A linear SLAM backend optimization algorithm based on virtual coordinate system in 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. Representing triangles with linear equations

We use linear equations to represent the shape of any triangle in a plane. Let the coordinates of the three vertices of a triangle ijk in the plane be $p_i, p_j, p_k \in R^2$. Given the length of edge (i, j) as ρ_{ij}, the length of edge (i, k) as ρ_{ik}, and the angle $\angle jik$ as θ_{jik}, we can derive the congruence constraints of the triangle based on these three conditions. To represent the positional constraints between nodes using linear equations, we relax the congruence constraints to similarity constraints, reformulating them as $\rho_{jik} = \rho_{ik}/\rho_{ij}, \theta_{jik}$. This relaxation ensures that triangle ijk retains its shape but allows for uniform scaling of the entire triangle. Under this condition, the similarity constraints of the triangle can be expressed using the following system of linear equations.

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

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

The variables p_i, p_j and p_k represent the coordinates of nodes i, j, and k, respectively, and are variables that require constraints. The term $R_{jik}(p_j - p_i)$ denotes the rotation of vector ij by an angle θ_{jik}, resulting in a vector aligned in the same direction as vector ik. Subsequently, there exists a proportional relationship $\rho_{jik} = \rho_{ik}/\rho_{ij}$ between $R_{jik}(p_j - p_i)$ and the 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.

In [1], an equivalent form of the above equation is presented using a linear equation in the complex domain. Let the complex coordinate of node i in the set V be defined as $p_i^c = x_i + iy_i$, where x_i is the x-component of the coordinate of node i, and y_i is the y-component. Consequently, the above system of linear equations can be expressed as the following complex linear equation.

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

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

We define p_{ki}^c as the complex coordinate form of vector ik, and θ_{ki} as the angle of vector ik. Similarly, let p_{ji}^c denote the complex coordinate form of vector ij, and θ_{ji} denote the angle of vector ij.

$$p_{ki}^c = p_k^c - p_i^c = \rho_{ki} e\text{^}(i\theta_{ki})$$

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

By substituting p^c_{ki} and p^c_{ji} into the above equations in place of p^c_k, p^c_i and p^c_j, we obtain the following equations.

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

$$\rho_{ki} e^\wedge(i\theta_{ki}) = \rho_{jik} \rho_{ji} e^{i(\theta_{ji}+\theta_{jik})}$$

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

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

$$\theta_{ji} + \theta_{jik} = \theta_{ki}$$

The constraints described here are precisely the same as the similarity triangle constraints for nodes i, j, and k. Therefore, the aforementioned complex linear equation can equivalently represent the shape constraint of triangle jik. Whether using the linear equation or the complex linear equation, both can equivalently express the constraints of triangle jik. In the subsequent text, we will use the complex linear equation to describe the pose constraints among the three nodes, as the complex linear equation is relatively more concise, comprising only a single equation. For any node i, its coordinates are represented in the form of a complex number as $p^c_i = x_i + iy_i$.

If any two nodes in triangle ijk coincide, the equation degenerates. Assume nodes i and k coincide, then $\rho_{ik} = 0$ and $\rho_{jik} = \rho_{ik}/\rho_{ij} = 0$. The parameter $\omega_{jik} = \rho_{jik} e^{i\theta_{jik}} = 0$. The resulting complex linear equation is as follows.

$$p^c_k - p^c_i = 0 \times (p^c_j - p^c_i)$$

When node i and node k coincide, the equation degenerates into $p^c_i = p^c_k$. Similarly, if node i and node j coincide, the equation degenerates into $p^c_i = p^c_j$. If node j and node k coincide, the equation becomes $p^c_j = p^c_k$. Finally, if nodes i, j, and k coincide, the equation reduces to $p^c_i = p^c_j = p^c_k$. Therefore, we introduce the following judgment steps.

$$p^c_i = p^c_j, \text{if } \rho_{ij} < \epsilon_{thres}$$

$$p^c_i = p^c_k, \text{if } \rho_{ik} < \epsilon_{thres}$$

$$p^c_j = p^c_k, \text{if } \rho_{jk} < \epsilon_{thres}$$

$$p^c_i = p^c_j = p^c_k, \text{if any two of } \rho_{ij}, \rho_{ik}, \rho_{jk} \text{ are less than } \epsilon_{thres}$$

Here, ϵ_{thres} is the threshold used to determine whether the nodes coincide.

We then present the conditions for a specific localizability problem. In triangle ijk, if the coordinates of nodes i and k are known, and the coordinates of node j need to be calculated using a linear equation, the conditions for the localizability of node j are as follows. If nodes i and k are not coincident, then node j is localizable. If nodes i and k are coincident, the linear equation degenerates into $p^c_i = p^c_j$. This equation is unrelated to the coordinates of node j, and thus the coordinates of node j cannot be determined using the linear equation. From a geometric perspective, we can arrive at the same conclusion: if nodes i and k are coincident, knowing ρ_{ij} and ρ_{kj} is insufficient to determine the coordinates of node j. In this case, node j lies on a circle.

3. Representing pose transformation with linear equations

3.1 Constructing linear equations

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

Table 1. Camera pose estimation algorithms

Feature point type	Pose transformation algorithm
2d-2d	epipolar geometry
2d-2d(planar)	homography matrix
2d-3d	DLT, EPnP
3d-3d	Icp

Similarly, if two frames of point clouds are obtained using a LiDAR sensor, various point cloud registration algorithms can be employed to determine the pose transformation. Additionally, pose transformations between the two frames can also be derived using methods such as IMU sensors, GPS sensors, wheel odometers, and loop closure detection.

Regardless of the type of sensor used, the initial information we obtain consists of the relative pose transformations between two keyframes (nodes). The purpose of back-end optimization is to utilize these pose transformation data to estimate an accurate global map. Here, we use the node $i \in V$ to represent a keyframe, where each node i is associated with its own local coordinate system Σ_i. There exists an unknown rotation matrix R_i that relates the local coordinate system Σ_i of node i to the global coordinate system Σ_g. Let the coordinates of node i in a 2D space be $p_i = [x_i, y_i]^T$, and let the complex coordinate of node i in the 2D plane be $p^c_i = x_i + iy_i$. In the complex coordinate of a node, the real part represents the x-axis component of the node's coordinates, while the imaginary part represents the y-axis component. In the subsequent sections of this paper, we will use the vector representation of a node's coordinates (e.g., p_i) along with the subscript c to denote the corresponding complex coordinate (e.g., p^c_i), with the x-axis and y-axis components corresponding sequentially. Let the set of all pose transformations T_{ij} be $E = \{T_{ij}, i, j \in V\}$. We can then define a directed graph $G = [V, E]$, where V represents the set of all nodes (keyframes) and E represents the set of all pose transformations.

We convert the pose transformation $T_{ij} \in R^{3\times3}$ between any two nodes (node i and node j) into a linear equation constraint. Assume the coordinates of node i are $p_i \in R^2$, and its complex coordinate is $p^c_i \in C$. We define a virtual node i_x located on the x-axis of the local coordinate frame Σ_i, with coordinates p_{ix}, which satisfies $||p_{ix} - p_i|| = 1$. The complex coordinate of i_x is denoted as $p^c_{ix} \in C$. Similarly, we define a virtual node i_y located on the y-axis of the local coordinate frame Σ_i, with coordinates p_{iy}, which satisfies $||p_{iy} - p_i|| = 1$. The complex coordinate of i_y is denoted as $p^c_{iy} \in C$. Hence, in the local coordinate frame Σ_i, node i serves as the origin of the local coordinate system, and i_x and i_y represent the nodes on the axes of the unit vectors in this local frame.

The coordinates of the same node j are denoted as $p_j \in R^2$, with its complex coordinate represented as $p^c_j \in C$. We define a virtual node j_x to be located on the x-axis of the local coordinate system Σ_j, with coordinates p_{jx}, satisfying $||p_{jx} - p_j|| = 1$. The complex coordinate of the virtual node j_x is denoted as $p^c_{jx} \in C$. Similarly, we define a virtual node j_y to be located on the y-axis of the local coordinate system Σ_j, with coordinates $p^c_{jy} \in C$, satisfying $||p_{jy} - p_j|| = 1$. The complex coordinate of the virtual node j_y is denoted as $p^c_{jy} \in C$. Therefore, in the local coordinate system Σ_j, the node j serves as the origin of the local coordinate system, while p_{jx} and p_{jy} are the unit coordinate vectors along the two axes of the local coordinate system.

If the coordinates of all points in the local coordinate system Σ_i are known as $(p^c_i, p^c_{ix}, p^c_{iy})$, we need to construct linear equations to represent the coordinates of all points in another local coordinate system Σ_j, denoted as $(p^c_j, p^c_{jx}, p^c_{jy})$. Through this approach, starting from the first node, we can recursively determine the local coordinate systems for all nodes.

Given the coordinates of two axes in the local coordinate system Σ_i, p^c_{ix} and p^c_{iy}, we first construct the linear equation for the virtual node coordinate p^c_{jx} in the local coordinate system Σ_j. Using the method of representing triangles with linear equations, we can construct the linear equation for the triangle $j_x - i_x - i_y$.

$$p^c_{iy} - p^c_{jx} = \omega_{ix,jx,iy}(p^c_{ix} - p^c_{jx})$$

The variables $p^c_{ix}, p^c_{iy}, p^c_{jx} \in C$ represent the coordinates of nodes i_x, i_y and j_x, respectively. The parameter $\omega_{ix,jx,iy}$ will be specified in detail later.

Subsequently, we use the virtual node coordinate p^c_{jy} in Σ_j and the two points p^c_{ix}, p^c_{iy} in Σ_i to construct a linear equation. By employing the representation method of triangles, we construct the linear equation of the triangle $j_y - i_x - i_y$.

$$p^c_{iy} - p^c_{jy} = \omega_{ix,jy,iy}(p^c_{ix} - p^c_{jy})$$

The variables $p^c_{ix}, p^c_{iy}, p^c_{jy} \in C$ represent the coordinates of nodes i_x, i_y and j_y, respectively. The parameters $\omega_{ix,jy,iy}$ will be specified later in detail.

Then, we construct a linear equation using the coordinates of the node $p^c_i \in \Sigma_i$ and two other points p^c_{ix} and p^c_{iy} from Σ_i. Using the representation of triangles, we formulate the linear equation for the triangle $i - i_x - i_y$.

$$p^c_{ix} - p^c_i = \omega_{ix,i,iy}(p^c_{ix} - p^c_i)$$

Here, $p^c_{ix}, p^c_{iy}, p^c_i \in C$ are the coordinates of nodes i_x, i_y and i, respectively. The parameters $\omega_{ix,i,iy}$ will also be specified in detail later.

If we use two virtual nodes, i_x and i_y, in the local coordinate system Σ_i to represent a node j and two virtual nodes, j_x and j_y, in the local coordinate system Σ_j, then, since the virtual nodes i_x and i_y are unit vectors along the two axes of the local coordinate system Σ_i, the nodes i_x and i_y are not coincident (satisfying the condition for localizability of the triangle). If the coordinates of the two virtual nodes in the local coordinate system Σ_i, denoted as p^c_{ix} and p^c_{iy}, are known, we can calculate the coordinates of the node j (p^c_j) and the two virtual nodes (p^c_{jx} and p^c_{iy}) in the local coordinate system Σ_j. By following this approach iteratively, all nodes (keyframes) in the SLAM problem can be sequentially connected into a chain. If the pose of the first node (i.e., the coordinates of its two virtual nodes, p^c_{1x} and p^c_{1y}) is known, the pose of any node i (i.e., the coordinates of its two virtual nodes, p^c_{ix} and p^c_{iy}) can be calculated.

Given the coordinates of two virtual nodes, i_x and i_y, in the local coordinate system Σ_i, the coordinates of two virtual nodes, j_x and j_y, in the local coordinate system Σ_j can be computed using a set of linear equations. Now, we consider the inverse problem: if the coordinates of the two virtual nodes j_x and j_y in Σ_j are known, can the coordinates of i_x and i_y in Σ_i be computed using the same linear equations? To address this, we can discuss the geometric meaning of these linear equations. The linear equation corresponding to the triangle $i_x - i_y - j_x$ represents the shape of the triangle defined by nodes i_x, i_y and j_x (i.e., a similarity transformation), but the scale of the triangle can vary. Similarly, the linear equation for the triangle $i_x - i_y - j_y$ constrains the shape of the triangle defined by nodes i_x, i_y and j_y, again allowing for scaling. Thus, the geometric structure formed by the four nodes (i_x, i_y, i_z, j_x) is fixed in terms of shape, though its scale can vary. Consequently, given the coordinates of the virtual nodes j_x and j_y, it is indeed possible to compute the coordinates of i_x and i_y using the same set of linear equations. Therefore, the two linear equations corresponding to the two triangles ($i_x - i_y - j_x$ and $i_x - i_y - j_y$) can equivalently represent the pose transformation T_{ij}.

In the above system of linear equations, we use two virtual nodes, i_x and i_y, in the local coordinate system Σ_i to represent the two virtual nodes, j_x and j_y, in the local coordinate system Σ_j. To ensure symmetry, we could alternatively use the two virtual nodes, j_x and j_y, in the local coordinate system Σ_j to represent the two virtual nodes, i_x and i_y, in the local coordinate system Σ_i, and formulate the corresponding two linear equations. However, as previously proven, the two linear equations using i_x and i_y to represent j_x and j_y can already equivalently represent the pose transformation T_{ij}. Therefore, using j_x and j_y again to represent ix and iy is redundant.

In the graph $G = [V, E]$, V represents the set of all nodes (keyframes). Assume that there are n nodes in the set V. For each node $i \in V$, two virtual nodes i_x and i_y are defined in the local coordinate frame Σ_i of node i. Therefore, we define the virtual node set as $V_v = \{i_x, i_y, i \in V\}$. The coordinates of these virtual nodes are represented as $p_{ix}, p_{iy} \in R^2$, where $i_x, i_y \in V_v$. The coordinate vector of the virtual nodes is defined as $p_v = [p_{1x}, p_{1y}, \dots, p_{nx}, p_{ny}] \in R^{2n}$. Furthermore, the complex coordinates of the virtual nodes are denoted as $p^c_{ix}, p^c_{iy} \in C$, where $i_x, i_y \in V_v$. The complex coordinate vector of the virtual nodes is then defined as $p^c_v = [p^c_{1x}, p^c_{1y}, \dots, p^c_{nx}, p^c_{ny}] \in C^n$.

The set of all pose transformations T_{ij} is denoted as $E = \{T_{ij}, i, j \in V\}$. The pose transformation T_{ij} between any two nodes (node i and node j) can be converted into two linear equations. Thus, the system of linear equations corresponding to all pose transformations $T_{ij} \in E$ is expressed as follows.

$$Ap^c_v = b$$

In addition, we represent the coordinates of node i using the coordinates of two virtual nodes i_x and i_y in the local coordinate system Σ_i. Once the coordinates of all virtual nodes p^c_v have been computed, we can use p^c_v to calculate the coordinates of each node $i \in V$ in the graph $G = [V, E]$. The coordinate vector of all nodes $i \in V$ is defined as $p = [p_1, p_2, \dots, p_n] \in R^{2n}$, and the complex coordinate vector of all nodes $i \in V$ is defined as $p^c = [p^c_1, p^c_2, \dots, p^c_n] \in C^n$. The equation for calculating the complex coordinates p^c of all nodes $i \in V$ using the virtual node coordinates p^c_v is as follows.

$$Cp^c = Dp^c_v$$

The pseudocode of the algorithm is as follows.

Procedure 1: Representing pose transformation with linear equations

```
Input: A graph $G = [V, E]$, where V represents a set of nnn nodes, and E represents the set of all pose transformations
Output: Matrices $A, b, C, D$
The vector of all nodes: $p = [p_1, p_2, \dots, p_n] \in R^{2\times n}$.
The complex vector of all nodes: $p^c = [p^c_1, p^c_2, \dots, p^c_n]^T \in C^n$.
The vector of all virtual nodes: $p_v = [p_{1x}, p_{1y}, \dots, p_{nx}, p_{ny}] \in R^{2\times 2n}$.
The complex vector of all virtual nodes: $p^c_v = [p^c_{1x}, p^c_{1y}, \dots, p^c_{nx}, p^c_{ny}]^T \in C^{2n}$.
For ($T_{ij} \in V$):
    Construct the linear equation for triangle $i_x - i_y - j_x$: $p^c_{iy} - p^c_{jx} = \omega_{ix,jx,iy}(p^c_{ix} - p^c_{jx})$; Incorporate it into the linear system $Ap^c_v = b$.
    Construct the linear equation for triangle $i_x - i_y - j_y$: $p^c_{iy} - p^c_{jy} = \omega_{ix,jy,iy}(p^c_{ix} - p^c_{jy})$; Incorporate it into the linear system $Ap^c_v = b$.
For ($i \in V$):
    Construct the linear equation for triangle $i_x - i_y - i$: $p^c_{iy} - p^c_i = \omega_{ix,i,iy}(p^c_{ix} - p^c_i)$; Incorporate it into the linear system $Cp^c = Dp^c_v$.
```

3.2 Compute the linear equation for the triangle ix-iy-jx

In the previous sections, we constructed the following linear equation using the triangle $i_x - i_y - j_x$.

$$p^c_{iy} - p^c_{jx} = \omega_{ix,jx,iy}(p^c_{ix} - p^c_{jx})$$

Here, $p^c_{ix}, p^c_{iy}, p^c_{jx} \in C$ represent the coordinates of the nodes i_x, i_y, j_x, respectively. The parameters $\omega_{ix,jx,iy}$ were not explicitly described in the previous sections. In this section, we will provide the calculation methods for these parameters. In the local coordinate system Σ_i, the coordinates of the virtual nodes i_x, i_y, i_z are as follows.

$$p^i_{ix} = [1,0]^T$$

$$p^i_{iy} = [0,1]^T$$

Let the pose transformation from node i to node j be denoted as $T_{ij} \in R^{3\times 3}$.

$$T_{ij} = \begin{bmatrix} R_{ij} & t_{ij} \\ 1 & 0 \end{bmatrix} \in R^{3\times 3}, R_{ij} \in R^{2\times 2}$$

$$t_{ij} = [x_{ij}, y_{ij}]^T \in R^2$$

Then, the position of the virtual node j_x in the local coordinate system Σ_i is as follows.

$$p^i_{jx} = R_{ij}[1,0]^T + t_{ij}$$

Below, we provide the relative positions of the vectors (j_x, i_x) and (j_x, i_y) in the local coordinate system Σ_i.

$$p^i_{jx,ix} = p^i_{ix} - p^i_{jx} = [x^i_{jx,ix}, y^i_{jx,ix}]^T$$

$$p^i_{jx,iy} = p^i_{iy} - p^i_{jx} = [x^i_{jx,iy}, y^i_{jx,iy}]^T$$

The subscripts and superscripts of the coordinates in this section are relatively complex. We will redefine the meanings of these indices for clarity. In the coordinate $p^i_{jx,ix}$, the subscripts j_x, i_x indicate that this is the coordinate of the vector (j_x, i_x). The superscript i specifies that the coordinate is expressed in the local coordinate system Σ_i.

Next, we calculate the parameter $\omega_{ix,jx,iy}$. The length between the coordinates $p^i_{jx,ix}$ and $p^i_{jx,iy}$ is given by:

$$\rho_{jx,ix} = ||p^i_{jx,ix}||$$

$$\rho_{jx,iy} = ||p^i_{jx,iy}||$$

The ratio of the magnitudes is as follows.

$$\rho_{ix,jx,iy} = \frac{\rho_{jx,iy}}{\rho_{jx,ix}}$$

The angle is calculated as follows.

$$\theta_{ix,jx,iy} = \arccos\left(\frac{p^i_{jx,ix} p^i_{jx,iy}}{\left\|p^i_{jx,ix}\right\| \left\|p^i_{jx,iy}\right\|}\right)$$

The parameter $\omega_{ix,jx,iy}$ is defined as follows.

$$\omega_{ix,jx,iy} = \rho_{ix,jx,iy} e^{i\theta_{ix,jx,iy}}$$

When node i_x coincides with node j_x, the equation degenerates to $p^c_{ix} = p^c_{jx}$. Similarly, if node i_y coincides with node j_x, the equation degenerates to $p^c_{iy} = p^c_{jx}$. Nodes i_x and i_y represent the unit vectors of two axes in the local coordinate system Σ_i and therefore cannot coincide. We introduce the following judgment steps.

$$p^c_{ix} = p^c_{jx}, \text{ if } \rho_{jx,ix} < \epsilon_{thres}$$

$$p^c_{iy} = p^c_{jx}, \text{ if } \rho_{jx,iy} < \epsilon_{thres}$$

Here, ϵ_{thres} is the threshold used to determine whether nodes coincide. Using the pose transformation T_{ij} from node i to node j, the linear equation of the triangle $i_x - i_y - j_x$ can be computed. The pseudocode is described as follows.

Process 2. Compute the linear equation for the triangle ix-iy-jx

```
Input: Pose transformation $T_{ij}$; virtual node $j_x$'s coordinate in the local coordinate system $\Sigma_j$: $p^j_{jx} = [1,0]^T$; distance threshold $\epsilon_{thres}$
Output: The linear equation involving nodes $i_x, i_y$ and $j_x$
Position of virtual nodes $i_x$ and $i_y$ in the local coordinate system $\Sigma_i$: $p^i_{ix} = [1,0]^T, p^i_{iy} = [0,1]^T$.
Position of virtual node $j_x$ in the local coordinate system $\Sigma_i$: $p^i_{jx} = R_{ij} p^j_{jx} + t_{ij}$.
Relative positions of vectors $(j_x, i_x)$ and $(j_x, i_y)$ in local coordinates $\Sigma_i$:
$p^i_{jx,ix} = p^i_{ix} - p^i_{jx} = [x^i_{jx,ix}, y^i_{jx,ix}]^T$
$p^i_{jx,iy} = p^i_{iy} - p^i_{jx} = [x^i_{jx,iy}, y^i_{jx,iy}]^T$
Magnitudes of vectors $p^i_{jx,ix}$ and $p^i_{jx,iy}$:
$\rho_{jx,ix} = \left\|p^i_{jx,ix}\right\|, \rho_{jx,iy} = \left\|p^i_{jx,iy}\right\|$
If($\rho_{jx,ix} < \epsilon_{thres}$):
    The equation degenerates to: $p^c_{ix} = p^c_{jx}$.
Elseif($\rho_{jx,iy} < \epsilon_{thres}$):
    The equation degenerates to: $p^c_{iy} = p^c_{jx}$.
Else:
    Magnitude ratio: $\rho_{ix,jx,iy} = \rho_{jx,iy}/\rho_{jx,ix}$.
    Angle calculation: $\theta_{ix,jx,iy} = \arccos\left(\frac{p^i_{jx,ix} p^i_{jx,iy}}{\left\|p^i_{jx,ix}\right\| \left\|p^i_{jx,iy}\right\|}\right)$.
    Parameter calculation: $\omega_{ix,jx,iy} = \rho_{ix,jx,iy} e^{i\theta_{ix,jx,iy}}$.
    Linear equation: $p^c_{iy} - p^c_{jx} = \omega_{ix,jx,iy}(p^c_{ix} - p^c_{jx})$.
```

3.3 Compute the linear equations for the triangle ix-iy-jy, ix-iy-i

In the previous sections, we constructed the following linear equations based on the triangle $i_x - i_y - j_y$.

$$p^c_{iy} - p^c_{jy} = \omega_{ix,jy,iy}(p^c_{ix} - p^c_{jy})$$

Let $p^c_{ix}, p^c_{iy}, p^c_{jy} \in C$ represent the coordinates of the nodes i_x, i_y and j_y, respectively. The parameters $\omega_{ix,jy,iy}$ were not explicitly described in terms of their calculation methods in the previous sections. Below, we provide the pseudocode for their computation.

Process 3. Compute the linear equations for the triangle ix-iy-jy

```
Input: Pose transformation $T_{ij}$; coordinates of the virtual node $j_y$ in the local coordinate system $\Sigma_j$, given by $p^j_{jy} = [0,1]^T$; distance threshold $\epsilon_{thres}$
Output: The linear equations describing the relationship among the nodes $i_x, i_y$ and $j_y$
Position of virtual nodes $i_x$ and $i_y$ in local coordinate system $\Sigma_i$: $p^i_{ix} = [1,0]^T, p^i_{iy} = [0,1]^T$.
Position of virtual node $j_y$ in local coordinate system $\Sigma_i$: $p^i_{jy} = R_{ij} p^j_{jy} + t_{ij}$.
Relative positions of vectors $(j_y, i_x)$ and $(j_y, i_y)$ in $\Sigma_i$:
$p^i_{jy,ix} = p^i_{ix} - p^i_{jy} = [x^i_{jy,ix}, y^i_{jy,ix}]^T$
$p^i_{jy,iy} = p^i_{iy} - p^i_{jy} = [x^i_{jy,iy}, y^i_{jy,iy}]^T$
The magnitudes of the vector coordinates $p^i_{jy,ix}$ and $p^i_{jy,iy}$ are defined as:
$\rho_{jy,ix} = \left\|p^i_{jy,ix}\right\|, \rho_{jy,iy} = \left\|p^i_{jy,iy}\right\|$
If($\rho_{jy,ix} < \epsilon_{thres}$):
    The linear equation degenerates into: $p^c_{ix} = p^c_{jy}$.
Elseif($\rho_{jy,iy} < \epsilon_{thres}$):
    The linear equation degenerates into: $p^c_{iy} = p^c_{jy}$.
Else:
    The ratio of the magnitudes is given by: $\rho_{ix,jy,iy} = \rho_{jy,iy}/\rho_{jy,ix}$.
    The angle is calculated as:
```

```
$\theta_{ix,jy,iy} = \arccos\left(\frac{p^i_{jy,ix} p^i_{jy,iy}}{\left\|p^i_{jy,ix}\right\| \left\|p^i_{jy,iy}\right\|}\right)$
    The parameter is computed as: $\omega_{ix,jy,iy} = \rho_{ix,jy,iy} e^{i\theta_{ix,jy,iy}}$.
    The linear equation becomes: $p^c_{iy} - p^c_{jy} = \omega_{ix,jx,iy}(p^c_{ix} - p^c_{jy})$.
```

In the previous sections, we constructed the following linear equation using the triangle $i_x - i_y - i$.

$$p^c_{iy} - p^c_i = \omega_{ix,i,iy}(p^c_{ix} - p^c_i)$$

Here, $p^c_{ix}, p^c_{iy}, p^c_i \in C$ represent the coordinates of nodes i_x, i_y and i, respectively. The parameters $\omega_{ix,i,iy}$ were not explicitly described in terms of their computation in the previous sections. Below, we provide the pseudocode for their calculation.

Process 4. Compute the linear equations for the triangle ix-iy-i

```
Input: Pose transformation $T_{ij}$; the coordinates of node i in the local coordinate system $\Sigma_i$, denoted as $p^i_i = [0,0]^T$
Output: The linear equation for $i_x, i_y$ and $j_y$ of the triangle
The positions of the virtual nodes $i_x$ and $i_y$ in the local coordinate system $\Sigma_i$ are: $p^i_{ix} = [1,0]^T, p^i_{iy} = [0,1]^T$.
The relative positions of vectors $(i, i_x)$ and $(i, i_y)$ in the local coordinate system $\Sigma_i$ are:
$p^i_{i,ix} = p^i_{ix} - p^i_i = \left[x^i_{i,ix}, y^i_{i,ix}\right]^T$
$p^i_{i,iy} = p^i_{iy} - p^i_i = \left[x^i_{i,iy}, y^i_{i,iy}\right]^T$
The magnitudes of the vectors $p^i_{i,ix}$ and $p^i_{i,iy}$ are:
$\rho_{i,ix} = \left\|p^i_{i,ix}\right\| = 1, \rho_{i,iy} = \left\|p^i_{i,iy}\right\| = 1$
The ratio of the magnitudes is: $\rho_{ix,i,iy} = \rho_{i,iy}/\rho_{i,ix}$.
The angle calculation: $\theta_{ix,i,iy} = \arccos\left(\frac{p^i_{i,ix} p^i_{i,iy}}{\left\|p^i_{i,ix}\right\| \left\|p^i_{i,iy}\right\|}\right) = \frac{\pi}{2}$.
The parameter calculation: $\omega_{ix,i,iy} = \rho_{ix,i,iy} e^{i\theta_{ix,i,iy}} = e^{\frac{i\pi}{2}}$.
The linear equation: $p^c_{iy} - p^c_i = \omega_{ix,i,iy}(p^c_{ix} - p^c_i)$.
```

4. Pose transformation of different sensors

In this section, we discuss how data obtained from different types of sensors can be transformed into relative pose transformations between 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 individually address the pose transformations obtained from each type of sensor.

4.1 Non-GPS sensors

We first discuss the case of camera and LiDAR sensors. These two types of sensors can obtain a deterministic relative pose transformation, T_{ij}, between two nodes (e.g., node i and node j). Typically, nodes i and j correspond to two adjacent keyframes or two keyframes that are spatially very close.

Next, for IMU sensors and wheel odometers, these sensors generally provide the relative pose transformation, $T_{i-1,i}$, between two consecutive keyframes.

The loop closure detection process also yields the relative pose transformation T_{ij} between two nodes (node i and node j). Typically, nodes i and j are adjacent nodes identified when the robot returns to its starting point after completing a loop. Although nodes i and j are temporally distant in terms of the robot's movement, they are spatially close in terms of distance.

The relative pose transformation T_{ij} between two nodes (node i and node j) obtained through non-GPS sensors can be transformed into a set of linear equations between two pairs of virtual nodes (virtual nodes i_x, i_y and virtual nodes j_x, j_y).

4.2 GPS sensors

The processing method for GPS sensor data has already been provided in [2]. However, for the sake of completeness in the SLAM framework, we will still elaborate on the specific approach to handling GPS data.

The case of GPS sensors is somewhat unique, as these sensors provide data in the form of latitude and longitude. Using appropriate formulas, latitude and longitude can be converted into specific distance values. Therefore, the position detected by a GPS sensor is always 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 LiDARs) can only provide the relative pose transformation, T_{ij}, between two specific nodes (node i and node j). If the pose transformation between two distant nodes (node i and node $i + n$) is required, it can only be obtained by cumulative multiplication: $T_{i,i+n} = T_{i,i+1}T_{i+1,i+2} \ldots T_{i+n-1,i+n}$. In this case, the error of the cumulative pose transformation $T_{i,i+n}$ will gradually accumulate. However, for GPS sensors, since they rely on a fixed latitude-longitude coordinate system, the translational component $t_{i,i+n}$ between two distant nodes (node i and node $i + n$) can still maintain very high accuracy. As a result, GPS sensor data plays a significant role in ensuring the accuracy of the overall map shape in large-scale mapping tasks.

Another unique characteristic of GPS sensors is that GPS data provides only displacement information without any rotational information. Suppose we obtain GPS data for node i (its position in the geographic coordinate system, represented by latitude and longitude) and subsequently obtain GPS data for node j. In this case, we can only derive the translational vector $t_{ij} \in R^2$ from node i to node j in the geographic coordinate system, without any rotational information. Consequently, the translational 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 construct nonlinear constraints in a pose graph optimization framework, this approach differs from our intended goal of formulating a linear equation-based optimization algorithm.

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

Let the GPS latitude and longitude data of the i-th node be denoted as $pos_{g(i)} = [lon_{g(i)}, lat_{g(i)}]^T$, and that of the j-th node as $pos_{g(j)} = [lon_{g(j)}, lat_{g(j)}]^T$. The relative translation vector between node i and node j can be derived from their latitude and longitude data using either the spherical distance formula or the UTM planar coordinate system.

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

The function f is used to transform the latitude and longitude data of two nodes, $pos_{g(i)}$ and $pos_{g(j)}$, into a translation vector $t_{g(i),g(j)}$.

After obtaining data from the GPS sensor, we construct two types of linear equations: (1) linear equations for adjacent GPS data and (2) linear equations for GPS data across nodes. 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 represented as $t_{g(i+1),g(i)} = f(pos_{g(i+1)}, pos_{g(i)})$, while the relative position from node g_{i+1} to node g_{i+2} is t_g(i+1),g(i+2) = f(pos_g(i+1), pos_g(i+2)). Similarly, the relative position from node g(i) to node g(i+2) is $t_{g(i+1),g(i+2)} = f(pos_{g(i+1)}, pos_{g(i+2)})$. First, it is necessary to ensure that these three nodes do not overlap. The method for determining whether the nodes overlap is as follows.

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

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

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

After confirming 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 then be computed as follows.

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

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

In addition, we can construct linear equations based on GPS data across nodes. The translational vector $t_{g(i),g(i+1)}$ calculated from any two adjacent GPS data points cannot fully utilize the information provided by GPS. This is because the strength of GPS data lies in its ability to maintain high accuracy in the relative position measurements between two nodes that are far apart. Therefore, in addition to constructing the aforementioned linear equations, we also need to use GPS data to construct linear equations for nodes that are far apart.

To achieve this, we need to identify three such nodes (nodes g_i, g_j, and g_k) from the GPS data set G. These three nodes must be distinct and non-overlapping.

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

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

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

The distances between these three nodes are approximately equal, or we can describe this 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 ratio between two sides. We need to ensure that the ratio of any two sides in the triangle $g_i g_j g_k$ is less than p_{thres} (the ratio of the longer side to the shorter side).

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. Utilizing the similar triangle formed by these three points to construct a linear equation can effectively constrain the overall shape of the map. To ensure that the similar triangle formed by these three points is preserved as much as possible, the positions of nodes g_i, g_j and g_k must undergo uniform and relatively large adjustments.

To address the problem of finding as many such triangular combinations as possible in the GPS data set G, we adopt the following approach: three points are randomly selected from the GPS data set, and it is determined whether they satisfy the aforementioned edge ratio conditions. If the conditions are not met, a new selection is made. We can define a maximum search count f_{max} and the required number of triangles Δ_{req}. Once three nodes g_i, g_j and g_k that meet the requirements are obtained, the following linear equations can be constructed.

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

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

We construct a system of linear equations based on the aforementioned linear equations, which is denoted as follows.

$$Ep^c = 0$$

Let $p^c \in C^n$ represent the planar coordinate vector of all nodes i ∈ V. For the GPS data set G, we construct two types of linear equations: the linear equations for adjacent GPS data and the linear equations for cross-node GPS data. Below, we first introduce the linear equations for adjacent GPS data. The specific methods are presented in the following pseudocode.

Process 5. Constructing linear equations from adjacent GPS data

```
Input: GPS data set $G = \{pos_{g(i)} = [lon_{g(i)}, lat_{g(i)}]^T, g_i \in [1, n], i \in [1, m]\}$; overlapping node distance threshold $\epsilon_{thres}$
Output: Plane coordinate vector $p^c = [p^c_1, p^c_2, \dots, p^c_n]^T$; matrix E
For (i in $[1, m-2]$):
    Compute the relative translation between adjacent nodes $g_i, g_{i+1}$: $t_{g(i),g(i+1)} = f(pos_{g(i)}, pos_{g(i+1)})$; distances, $\rho_{g(i),g(i+1)}$.
    Compute the relative translation between adjacent nodes $g_{i+1}, g_{i+2}$: $t_{g(i+1),g(i+2)} = f(pos_{g(i+1)}, pos_{g(i+2)})$; distances, $\rho_{g(i+1),g(i+2)}$.
    Compute the relative translation between adjacent nodes $g_i, g_{i+2}$: $t_{g(i),g(i+2)} = f(pos_{g(i)}, pos_{g(i+2)})$; distances $\rho_{g(i),g(i+2)}$.
    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 E.
```

Process 6. Constructing linear equations from 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 proportion threshold: $k_{thres}$; maximum search iterations: $f_{max}$; required number of triangles $\Delta_{req}$
Output: Plane coordinate vector $p^c = [p^c_1, p^c_2, \dots, p^c_n]^T$; matrix E
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 $g_i, g_j$ and $g_k$ from the 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$\left(\frac{\rho_{g(i),g(j)}}{\rho_{g(i),g(k)}} > p_{thres} \text{ and } \rho_{g(i),g(j)} \geq \rho_{g(i),g(k)}\right)$:
        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$\left(\frac{\rho_{g(k),g(j)}}{\rho_{g(k),g(i)}} < p_{thres}$ and $\rho_{g(k),g(i)} < \rho_{g(k),g(j)}\right)$:
        break
    Construct a linear equation using nodes $g_i, g_j$ and $g_k$:
    $p^c_{g(k)} - p^c_{g(i)} = \omega_{g(j),g(i),g(k)}(p^c_{g(j)} - p^c_{g(i)})$, where the
    equation is added to the matrix E.
    $\Delta_{num} = \Delta_{num} + 1$.
```

4.3 Variance of pose transformation

The relative pose transformation between two frames can be obtained using a combination of sensors, such as cameras, LiDAR, IMU sensors, GPS sensors, wheel odometers, and loop closure detection. Typically, the pose transformations obtained from different sensors have varying levels of uncertainty, which can be characterized by different variances. These variances can either be predefined based on the inherent characteristics of each sensor or calculated during the process of determining the pose transformation T_{ij} in the front-end module of the SLAM algorithm. The pose 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.

The linear equations between each node $i \in V$ and virtual nodes $i_x, i_y, i_z \in V_v$ are given as follows.

$$Ap^c_v = b + v_1$$

$$Cp^c = Dp^c_v + v_2$$

$$Ep^c = 0 + v_3$$

The error vectors are defined as $v_1 \sim N(0, \Sigma_1), v_2 \sim N(0, \Sigma_2)$ and $v_3 \sim N(0, \Sigma_3)$. By neglecting the correlations between different linear equations, the covariance matrix Σ is assumed to be a diagonal matrix.

For the variance Σ_1 of the error vector v_1, we use the variance σ^2_{ij} of the pose transformation T_{ij} corresponding to the linear equation as the variance of the equation. For the variance Σ_2 of the error vector v_2, it can be set to 0, because this system of linear equations uses the virtual coordinates i_x, i_y of node $i \in V$ to represent the coordinates of node i, without introducing observational data. For the error vector v_3, each linear equation is constructed using two pose transformations (e.g., T_{ij} and T_{ik}). For simplicity, we set the variance of the corresponding linear equation to $\sigma^2 = \sigma^2_{ij} + \sigma^2_{ik}$.

5. Map scale calculation

5.1 Without using GPS sensors

In the graph $G = [V, E]$, V represents the set of all nodes (keyframes). Assume there are n nodes in the set V. For each node $i \in V$, two virtual nodes, i_x and i_y, are defined in its local coordinate system Σ_i. The set of virtual nodes is denoted as $V_v = \{i_x, i_y, i \in V\}$, and the coordinate vector of the virtual nodes is $p_v = [p_{1x}, p_{1y}, \ldots, p_{nx}, p_{ny}] \in R^{2\times n}$. The complex vector of the virtual nodes is expressed as $p^c_v = [p^c_{1x}, p^c_{1y}, \ldots, p^c_{nx}, p^c_{ny}] \in C^n$.

Given the set of all pose transformations T_{ij}, denoted as $E = \{T_{ij}, i, j \in V\}$, the pose transformation T_{ij} between any two nodes (i, j) can be converted into two linear equations. Therefore, let the system of linear equations corresponding to all pose transformations $T_{ij} \in E$ be expressed as follows.

$$Ap^c_v = b$$

Let us denote the local coordinate system of the first node, Σ_1, as the global coordinate system Σ_g. Using this setup, we can compute the coordinates of other nodes. If we instead use another coordinate system, Σ_i, as the global coordinate system Σ_g, the corresponding pose transformation T_{1i} can be applied to transform all coordinates into the coordinate system Σ_i. Let the complex coordinate vector be $p^c_v \in C^{2n}$, where the complex coordinates of the virtual node 1_x are $p^c_{1x} = 1 + i \times 0$, the complex coordinates of the virtual node 1_y are $p^z_{1y} = 0 + i \times 1$. According to the node localizability conditions, as long as the coordinates of one node (e.g., node 1 and its virtual nodes 1_x and 1_y) are determined, the coordinates of all other nodes can be calculated. Substituting p^c_{1x}, p^c_{1y} into the coordinate vector p^c_v, we can then solve the linear equation $Ap^c_v = b$ to determine the coordinates of the other nodes contained in the vector p^c_v.

The geometric meaning of the linear equation $Ap^c_v = b$ in the planar direction is that the shape of the graph formed by all nodes $i \in V$ is fixed, but its scale can be adjusted. This is because each linear equation is equivalent to a constraint of a similar triangle. If the coordinates of other nodes in p^c_v are calculated using the two planar coordinates p^c_{1x} and p^c_{1y}, it is essentially equivalent to using these two coordinates to determine the scale of the entire planar graph. However, as this scale information propagates outward from node 1 to other nodes, the error gradually accumulates. For example, if we set the unit vector magnitudes of the two axes in the local coordinate system Σ_1 to be 1 ($\rho_{i,ix} = 1, \rho_{i,iy} = 1$), the vector magnitudes of the two axes in the local coordinate system Σ_n (formed by node n and its virtual nodes n_x and n_y) after calculating the coordinates of node n may not necessarily remain 1 ($\rho_{n,nx}, \rho_{n,ny}$). To address this issue and determine a more accurate map scale, we can adopt the following method.

Let the complex coordinate of node 1 be $p^c_1 = 0 + i \times 0$, the complex coordinate of the virtual node 1_x be $p^c_{1x} = \rho + i \times 0$, and the complex coordinate of the virtual node 1_y be $p^c_{1y} = 0 + i \times \rho$. Let the complex coordinate vector of the virtual nodes $i_x, i_y \in V_v/(1_x, 1_y)$ be $p^c_{sub} = [p^c_{2x}, p^c_{2y}, \ldots, p^c_{nx}, p^c_{ny}] \in C^{2n-2}$. Substituting the coordinates p^c_{1x}, p^c_{1y} into the equations, the linear system $Ap^c_v = b + v_1$ becomes the following.

$$A_{sub}p^c_{sub} = b_{sub} + v_{sub1}$$

The term $v_{sub1} \sim N(0, \Sigma_{sub1})$ represents the noise associated with each linear equation. The variance of each linear equation is expressed using the variance σ^2_{ij} corresponding to the pose transformation T_{ij}. The number of linear equations in the system remains unchanged, and the variance associated with each linear equation does not undergo any transformation. Therefore, the error vector $v_{sub1} = v_1$, and the covariance matrix $\Sigma_{sub1} = \Sigma_1$. The least squares solution for this linear equation system is given as follows.

$$p_{sub}^c = \left(A_{sub}^T \Sigma_1^{-1} A_{sub}\right)^{-1} A_{sub}^T \Sigma_1^{-1} b_{sub}$$

The complex coordinate vector p_{sub}^c represents the complex coordinates of the $2(n-1)$ virtual nodes from node 2 to node n. Each complex coordinate (e.g., p_{ix}^c, p_{iy}^c) has a real part and an imaginary part, both of which are linear functions of the unknown variable ρ.

$$p_{ix}^c = (a_{ix}\rho + b_{ix}) + i(c_{ix}\rho + d_{ix})$$
$$p_{iy}^c = (a_{iy}\rho + b_{iy}) + i(c_{iy}\rho + d_{iy})$$

Simultaneously, the complex coordinates of the two virtual nodes 1_x and 1_y associated with node 1 also have their real and imaginary parts as linear functions of the unknown variable ρ. Therefore, for all virtual nodes $i_x, i_y \in V_v$, the real and imaginary parts of their complex coordinates are all linear functions of the unknown variable ρ.

After calculating the coordinates of all virtual nodes p_v, we can use p_v to compute the coordinates of each node i∈Vi \in Vi∈V in the graph $G = [V, E]$. The coordinate vector of all nodes $i \in V$ is denoted as $p = [p_1, p_2, \dots, p_n] \in R^{2\times n}$, and the complex coordinate vector of all nodes $i \in V$ is denoted as $p^c = [p_1^c, p_2^c, \dots, p_n^c] \in C^n$. The equation for computing the complex coordinates p^c of all nodes $i \in V$ using the complex coordinates of the virtual nodes p_v^c is as follows.

$$Cp^c = Dp_v^c$$

The least squares solution to the above equation is given by:

$$p^c = (C^T C)^{-1} C^T D p_v^c$$

In the coordinate vector $p^c \in C^n$, each node $i \in V$ has a complex coordinate, where the real and imaginary parts of each complex coordinate are linear functions of an unknown variable ρ.

$$p_i^c = (a_i\rho + b_i) + i(c_i\rho + d_i)$$

Next, we define the following error function J_1. This error function represents the condition that, in the coordinate system Σ_i of each node i, the distance from node i to the virtual node i_x should be 1, and the distance from node i to node i_y should also be 1.

$$J_1 = \sum_{i\in V} \left(\left(\left\|p_i - p_{ix}\right\|^2 - 1 \right)^2 + \left(\left\|p_i - p_{iy}\right\|^2 - 1 \right)^2 \right)$$
$$= \sum_{i\in V} \Big(((a_i\rho - a_{ix}\rho)^2 + (b_i - b_{ix})^2 - 1)^2$$
$$+ \left(\left(a_i\rho - a_{iy}\rho\right)^2 + \left(b_i - b_{iy}\right)^2 - 1 \right)^2$$
$$+ ((a_i\rho - a_{iz}\rho)^2 + (b_i - b_{iz})^2 - 1)^2 \Big)$$
$$= A_1\rho^4 + B_1\rho^3 + C_1\rho^2 + D_1\rho + E_1$$

The error function J_2 is defined as follows: if there exists a pose transformation $T_{ij} \in R^{3\times3}$ between node i and node j, where the translational component is $t_{ij} \in R^2$, then the length between node i and node j is $\rho_{ij} = \left\|t_{ij}\right\|$. At the same time, the norm (a linear function of the unknown ρ) can be computed based on the solved coordinates p_i of node i and p_j of node j. The error function is defined using the difference between the two norms.

$$J_2 = \sum_{T_{ij}\in E} \left(\left\|p_i - p_j\right\|^2 - \rho_{ij}^2 \right)^2$$
$$= \sum_{T_{ij}\in E} \left(\left(a_i\rho - a_j\rho\right)^2 + \left(b_i - b_j\right)^2 - \rho_{ij}^2 \right)^2$$
$$= A_2\rho^4 + B_2\rho^3 + C_2\rho^2 + D_2\rho + E_2$$

By summing the error function J_1 and the error function J_2, we obtain the total error function J.

$$J = J_1 + J_2$$
$$= A\rho^4 + B\rho^3 + C\rho^2 + D\rho + E$$

To minimize this error function, we compute the derivative of J.

$$\frac{dJ}{d\rho} = 4A\rho^3 + 3B\rho^2 + 2C\rho + D = 0$$

Then, the three roots, ρ_1, ρ_2 and ρ_3, are calculated using the cubic equation formula. After discarding the complex roots among the three, the second derivative of J is used to evaluate the remaining roots.

$$\frac{dJ^2}{d\rho^2} = 12A\rho^2 + 6B\rho + 2C$$

If the second derivative of J at root $\rho_i, i \in [1,3]$ is positive, then ρ_i corresponds to the minimum value of the function J. If the second derivative of J at root ρ_i is negative, then ρ_i represents a local maximum of the function J. Once the optimized map scale ρ is obtained, it can be substituted into the coordinates $p_v^c \in C^{2n}, p^c \in C^n$. The pseudocode for computing the map scale and node coordinates is as follows.

Process 7. Calculation of map scale and node coordinates (without using GPS data)

Inputs: Linear equations $Ap_v^c = b, Cp^c = Dp_v^c$; covariance matrices Σ_1, Σ_3

Outputs: Virtual node coordinate vector $p_v^c \in C^{2n}$; node coordinate vector $p^c \in C^n$; map scale ρ

Let the coordinate vector of node $i \in V$ be $p = [p_1, p_2, \dots, p_n] \in R^{2\times n}$.

Let the complex coordinate vector of node $i \in V$ be $p^c = [p_1^c, p_2^c, \dots, p_n^c] \in C^n$.

Let the virtual node coordinate vector be $p_v = \left[p_{1x}, p_{1y}, \dots, p_{nx}, p_{ny}\right] \in R^{2\times 2n}$.

Let the complex coordinate vector of virtual node be

$p_v^c = [p_{1x}^c, p_{1y}^c, \dots, p_{nx}^c, p_{ny}^c] \in C^{2n}$.

Assume the complex coordinate of node 1 is $p_1^c = 0 + i \times 0$, the complex coordinates of the virtual node 1_x is $p_{1x}^c = \rho + i \times 0$, the complex coordinates of the virtual node 1_y is $p_{1y}^c = 0 + i \times \rho$.

Let the complex coordinate vector of virtual nodes $i_x, i_y \in V_v/(1_x, 1_y)$ be $p_{sub}^c = [p_{2x}^c, p_{2y}^c, \dots, p_{nx}^c, p_{ny}^c] \in C^{2n-2}$.

Substituting the complex coordinates p_{1x}^c and p_{1y}^c into the equations, the linear equation $Ap_v^c = b$ is reduced to: $A_{sub}p_{sub}^c = b_{sub}$.

The complex coordinates of the virtual node, denoted as p_{sub}^c, are calculated using the linear equation system $A_{sub}p_{sub}^c = b_{sub} + v_{sub1}$. The least-squares solution to this system is given by: $(A_{sub}^T\Sigma_1^{-1}A_{sub})^{-1}A_{sub}^T\Sigma^{-1}b_{sub}$(a linear function of ρ).

The complex coordinates of node $i \in V$, denoted as p^c, are calculated using the linear equation system $Cp^c = Dp_v^c$. The least-squares solution to this system is given by: $p^c = (C^TC)^{-1}C^TDp_v^c$ (a linear function of ρ).

The error function J₁ is defined as: $J_1 = \sum_{i\in V}\left(\left(||p_i - p_{ix}||^2 - 1\right)^2 + \left(||p_i - p_{iy}||^2 - 1\right)^2\right) = A_1\rho^4 + B_1\rho^3 + C_1\rho^2 + D_1\rho + E_1$.

The error function J₂ is defined as: $J_2 = \sum_{T_{ij}\in E}\left(||p_i - p_j||^2 - \rho_{ij}^2\right)^2 = A_2\rho^4 + B_2\rho^3 + C_2\rho^2 + D_2\rho + E_2$.

By summing these two error functions, the total error function J is obtained: $J = J_1 + J_2 = A\rho^4 + B\rho^3 + C\rho^2 + D\rho + E$.

To find the parameter ρ that minimizes the error function, the derivative of J with respect to ρ is computed: $\frac{dJ}{d\rho} = 4A\rho^3 + 3B\rho^2 + 2C\rho + D = 0$.

The cubic equation is solved to obtain three roots: ρ_1, ρ_2, ρ_3. The complex roots among these are discarded, and the root ρ_i corresponding to the minimum value of J is selected by evaluating the second derivative of the function.

The selected parameter ρ is substituted back into the coordinates p_v^c and p^c, and the final coordinates p_v^c and p^c are computed.

5.2 Using GPS sensors

When GPS sensors are not used, the linear equations between each node $i \in V$ and the virtual nodes $i_x, i_y \in V_v$ are expressed as follows.

$$A_{sub}p_v^c = b + v_1 \quad (1)$$

$$Cp^c = Dp_v^c + v_2 \quad (2)$$

We define the following coordinates: $p_1^c = 0 + i \times 0, p_{1x}^c = \rho + i \times 0$, and $p_{1y}^c = 0 + i \times \rho$. Using Equation (1), the coordinate vector p_v^c is calculated. Then, substituting the coordinate vector p_v^c into Equation (2), we obtain the coordinate vector p^c.

When GPS sensors are used, the linear equations between each node $i \in V$ and the virtual nodes $i_x, i_y \in V_v$ need to be further extended as follows.

$$Ep^c = 0 + v_3 \quad (3)$$

At this point, after setting the coordinates p_1^c, p_{1x}^c, and p_{1y}^c, it is necessary to combine equations (1) and (3) to compute the node vector p_v^c and p^c. We define the coordinate vector $p_{acc}^c = [p_v^{cT}, p^{cT}]^T \in C^{3\times n}$ as the concatenated vector of p_v^c and p^c. Subsequently, equations (1), (2), and (3) are merged into the following form.

$$Fp_{acc}^c = G + v_4$$

Where the error vector is defined as $v_4 = [v_1^T, v_2^T, v_3^T]^T$, and $v_4 \sim N(0, \Sigma_4)$. Assuming the independence of the individual linear equations, the covariance matrix Σ_4 is a diagonal matrix and satisfies the following relation: $\Sigma_4 = diag\{\Sigma_1, \Sigma_2, \Sigma_3\}, \Sigma_2 = 0$_(|v2 ∗ v2|).

By substituting the planar coordinates p_1^c, p_{1x}^c, and p_{1y}^c into the above equations, the equation transforms into the following form.

$$F_{sub}p_{sub,acc}^c = G_{sub} + v_{sub4}$$

The coordinate vector $p_{sub,acc}^c \in C^{3n-3}$ is obtained by removing $p_1^c, p_{1x}^c, p_{1y}^c$ from the coordinate vector p_{acc}^c. The error vector $v_{sub4} \sim N(0, \Sigma_{sub4})$ represents the noise corresponding to each linear equation. The number of linear equations in the system remains unchanged, and the variance corresponding to each linear equation also remains unchanged. Therefore, the error vector satisfies $v_{sub4} = v_4$, and the covariance matrix satisfies $\Sigma_{sub4} = \Sigma_4$. The least squares solution of this linear equation system is given as follows.

$$p_{sub,acc}^c = \left(F_{sub}^T(\Sigma_4)^{-1}F_{sub}\right)^{-1}F_{sub}^T\Sigma_4^{-1}G_{sub}$$

The pseudocode for calculating the map scale and node coordinates is as follows.

Process 8. Calculation of map scale and node coordinates (using GPS data)

Input: Linear equation systems $Ap_v^c = b, Cp^c = Dp_v^c, Ep^c = 0$; variance matrices Σ_1, Σ_2

Output: Virtual node coordinate vector $p_v \in R^{2\times 2n}$; node coordinate vector $p \in R^{2\times n}$; map scale ρ

Define the coordinate vector of node $i \in V$ as $p = [p_1, p_2, \dots, p_n] \in R^{2\times n}$.

Define the complex coordinate vector of node $i \in V$ as $p^c = [p_1^c, p_2^c, \dots, p_n^c] \in C^n$.

Define the virtual node coordinate vector as $p_v =$

$[p_{1x}, p_{1y}, \dots, p_{nx}, p_{ny}] \in R^{2\times n}$.

Define the planar virtual node coordinate vector as $p_v^c = [p_{1x}^c, p_{1y}^c, \dots, p_{nx}^c, p_{ny}^c] \in C^{2n}$.

Assume the complex coordinate of node 1 is $p_1^c = 0 + i \times 0$; assume the complex coordinates of virtual node 1_x is $p_{1x}^c = \rho + i \times 0$; assume the complex coordinates of virtual node 1_y is $p_{1y}^c = 0 + i \times \rho$.

For virtual nodes $i_x, i_y \in V_v/(1_x, 1_y)$, define their complex coordinate vector as $p_{sub}^c = [p_{2x}^c, p_{2y}^c, \dots, p_{nx}^c, p_{ny}^c] \in C^{2n-2}$.

Substitute the complex coordinates p_{1x}^c, p_{1y}^c into the equations. The linear equation system $Ap_v^c = b$ is transformed into: $A_{sub}p_{sub}^c = b_{sub}$.

Define the coordinate vector as $p_{acc}^c = \left[p_v^{cT}, p^{cT}\right]^T \in C^{3\times n}$. By combining equations (1), (2), and (3), we obtain the following linear relationship: $Fp_{acc}^c = G$.

Let the coordinate vector $p_{sub,acc}^c \in C^{3n-3}$ be the result of removing $p_1^c, p_{1x}^c, p_{1y}^c$ from the vector p_{acc}^c. The equation becomes: $F_{sub}p_{sub,acc}^c = G_{sub}$.

The planar components of the nodes, $p_{sub,acc}^c$, can be computed by solving the linear system $F_{sub}p_{sub,acc}^c = G_{sub}$. The least-squares solution to this linear system is given by: $p_{sub,acc}^c = \left(F_{sub}^T(\Sigma_6)^{-1}F_{sub}\right)^{-1}F_{sub}^T\Sigma_6^{-1}G_{sub}$ (a linear function of ρ).

Define the error function J_1 as $J_1 = \sum_{i\in V}\left(\left(\left\|p_i - p_{ix}\right\|^2 - 1\right)^2 + \left(\left\|p_i - p_{iy}\right\|^2 - 1\right)^2\right) = A_1\rho^4 + B_1\rho^3 + C_1\rho^2 + D_1\rho + E_1$.

Define the error function J_2 as $J_2 = \sum_{T_{ij}\in E}\left(\left\|p_i - p_j\right\|^2 - \rho_{ij}^2\right)^2 = A_2\rho^4 + B_2\rho^3 + C_2\rho^2 + D_2\rho + E_2$.

By summing these two error functions, the total error function J is $J = J_1 + J_2 = A\rho^4 + B\rho^3 + C\rho^2 + D\rho + E$.

To find the parameter ρ that minimizes this error function, compute the derivative of J with respect to ρ: $\frac{dJ}{d\rho} = 4A\rho^3 + 3B\rho^2 + 2C\rho + D = 0$.

Using the cubic equation formula, solve for the three roots ρ_1, ρ_2, ρ_3. Discard any complex roots among these solutions, and use the second derivative of J to select the root ρ_i corresponding to the minimum value of J.

Substitute the parameter ρ into the coordinates p_v^c and p^c to compute the final values of p_v^c and p^c.

6. Outlier removal algorithm

In this paper, the linear equations are constructed through the relative pose transformations T_{ij} between nodes. If the set of pose transformations $E = \{T_{ij}, i, j \in V\}$ in the graph $G = [V, E]$ contains outliers, the corresponding coefficients of the linear equations will also exhibit outliers. When solving the system of linear equations using the least squares method, each linear equation corresponds to an error term. Outliers can cause significant deviations in the solution of the equations.

In nonlinear optimization-based SLAM methods, each pose transformation T_{ij} is used to construct an error function $h(T_{ij})$ via an observation function. These error functions then form error terms $e(T_{ij}) = h(T_{ij})^2$, and the sum of these error terms produces the overall error function JJJ. However, a robust function is often applied to the error terms, as shown below.

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

The robust function ρ increases rapidly near zero, but its growth slows significantly beyond a certain range. After introducing the robust function, the error function J is expressed as follows.

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

Due to the presence of robust functions, even if a certain pose transformation T_{ij} is an outlier, the corresponding error term will not have a significantly large value. Therefore, the outliers have limited influence on the minimum value of the overall error function. Moreover, the original error term $e(T_{ij}) = h(T_{ij})^2$ is a nonlinear function, and the error term after introducing the robust function, $\rho\left(h(T_{ij})^2\right)$, remains a nonlinear function. We still employ numerical methods to solve for the minimum value of the error function.

However, for the linear equations discussed in this paper, the robust function approach cannot be applied. If robust functions are added to the error terms for each linear equation, the resulting function would become nonlinear. The purpose of formulating pose transformations into linear equations is to reduce computational complexity. If nonlinear equations were used, this goal would no longer be achievable.

Paper [3] presents a method for removing outliers in pose transformations. Since the specific steps of the method are numerous, we provide a brief overview of the general approach. The algorithm takes as input the set of all pose transformations in the graph $G = [V, E]$, denoted as $E = \{T_{ij}, i, j \in V\}$, and outputs the set of pose transformations with outliers removed, denoted as $E' = \{T_{ij}, i, j \in V\}$. By constructing linear equations using all pose transformations in E', the influence of outliers on the equations can be effectively avoided.

The algorithm requires each pose transformation T_{ij} in the input set $E = \{T_{ij}, i, j \in V\}$ to be assigned a prior probability p_{ij}, representing the likelihood of T_{ij} being an outlier. This probability does not need to be an absolutely precise value; an approximate value can be derived using the variance σ_{ij}^2

associated with the pose transformation T_{ij}. For example, the following scheme 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} in the outlier removal algorithm is used to identify paths between different nodes (node i, node j); the pose transformation T_{ij} represents the edge in the path. The algorithm aims to find a path composed of edges with fewer outliers to achieve the transformation from node i to node j.

The general method of this outlier removal algorithm is as follows: Suppose there exist two nodes $p_1, p_2 \in V$ in $G = [V, E]$, and there are three rotation-translation matrices $T_{12}^1, T_{12}^2, T_{12}^3$ between nodes p_1 and p_2.

At the same time, a connection can also be formed between p_1 and p_2 via other intermediate nodes (e.g., p_3, p_4, p_5). In this case, the pose transformation from p_1 to p_2 can be calculated as follows.

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

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

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

The nodes p_1 and p_2 can form a connection through two other 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.

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

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

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

Thus, we obtain a total of 9 groups of pose transformations between node 1 and node 2, denoted as T_{12}^i where $i \in [1,9]$. If all the pose transformations used (set E) contain no outliers, these transformations T_{12}^i should be highly similar. However, if any of the transformations is an outlier, the corresponding pose transformation T_{12}^i will significantly differ from the others. We can convert the pose transformations T_{12}^i into translation vectors and rotation vectors. Then, for each component, we apply a statistical method based on the Interquartile Range (IQR) algorithm to remove outliers.

7. Calculation of local coordinate system

We have calculated the coordinates of each node in the global coordinate system Σ_g for the graph $G = [V, E]$, denoted as $p = [p_1, p_2, \ldots, p_n] \in R^{2\times n}$. Let the global coordinates of node i be $p_i = [x_i, y_i]^T$, and let the neighboring nodes of node i be denoted as $j \in N_i$, where the global coordinates of node j in Σ_g are $p_j = [x_j, y_j]^T$. The relative coordinates of node j with respect to node i are given by $p_j^r = [x_j - x_i, y_j - y_i]^T$. The relative coordinate vectors of these neighboring nodes in the global coordinate system can then be represented as $P_i^n = [p_{j1}^r, \ldots, p_{jn}^r] \in R^{2\times|N_i|}$, where $j_1, \ldots, j_n \in N_i$. Let the pose transformation from node i to its neighboring node $j \in N_i$ be represented as $T_{ij} = \begin{bmatrix} R_{ij} & t_{ij} \\ 1 & 0 \end{bmatrix} \in R^{3\times 3}$, where the translation vector is $t_{ij} = [x_{ij}, y_{ij}]^T$. Then, the relative position vectors of the neighboring nodes $j \in N_i$ in the local coordinate system Σ_i can be expressed as $Q_i^n = [t_{ij1}, \ldots, t_{ijn}] = [q_{ij1}, \ldots, q_{ijn}] \in R^{2\times|N_i|}$, where $j_1, \ldots, j_n \in N_i$.

We have computed the global coordinates of each node $i \in V$ and its corresponding virtual nodes $i_x, i_y \in V_v$ in the global coordinate system Σ_g, denoted as $p_{ix} = [x_{ix}, y_{ix}]^T$ and $p_{iy} = [x_{iy}, y_{iy}]^T$. The global coordinates of node iii are represented as $p_i = [x_i, y_i]^T$. The relative coordinates of the virtual nodes i_x and i_y with respect to node i are given as follows.

$$p_{ix}^r = [x_{ix} - x_i, y_{ix} - y_i]^T$$

$$p_{iy}^r = [x_{iy} - x_i, y_{iy} - y_i]^T$$

The relative coordinate vector of the virtual nodes i_x and i_y in the global coordinate system is $P_i^v = [p_{ix}^r, p_{iy}^r]$. In the local coordinate system Σ_i, the coordinates of the virtual nodes i_x and i_y are $p_{ix}^i = [1,0]^T$ and $p_{iy}^i = [0,1]^T$, respectively. Let the coordinate vector of the virtual nodes i_x and i_y in the local coordinate system Σ_i be $Q_i^v = [p_{ix}^i, p_{iy}^i]$.

The coordinate vectors P_i^n and P_i^v are combined to form $P_i = [P_i^n, P_i^v] \in R^{2\times|N_i|+2}$, and similarly, the coordinate vectors Q_i^n and Q_i^v are combined to form $Q_i = [Q_i^n, Q_i^v] \in R^{2\times|N_i|+2}$. Using these, the rotation matrix R_i for the local coordinate system of node i can be calculated via point set registration methods.

Here, the construction of the coordinate vectors P_i and Q_i incorporates the neighboring nodes $j \in N_i$ of node i, as well as the virtual nodes i_x and i_y. Although it is possible to include the virtual coordinates j_x and j_y of the neighboring nodes $j \in N_i$ in the coordinate vectors, they are excluded here for the following reasons: First, a rotation matrix R_i can be determined as long as there are three nodes, such as the virtual nodes i_x and i_y. Second, the information of the neighboring nodes $j \in N_i$ already sufficiently represents the positions of these nodes. Adding the virtual coordinates j_x and j_y of nodes $j \in N_i$ may unnecessarily increase computational complexity.

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

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

We need to find the rotation matrix $R_i \in R^{2\times 2}$ that minimizes the error function $E(R_i)$. This problem falls under the category of point cloud registration. To solve it, we can use the Singular Value Decomposition (SVD) method from the

Iterative Closest Point (ICP) algorithm to compute the matrix R_i. The matrix R_i represents the rotation required to transform the 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 with respect to the global coordinate system Σ_g is $(R_i)^{-1}$. According to the SVD-based method in the ICP algorithm, the rotation matrix R_i that minimizes the error function E(R) is given as follows.

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

For the matrix $H \in R^{2\times 2}$, perform singular value decomposition (SVD) to obtain the orthogonal matrices U and V. The rotation matrix is calculated as $R_i = VR^T$. However, if the point set P_i or Q_i is degenerate, the resulting matrix R_i may become a reflection matrix (i.e., $\det(R_i) = -1$). We can adjust the reflection matrix R_i into a proper rotation matrix using the following procedure.

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

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

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

Process 9. Calculation of local coordinate system

Input: The coordinates of node i, denoted as $p_i \in R^2$; the neighboring nodes of node i, denoted as $j \in N_i$; the coordinates of neighboring nodes $j \in N_i$, denoted as $p_j \in R^2$; the pose transformation of neighboring nodes $j \in N_i$, denoted as T_{ij}; the coordinates of virtual nodes i_x and i_y, denoted as p_{ix} and p_{iy}

Output: The rotation matrix $(R_i)^{-1}$ 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 expressed as: $p_j^r = \left[x_j - x_i, y_j - y_i\right]^T$.

The relative coordinate vector of neighboring nodes $j \in N_i$ in the global coordinate system Σ_g is represented as: $P_i^n = \left[p_{j1}^r, \dots, p_{jn}^r\right] \in R^{2\times|N_i|}, j_1, \dots, j_n \in N_i$.

The coordinates of neighboring nodes $j \in N_i$ in the local coordinate system Σ_i are given as: $Q_i^n = \left[t_{ij1}, \dots, t_{ijn}\right] = \left[q_{ij1}, \dots, q_{ijn}\right] \in R^{2\times|N_i|}, j_1, \dots, j_n \in N_i$.

The coordinates of virtual nodes i_x and i_y relative to node i are: $p_{ix}^r = [x_{ix} - x_i, y_{ix} - y_i]^T, p_{iy} = \left[x_{iy} - x_i, y_{iy} - y_i\right]^T, p_{iz} = [x_{iz} - x_i, y_{iz} - y_i]^T$; combining these, we obtain the vector: $P_i^v = \left[p_{ix}^r, p_{iy}^r\right] \in R^{2\times 2}$.

The coordinates of virtual nodes i_x and i_y in the local coordinate system Σ_i are defined as: $p_{ix}^i = [1,0]^T, p_{iy}^i = [0,1]^T$; combining these, we obtain the vector: $Q_i^v = \left[p_{ix}^i, p_{iy}^i\right] \in R^{2\times 2}$.

The coordinate vectors are combined as follows: $P_i = [P_i^n, P_i^v] \in R^{2\times(2|N_i|+2)}$, $Q_i = [Q_i^n, Q_i^v] \in R^{2\times(2|N_i|+2)}$.

Compute the matrix $H = P_i Q_i^T$.

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

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

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

REFERENCES

[1] 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.

[2] Z. Wu, "A linear SLAM backend optimization algorithm on 2D space, " 10.5281/zenodo.14468254

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