ and $pos_{g(j)}$, into a translation vector $t_{g(i),g(j)}$.

After acquiring the GPS sensor data, we construct two types of linear equations: one for adjacent GPS data and another for cross-node GPS data. Below, we first introduce the linear equations for adjacent GPS data.

Given that the i-th GPS data is denoted as $pos_{g(i)}$, the (i+1)-th GPS data as $pos_{g(i+1)}$, and the (i+2)-th GPS data as $pos_{g(i+2)}$, we use the node g_{i+1} as an intermediate node. The relative position from node g_{i+1} to node g_i is expressed as $t_{g(i+1),g(i)} = f(pos_{g(i+1)}, pos_{g(i)})$. Similarly, the relative position from node g_{i+1} to node g_{i+2} is given by $t_{g(i+1),g(i+2)} = f(pos_{g(i+1)}, pos_{g(i+2)})$, and the relative position from node g_{i+1} to node g_{i+2} is represented as $t_{g(i+1),g(i+2)} = f(pos_{g(i+2)}, pos_{g(i+1)})$. First, it is necessary to ensure that there are no overlapping nodes among these three nodes. The method for determining this is as follows.

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

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

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

After ensuring that there are no overlapping nodes, the following linear equations can be constructed.

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

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

The parameters ρ_{i+1} and θ_{i+1} can be calculated as follows.

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

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

In addition, we can construct linear equations based on GPS data across distant nodes. The translation vector $t_{g(i),g(i+1)}$

calculated using any two adjacent GPS data points cannot fully utilize the information provided by GPS. This is because the advantage of GPS lies in its ability to maintain high accuracy in measuring the relative positions between two nodes that are far apart. Therefore, in addition to constructing the aforementioned linear equations, it is also necessary to use GPS data to construct linear equations for nodes that are separated by significant distances.

The given set of GPS latitude and longitude data is defined as $G = \{pos_{g(i)} = [lon_{g(i)}, lat_{g(i)}]^T, g_i \in [1, n], i \in [1, m]\}$. We propose a method to pair the first GPS data point $pos_{g(1)}$ with any other GPS data point $pos_{g(i)}, i \in [2, m]$, and compute the translation vector $t_{g(1),g(i)}, i \in [2, m]$. However, this approach has the following issue: if the first GPS data point $pos_{g(1)}$ contains a significant error, then all the computed translation vectors $t_{g(1),g(i)}$ will also be heavily affected by this error. Therefore, when calculating the pose transformations, it is not advisable to rely entirely on a single GPS data point. Instead, all GPS data points should be utilized in a manner that ensures approximately uniform contribution across the dataset.

Then, we propose whether it is feasible to randomly select two data points from the GPS dataset, denoted as $pos_{g(i)}$ and $pos_{g(j)}$, to compute the translation vector $t_{g(i),g(j)}$. Subsequently, we use the GPS data of the adjacent node g_{i+1} and g_i, namely $pos_{g(i+1)}$ and $pos_{g(i)}$, to calculate the translation vector $t_{g(i),g(i+1)}$. Finally, we construct a linear equation system using the nodes g_i, g_j and g_{i+1}.

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

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

The proposed scheme has the following issues: nodes g_i and g_j are randomly selected, meaning that the distance between the two data points $pos_{g(i)}$ and $pos_{g(j)}$ is typically quite large (since the two points are randomly chosen across the entire map). Consequently, the magnitude of the translation vector $t_{g(i),g(j)}$ will be significantly greater than the magnitude of the translation vector $t_{g(i),g(i+1)}$, i.e., $\rho_{g(i),g(j)} \gg \rho_{g(i),g(i+1)}$. The equivalent constraint of similar triangles corresponding to the above linear equations is as follows.

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

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

Because $\rho_{g(i),g(j)} \gg \rho_{g(i),g(i+1)}$, the value of $\rho_{g(i+1),g(i),g(j)}$ is significantly large. When using the least squares method to solve the linear system $Lp = 0$, the above linear equation corresponds to an error term. To reduce this error term, there is a tendency to adjust the coordinates of node g_{i+1}. This is because the coefficient $\rho_{g(i+1),g(i),g(j)}$ associated with node g_{i+1} is very large, and a slight adjustment to g_{i+1} can significantly reduce the error term. Conversely, there is less inclination to adjust node g_j, as the coefficient associated with g_j (which is 1) is much smaller than $\rho_{g(i+1),g(i),g(j)}$. A substantial movement of node g_j would be required to achieve a similar reduction in the error term.

From a geometric perspective, nodes g_{i+1}, g_i and g_j form a constraint based on similar triangles. Since the distance between node g_j and node g_i (denoted as $\rho_{g(i),g(j)}$) is significantly larger than the distance between node g_i and node g_{i+1} (denoted as $\rho_{g(i),g(i+1)}$), there is a tendency to slightly adjust the position of node g_{i+1} in order to satisfy the properties of similar triangles as much as possible. As a result, although the linear equation constraint is constructed based on the distant nodes g_i and g_j, the actual geometric effect of this constraint is primarily a local adjustment to the position of g_{i+1} near g_i. Consequently, such a linear equation fails to impose a meaningful constraint on the global shape of the map.

We need to identify three nodes (node g_i, node g_j, and node g_k) in the GPS data set G, where these three nodes are distinct and do not overlap.

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

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

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

Additionally, the distances between these three nodes should be approximately equal, which can be described using the following formula.

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

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

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

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

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

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

Here, p_{thres} represents the threshold for the side length ratio. We require that in the triangle formed by $g_i g_j g_k$, the ratio of any two sides (longer side to shorter side) must be less than p_{thres}.

Since nodes g_i, g_j and g_k are randomly selected across the entire map, the distances between these three points are typically large. Using these three points to construct a similar triangle and form a system of linear equations provides a strong constraint on the overall shape of the map. To ensure that the similar triangle formed by these three points is preserved, the positions of nodes g_i, g_j and g_k must undergo uniform and significant adjustments.

To maximize the number of such triangular combinations in the GPS data set G, we adopt the following approach: randomly select three points from the GPS data set and check whether they satisfy the required edge ratio conditions. If the condition is not met, another set of points is selected. We can

define a maximum search limit, f_{max}, and the required number of triangles, Δ_{req}.

Loop closure detection can also be used to obtain the pose transformation T_{ij} between two points (nodes i and j). These two points are separated by a long temporal distance during the robot's motion. However, will the aforementioned issue with GPS data also arise in this case? This should not occur because, although the two points obtained through loop closure detection are far apart in time, they are spatially close. Let the magnitude of the translation vector t_{ij} of the pose transformation T_{ij} be ρ_{ij}. Additionally, let the distance from node i to the virtual node v_i be ρ_{iv}. The lengths ρ_{ij} and ρ_{iv} should not differ significantly. Hence, the problem of similar triangle shapes mentioned earlier will not arise in this situation.

After obtaining three nodes g_i, g_j and g_k that meet the required conditions, we can construct the following linear equations.

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

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

For the GPS data set G, we construct two types of linear equations: linear equations for adjacent GPS data and linear equations for cross-node GPS data. Below, we first introduce the linear equations for adjacent GPS data. The specific methods are demonstrated using pseudocode as follows.

Process 3. constructing linear equations using adjacent GPS data

Input: A set of GPS data $G = \{pos_{g(i)} = [lon_{g(i)}, lat_{g(i)}]^T, g_i \in [1,n], i \in [1,m]\}$, threshold for overlapping nodes ϵ_{thres}

Output: Adjacent node positions $p = [p^c_1, p^c_{v1}, \ldots, p^c_n, p^c_{vn}]^T$, matrix L

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

Compute the relative translation between nodes g_i and g_{i+1}, $t_{g(i),g(i+1)} = f(pos_{g(i)}, pos_{g(i+1)})$, with distance $\rho_{g(i),g(i+1)}$.

Compute the relative translation between nodes g_{i+1} and g_{i+2}, $t_{g(i+1),g(i+2)} = f(pos_{g(i+1)}, pos_{g(i+2)})$, with distance $\rho_{g(i+1),g(i+2)}$.

Compute the relative translation between nodes g_i and g_{i+2}, $t_{g(i),g(i+2)} = f(pos_{g(i)}, pos_{g(i+2)})$, with distance $\rho_{g(i),g(i+2)}$.

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

Construct a linear equation, $p^c_{g(i+2)} - p^c_{g(i+1)} = \omega_{i+1}(p^c_{g(i)} - p^c_{g(i+1)})$, add this equation to matrix L.

Process 4. constructing linear equations using cross node GPS data

Input: GPS data set $G = \{pos_{g(i)} = [lon_{g(i)}, lat_{g(i)}]^T, g_i \in [1,n], i \in [1,m]\}$, edge ratio threshold k_{thres}, maximum number of iterations f_{max}, required number of triangles Δ_{req}

Output: Node adjacency positions $p = [p^c_1, p^c_{v1}, \ldots, p^c_n, p^c_{vn}]^T$, matrix L

Iteration count $iter_{num} = 0$, Number of triangles found $\Delta_{num} = 0$.

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

$iter_{num} = iter_{num} + 1$.

Randomly select three points g_i, g_j, g_k from GPS data set G.

Relative translation between nodes g_i and g_j, $t_{g(i),g(j)} = f(pos_{g(i)}, pos_{g(j)})$ with distance $\rho_{g(i),g(j)}$.

Relative translation between nodes g_i and g_k, $t_{g(i),g(k)} = f(pos_{g(i)}, pos_{g(k)})$ with distance $\rho_{g(i),g(k)}$.

Relative translation between nodes g_j and g_k, $t_{g(j),g(k)} = f(pos_{g(j)}, pos_{g(k)})$ with distance $\rho_{g(j),g(k)}$.

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

break

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

break

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

break

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

break

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

Construct the linear equation using nodes g_i, g_j and g_k, $p^c_{g(k)} - p^c_{g(i)} = \omega_{g(j),g(i),g(k)}(p^c_{g(j)} - p^c_{g(i)})$, add this equation to matrix L.

$\Delta_{num} = \Delta_{num} + 1$.

4.3 Variance of transformation

The relative pose transformation between two frames can be obtained through various sensors, including cameras, LiDAR, IMU sensors, GPS sensors, wheel encoders, and loop closure detection. Typically, the pose transformations obtained from different sensors exhibit varying degrees of variance. This variance can be predefined based on the characteristics of each sensor or derived from the pose transformation estimation process in the front-end module of the SLAM algorithm, which calculates the variance of the pose transformation T_{ij}. In the case of 2D plane pose transformations, the transformation matrix T_{ij} is a 3×3 matrix. For simplicity, we use a single variance σ^2_{ij} to represent the variance of the entire pose transformation matrix.

When constructing a linear equation using node i, node j, and node k, the pose transformation T_{ij} from node i to node j (with variance σ^2_{ij}) and the pose transformation T_{ik} from node i to node k (with variance σ^2_{ik}) are required. We assume that the constructed linear equation includes an error variable $v \sim N(0, \sigma^2_{jik})$.

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

For convenience, we define the variance of the error term v_{jik} as $\sigma^2_{jik} = \sigma^2_{ij} + \sigma^2_{ik}$. Thus, the resulting system of linear equations is as follows.

$$Bp_s + Hp_a = 0 + v$$

The error vector $v \sim N(0, \Sigma)$; we assume the independence of each linear equation, so the covariance matrix Σ is a diagonal matrix. The least squares solution to this system of equations is given as follows, and this solution can be used to compute more accurate node coordinates.

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

5. Outlier removal

We consider a graph $G = [V, E]$ with nnn nodes, where the position vectors of the nnn nodes are represented as $p = [p^c_1, p^c_{v1}, \ldots, p^c_n, p^c_{vn}]^T \in C^{2n}$. For each node iii, the coordinate p^c_i has a corresponding virtual node coordinate p^c_{vi}. By combining all the nodes into a system of linear equations, we can obtain the following linear equation system.

$$Lp = 0$$

Each linear equation is constructed based on the pose transformation T_{ij} between nodes. If there are outliers in the set of pose transformations $E_T = \{T_{ij}, i, j \in V\}$ within the graph $G_T = [V, E_T]$, the coefficients of the corresponding linear equations will also exhibit outliers. When solving the system of linear equations using the least squares method, each linear equation corresponds to an individual error term. The presence of outliers can cause significant deviations in the solution of the equations.

In nonlinear optimization-based SLAM schemes, each pose transformation T_{ij} is used to construct an error function $h(T_{ij})$ through an observation function. These error functions then form error terms $e(T_{ij}) = h(T_{ij})^2$, and the overall error function J is obtained by summing these error terms. However, a robust function is typically applied to the error terms as follows.

$$\rho(x) = \begin{cases} x^2, \text{if } -1 < x < 1 \\ x, \text{if } x \geq 1 \\ -x, \text{if } x \leq -1 \end{cases}$$

The robust function ρ increases rapidly near zero, while its growth slows significantly beyond a certain range. The error function J after incorporating the robust function can be expressed as follows.

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

Due to the presence of the robust function, even if a certain pose transformation T_{ij} is an outlier, the corresponding error term will not be excessively large. As a result, the influence of outliers on the minimum value of the overall error function is limited. Additionally, the original error term $e(T_{ij}) = h(T_{ij})^2$ is a nonlinear function, and after introducing the robust function, the modified error term$\rho\left(h(T_{ij})^2\right)$ remains nonlinear. We still use numerical optimization methods to compute the minimum value of the error function.

However, for the linear equation $Lp = 0$, we cannot adopt the robust function approach. If a robust function is applied to the error terms derived from each linear equation, the resulting function would become nonlinear. The purpose of constructing pose transformations into linear equations is to reduce computational complexity. If nonlinear equations are used, this purpose cannot be achieved.

Paper [3] proposes a method for outlier removal in pose transformations. As the method involves multiple detailed steps, we provide a brief overview of the general approach. The algorithm takes as input the set of all pose transformations in the graph $G_T = [V, E_T]$, denoted as $E_T = \{T_{ij}, i, j \in V\}$, and outputs a set of pose transformations with outliers removed, denoted as $E'_T = \{T_{ij}, i, j \in V\}$. By constructing linear equations using all pose transformations in E'_T, the algorithm effectively mitigates the influence of outliers on the equations.

For every pose transformation T_{ij} in the input set $E_T = \{T_{ij}, i, j \in V\}$, the algorithm preassigns a probability p_{ij} indicating the likelihood of the transformation being an outlier. This probability does not need to be an absolutely precise value; it can be roughly estimated using the variance σ_{ij}^2 associated with the pose transformation T_{ij}. For instance, the following approach can be employed:

$$p_{ij} = \begin{cases} 0.1, \text{if } \sigma_{ij}^2 < \sigma_{thres1}^2 \\ 0.2, \text{if } \sigma_{thres1}^2 \le \sigma_{ij}^2 < \sigma_{thres2}^2 \\ 0.3, \text{if } \sigma_{thres2}^2 \le \sigma_{ij}^2 \end{cases}$$

The probability p_{ij} is used in the outlier removal algorithm to find the path between different nodes (node i, node j); the pose transformation T_{ij} represents the edge in the path. The algorithm seeks to identify a path consisting of edges with fewer outliers to achieve the transition from node i to node j.

The general approach of this outlier removal algorithm is as follows: Assume that in $G_T = [V, E_T]$, there exist two nodes $p_1, p_2 \in V$. Between these two nodes p_1 and p_2, there are three rotation-translation matrices $T_{12}^1, T_{12}^2, T_{12}^3$.

At the same time, a connection between p_1 and p_2 can also be established through another intermediate node (e.g., p_3, p_4, p_5). Thus, the pose transformation from p_1 to p_2 can be calculated as follows.

$$T_{12}^4 = T_{13}T_{32}$$
$$T_{12}^5 = T_{14}T_{42}$$
$$T_{12}^6 = T_{15}T_{52}$$

Then, nodes p_1 and p_2 can form a connection through two additional intermediate nodes (e.g., p_6, p_7, or p_8, p_9, or p_{10}, p_{11}). The pose transformation from p_1 to p_2 is calculated as follows.

$$T_{12}^7 = T_{16}T_{67}T_{72}$$
$$T_{12}^8 = T_{18}T_{89}T_{92}$$
$$T_{12}^9 = T_{1,10}T_{10,11}T_{11,2}$$

As a result, we obtain 9 sets of pose transformations between node 1 and node 2, denoted as $T_{12}^i, i \in [1,9]$. If all the pose transformations used (denoted as the set E_T) contain no outliers, these pose transformations T_{12}^i should be highly consistent. However, if any pose transformation in E_T is an outlier, the corresponding T_{12}^i will significantly differ from the others. To address this, we can convert the pose transformations T_{12}^i into translation vectors and rotation vectors, and then apply the statistically-based IQR (Interquartile Range) algorithm to each component to remove outliers.

6. Calculation of local coordinate system

6.1 Non-overlapping nodes

We have computed the coordinates of each node in graph $G = [V, E]$ within the global coordinate system Σ_g, which are denoted as $p_{sub} = [p_1^c, p_2^c, \dots, p_n^c]^T$. Let the neighbors of node i be $j \in N_i$, where the coordinate of node j is $p_j^c = x_j + iy_i$. Using vector form, the coordinates of node j can be represented as $p_j = [x_j, y_j]^T$. The relative coordinate of node j with respect to node i is given by $p_j^r = [x_j - x_i, y_j - y_i]^T$. The relative coordinate vectors of all neighboring nodes in the global coordinate system are denoted as $P_i^{neig} = [p_{j1}^r, \dots, p_{jn}^r] \in R^{2|N_i|}, j_1, \dots, j_n \in N_i$. Let the pose transformation from node i to its neighboring node $j \in N_i$ be $T_{ij} = \begin{bmatrix} R_{ij} & t_{ij} \\ 1 & 0 \end{bmatrix} \in R^{3\times3}, j \in N_i$, where the translation vector is $t_{ij} = [x_{ij}, y_{ij}]^T \in R^2, j \in N_i$. Then, the relative position vectors of the neighbors $j \in N_i$ of node i in the local coordinate system Σ_i are expressed as $Q_i^{neig} = [t_{ij1}, \dots, t_{ijn}] = [q_{ij1}, \dots, q_{ijn}] \in R^{2|N_i|}, j_1, \dots, j_n \in N_i$.

Given a set of points $j \in N_i$ with position vectors $P_i^{neig} \in R^{2|N_i|}$ in the global coordinate system Σ_g, and their corresponding position vectors $Q_i^{neig} \in R^{2|N_i|}$ in the local coordinate system Σ_i, we aim to find a rotation matrix R_i by constructing the following error function.

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

We aim to find the rotation matrix $R_i \in R^{2\times2}$ that minimizes the error function $E(R_i)$. This problem belongs to the domain of point cloud registration, and the matrix R_i can be computed using the Singular Value Decomposition (SVD) method within the Iterative Closest Point (ICP) algorithm. The matrix R_i represents the rotation matrix that transforms a point set from the global coordinate system Σ_g to the local coordinate system Σ_i. Since the rotation of the coordinate system itself is opposite to the rotation of the point set, the rotation matrix of the local coordinate system Σ_i relative to the global coordinate system Σ_g is $(R_i)^{-1}$. The inverse rotation matrix $(R_i)^{-1}$ can be converted into an angle θ_i, which represents the orientation of the local coordinate system Σ_i with respect to the global coordinate system Σ_g.

Using the SVD method in the ICP algorithm, the rotation matrix R that minimizes the error function $E(R_i)$ is given as follows.

$$H = P_i^{neig} {Q_i^{neig}}^T$$
$$= U\Sigma V^T$$
$$R_i = VU^T$$

The singular value decomposition (SVD) of the matrix $H \in R^{2\times2}$ is performed, resulting in the orthogonal matrices U and V. The rotation matrix is computed as $R_i = VU^T$. However, if the point sets P_i^{neig} or Q_i^{neig} degenerate into a straight line, the resulting matrix R_i could be a reflection matrix (where $|R_i| = -1$). To address this, we can adjust the reflection matrix R_i into a proper rotation matrix through the following steps.

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

The complete formula for calculating the rotation matrix is as follows.

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

Process 5. Calculation of local coordinate system for non-overlapping nodes

Input: Coordinates of node i in the global coordinate system p_i^c, neighboring nodes of node i: $j \in N_i$, Coordinates of neighboring nodes p_j^c ($j \in N_i$), transformation matrices between node i and neighbors: T_ij

Output: The angle θ_i of the local coordinate system Σ_i relative to the global coordinate system Σ_g

The coordinates of neighboring nodes $j \in N_i$ relative to node i are computed as: $p_j^r = [x_j - x_i, y_j - y_i]^T$.

The relative coordinates of neighboring nodes $j \in N_i$ in the global coordinate system Σ_g are expressed as: $P_i^{neig} = [p_{j1}^r, \ldots, p_{jn}^r] \in R^{2|N_i|}, j_1, \ldots, j_n \in N_i$.

The coordinates of neighboring nodes $j \in N_i$ in the local coordinate system Σ_i are: $Q_i^{neig} = [t_{ij1}, \ldots, t_{ijn}] = [q_{ij1}, \ldots, q_{ijn}] \in R^{2|N_i|}, j_1, \ldots, j_n \in N_i$.

Rotation matrix R_i is computed using point set registration. First, define: $H = P_i^{neig} Q_i^{neig^T}$.

Perform singular value decomposition (SVD) on H: $H = U\Sigma V^T$.

The rotation matrix is then given by:
$R_i = V \times \mathrm{diag}([1, |VU^T|]) \times U^T$.

The angle θ_i of the local coordinate system Σ_i relative to the global coordinate system Σ_g is derived from the inverse of the rotation matrix $(R_i)^{-1}$.

6.2 Overlapping nodes

If node i is not an overlapping node, then it has only one local coordinate system Σ_i. However, if node i is an overlapping node, we need to compute a local coordinate system for each overlapping node. The specific method is as follows: In SLAM algorithms, we obtain a sequence of nodes connected in a line through consecutive keyframes, represented as $n_1 - m_1 - m_2 - \ldots - m_k - n_2$. Among them, the kkk nodes from m_1 to m_k overlap with one another, and we define the set of overlapping nodes as $M = \{m_i, i \in [1, k]\}$. During position computation, we treat the overlapping nodes m_1 to m_k as a single node m and compute the position p_m^c of node m in the global coordinate system Σ_g.

The set of overlapping nodes, $M = \{m_i, i \in [1, k]\}$, exhibits unique characteristics. Since the distance between node m_i and many of its neighboring nodes $j \in N_{mi}$ is zero, it is not feasible to compute the angles of the local coordinate system using point set registration. Instead, a nonlinear optimization approach can be employed, where the pose constraints of each overlapping node are used to construct an error function. By minimizing this error function, the optimized local coordinate system for each node can be obtained. However, in this paper, we aim to avoid the use of nonlinear functions to reduce computational complexity. Therefore, we choose to discard some pose constraint conditions, enabling the angles of the local coordinate system Σ_i for node m_i to be computed using linear equations instead. Our overall approach is as follows: for the endpoint nodes in the overlapping node set, specifically m_1 and m_k, the rotation matrices can still be computed using point set registration. For non-endpoint nodes m_i, where $i \in [2, k-1]$, the local coordinate system is calculated using angular constraints between overlapping nodes.

Let the neighboring nodes of the first overlapping node m_1 be denoted as $j \in N_{m1}$. The neighboring nodes $j \in N_{m1}$ can be classified into two categories. The first category includes nodes that belong to the set of overlapping nodes $M = \{m_i, i \in [1, k]\}$. Since the distance between the overlapping node m_1 and any node in the overlapping node set M is zero, no constraint information for the local coordinate system Σ_{m1} can be obtained via point set registration. Therefore, we ignore the neighboring nodes in the first category. The second category includes nodes satisfying $j \in N_{m1}$ and $j \notin M$. In the minimal case, this category contains only one neighboring node, denoted as n_1. Since the distance between m_1 and the second category of neighboring nodes is non-zero, the local coordinate system Σ_{m1} can be determined by point set registration. Let the set of second-category neighboring nodes be denoted as $N_{m1}^{part2} = \{j \in N_{m1} \text{ and } j \notin M\}$.

Below we calculate the pose of the local coordinate system Σ_{m1} using node m_1 and its neighbor node set N_{m1}^{part2}. Let the set of overlapping nodes $M = \{m_i, i \in [1, k]\}$ have coordinates in the global coordinate system $13\Sigma_g$ represented as $p_m^c = x_m + iy_m$. The coordinates of a neighbor node $j \in N_{m1}^{part2}$ in the global coordinate system Σ_g are given by $p_j^c = x_j + iy_i$. The relative coordinates of neighbor node $j \in N_{m1}^{part2}$ in the global coordinate system Σ_g are represented as $p_j^r = [x_j - x_m, y_j - y_m]^T$. The relative coordinate vector of all neighbor nodes $j \in N_{m1}^{part2}$ in the global coordinate system Σ_g is denoted as $P_{m1}^{neig} \in R^{2|N_{m1}^{part2}|}$. Similarly, the coordinate vector of neighbor nodes $j \in N_{m1}^{part2}$ in the local coordinate system Σ_{m1} is denoted as $Q_{m1}^{neig} \in R^{2|N_{m1}^{part2}|}$. Next, we use singular value decomposition (SVD) to compute the rotation matrix R_{m1} that maps the coordinate vector P_{m1}^{neig} to the coordinate vector Q_{m1}^{neig}. Consequently, the rotation matrix from the global coordinate system Σ_g to the local coordinate system Σ_{m1} is given by $(R_{m1})^{-1}$.

Let the neighboring nodes of the last overlapping node, m_k, be denoted as $j \in N_{mk}$. These neighboring nodes $j \in N_{mk}$ can be divided into two categories. The first category consists of nodes that belong to the overlapping node set $M = \{m_i, i \in$

$[1,k]\}$. Since the distance between the overlapping node m_k and any node in the set M is zero, no constraint information about the local coordinate system Σ_{mk} can be obtained through point set registration. Therefore, the first category of neighboring nodes is ignored. The second category includes nodes $j \in N_{mk}$ and $j \notin M$ (in the minimal case, there is only one neighboring node, such as node n_2). The distance between m_k and the second category of neighboring nodes is non-zero, allowing the use of point set registration to determine the angular constraints of the local coordinate system Σ_{mk}. Denote the set of second-category neighboring nodes as $N_{mk}^{part2} = \{j \in N_{mk}$ and $j \notin M\}$.

We utilize node m_k and its neighbor node set N_{mk}^{part2} to compute the pose of the local coordinate system Σ_{mk}. Let the coordinates of the overlapping node set $M = \{m_i, i \in [1,k]\}$ in the global coordinate system Σ_g be $p_m^c = x_m + iy_m$. The coordinates of a neighbor node $j \in N_{mk}^{part2}$ in the global coordinate system Σ_g are $p_j^c = x_j + iy_i$. Thus, the relative coordinates of the neighbor node $j \in N_{mk}^{part2}$ in the global coordinate system Σ_g are given by $p_j^r = [x_j - x_m, y_j - y_m]^T$. Consequently, the relative coordinate vector of all neighbor nodes $j \in N_{mk}^{part2}$ in the global coordinate system is denoted as $P_{mk}^{neig} \in R^{2|N_{mk}^{part2}|}$. Similarly, the coordinate vector of the neighbor nodes $j \in N_{mk}^{part2}$ in the local coordinate system Σ_{mk} is denoted as $Q_{mk}^{neig} \in R^{2|N_{mk}^{part2}|}$. We then apply Singular Value Decomposition (SVD) to compute the rotation matrix R^{mk} that transforms the coordinate vector P_{mk}^{neig} to Q_{mk}^{neig}. Accordingly, the rotation matrix from the global coordinate system Σ_g to the local coordinate system Σ_{mk} is given by $(R_{mk})^{-1}$.

Below, we need to compute the local coordinate system corresponding to the remaining overlapping nodes $M_{mid} = \{m_i, i \in [2, k-1]\}$. The neighboring nodes $j \in N_{mi}$ of each node m_i in the overlapping node set M_{mid} can be divided into two categories. The first category of neighboring nodes consists of nodes in the set $M = \{m_i, i \in [1,k]\}$, denoted as $N_{mi}^{part1} = \{j \in N_{mi}$ and $j \in M\}$. Typically, there exists a relative pose constraint $R_{i,i+1}$ between any two adjacent overlapping nodes (m_i and m_{i+1}). Additionally, there may also exist a relative pose constraint $R_{i,j}$ between any two overlapping nodes (m_i and m_j, where $i,j \in [1,k]$). These first-category neighboring nodes (N_{mi}^{part1}) will be used in subsequent computations of the local coordinate system Σ_{mi} for each node m_i. The second category of neighboring nodes consists of nodes that do not belong to the set M, denoted as $N_{mi}^{part2} = \{j \in N_i$ and $j \notin M\}$. When computing the local coordinate system Σ_{mi} for each node m_i in M_{mid}, we will ignore this category of neighboring nodes.

Given that the first-order neighboring nodes of node m_i in the overlapping node set M_{mid} are $j \in N_{mi}^{part1}$, and since both node m_i and node j are overlapping nodes, the pose constraint between these two nodes consists only of the rotational component R_{ij}. From the previous steps, we have already obtained the angle of the local coordinate system Σ_{m1} of node m_1 relative to the global coordinate system Σ_g, denoted as $(R_{m1})^{-1}$, which is converted to an angle θ_{m1}. Similarly, the angle of the local coordinate system Σ_{mk} of node m_k relative to the global coordinate system Σ_g is $(R_{mk})^{-1}$, converted to θ_{mk}. Let the angle of the local coordinate system Σ_{mi} of node m_i in the overlapping node set M_{mid} relative to the global coordinate system $\Sigma g\backslash Sigma\_g\Sigma g$ be θ_{mi}.

We construct linear equations for the angles using the first-order neighboring nodes $j \in N_{mi}^{part1}$ of node m_i. Since node m_i and node j are overlapping nodes, the pose transformation from m_i to j is represented by the rotation matrix $R_{mi,j}$. By converting the rotation matrix $R_{mi,j}$ into the angle $\theta_{mi,j}$, the following equation is satisfied.

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

For the set of overlapping nodes M_{mid}, there are $k-2$nodes, yielding $k-2$ groups of linear equations as follows.

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

$$\dots$$

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

The angles of node m_1 and node m_k (denoted as θ_{m1} and θ_{m2}) are known and can be substituted as given quantities into the linear equation system mentioned above. The unknown variables in this equation system are the angles θ_i of each node in the set M_{mid}, where $i \in [2, k-1]$. We rewrite the linear equation system in matrix form as follows.

$$A\theta = b$$

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

Next, we analyze whether the linear system of equations has a unique solution. If, in the set of overlapping nodes $M = \{m_i, i \in [1,k]\}$, each node m_i is constrained only by its predecessor m_{i-1} and successor m_{i+1} (the case with the minimal number of constraints), then the number of equations in the angular linear system is $k-1$, and the number of unknowns is $k-2$. Under these conditions, we can explicitly write the coefficient matrix A of the linear system and find that its rank satisfies $\mathrm{rank}(A) = k-2$. Therefore, in this minimally constrained scenario, the linear system has a unique least-squares solution. With the addition of more constraints, the linear system will still have a unique least-squares solution. The least-squares solution for this linear system is given as follows.

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

From the perspective of the graphical representation, the constraints of the aforementioned linear equation involve fine-tuning the angles of other nodes $m_i \in M_{mid}$, given the angles θ_{m1} of the known node m_1 and θ_{mk} of the known node m_k, to best satisfy the equation. If we aim to manually control the angles of each node $m_i \in M_{mid}$ in proportion, additional linear equations can be introduced as follows.

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

…

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

By adding the aforementioned linear equations and applying the least squares method to solve for the angle θ_{mi} of the node $m_i \in M_{mid}$, the result ensures that the angle θ_{mi} of the node m_i closely satisfies the proportional relationships between nodes. The parameter λ can be used to adjust the weight of this constraint.

Process 6. Calculation of local coordinate system for overlapping nodes

Input: The overlapping node set $M = \{m_i, i \in [1,k]\}$; the coordinates of the overlapping node set M in the global coordinate system: p_m^c; for each node m_i, its neighboring nodes $j \in N_i$; the coordinates of neighboring nodes $j \in N_i$ in the global coordinate system: p_j^c, where $j \in N_i$; the pose transformations T_{ij}, where $j \in N_i$

Output: The orientation θ_i of the local coordinate system Σ_i for each node $m_i, i \in [1,k]$, in the global coordinate system Σ_g

#part 1: local coordinate system of node m_1

The neighboring nodes of node m_1 are denoted as $j \in N_{m1}$. A subset of these neighboring nodes is defined as $N_{m1}^{part2} = \{j \in N_{m1} \text{ and } j \notin M\}$.

The relative coordinates of neighbors $j \in N_{m1}^{part2}$ in the global coordinate system Σ_g are given by: $p_{m1}^r = \left[x_j - x_m, y_j - y_m\right]^T$.

The relative coordinate vectors of all neighbors $j \in N_{m1}^{part2}$ in the global coordinate system Σ_g are represented as: $P_{m1}^{neig} = \left[p_{j1}^r, \ldots, p_{jn}^r\right] \in R^{2\left|N_{m1}^{part2}\right|}, j_1, \ldots, j_n \in N_{m1}^{part2}$.

The relative coordinate vectors of neighbors $j \in N_{m1}^{part2}$ in the local coordinate system Σ_{m1} are represented as: $Q_{m1}^{neig} = \left[t_{ij1}, \ldots, t_{ijn}\right] \in R^{2\left|N_{m1}^{neig}\right|}, j_1, \ldots, j_n \in N_{m1}^{part2}$.

By performing point set registration between P_{m1}^{neig} and Q_{m1}^{neig}, the rotation matrix R_{m1} is obtained.

The rotation matrix $(R_{m1})^{-1}$ is converted into the angle θ_{m1}.

#part 2: local coordinate system of node m_k

The neighbor nodes of node m_k are denoted as $j \in N_{mk}$, and the subset of partial neighbor nodes is defined as: $N_{mk}^{part2} = \{j \in N_{mk} \text{ and } j \notin M\}$.

The relative coordinates of the neighbor node $j \in N_{mk}^{part2}$ in the global coordinate system Σ_g are given by: $p_{m1}^r = \left[x_j - x_m, y_j - y_m\right]^T$.

The relative coordinate vector of the neighbor nodes $j \in N_{m1}^{part2}$ in the global coordinate system Σ_g is represented as: $P_{mk}^{neig} = \left[p_{j1}^r, \ldots, p_{jn}^r\right] \in R^{2\left|N_{mk}^{part2}\right|}, j_1, \ldots, j_n \in N_{mk}^{part2}$.

The relative coordinate vector of the neighbor nodes $j \in N_{mk}^{part2}$ in the local coordinate system Σ_{mk} is denoted as: $Q_{mk}^{neig} = \left[t_{ij1}, \ldots, t_{ijn}\right] \in R^{2\left|N_{mk}^{neig}\right|}, j_1, \ldots, j_n \in N_{mk}^{part2}$.

Using point set registration, P_{mk}^{neig} and Q_{mk}^{neig} are aligned to obtain the rotation matrix R_{mk}.

The rotation matrix $(R_{mk})^{-1}$ is then transformed into the angular parameter θ_{mk}.

#part 3: calculating the local coordinate system for each node m_i in the remaining node set $M_{mid} = \{m_i, i \in [2, k-1]\}$.

Given that the angle of node m_1 is θ_{m1}, and the angle of node m_k is θ_{mk}.

Let the neighbor nodes of node $m_i \in M_{mid}$ be $j \in N_{mi}$, and define a subset of m_i's neighbors as $N_{mi}^{part1} = \{j \in N_{mi} \text{ and } j \in M\}$.

For the neighbor nodes $j \in N_{mi}^{part1}$, let the rotation matrix from node i to node j be $R_{mi,j}$, which is converted into an angle $\theta_{mi,j}$.

For each neighbor node $j \in N_{mi}^{part1}$, construct the linear equation $\theta_{mi} + \theta_{mi,j} = \theta_j, j \in N_{mi}^{part1}$.

For each node in the set M_{mid}, construct the corresponding linear equations to form a linear system $A\theta = b$, where$\theta = [\theta_2, \ldots, \theta_{k-1}]^T \in R^{k-2}$.

The least squares solution of the linear system is given by: $\theta = (A^TA)^{-1}A^Tb$.

REFERENCES

[1] T. Lee, B. Jang and D. Cho, "A Non-Iterative Pose-Graph Optimization Algorithm for Fast 2D SLAM," Proceedings of the 2014 IEEE International conference on robotics and biomimetics December 5-10, 2014, Bali, Indonesia.

[2] Z. Lin, M. Fu, and Y. Diao, "Distributed Self Localization for Relative Position Sensing Networks in 2D Space," IEEE Transactions on signal processing, vol. 63, no. 14, july 15, 2015.

[3] Z. Wu, "Outlier Removal Algorithm in Pose Transformation" 10.5281/zenodo.14467742