A linear SLAM backend optimization algorithm based on virtual coordinate system and barycentric coordinates in 3D 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. The barycentric coordinates in 3D spaces

The barycentric coordinates are a geometric concept used to describe the position of a point relative to other points. Let four known points $\mathrm{j, k, l, h}$ in 3D space have Euclidean coordinates $\mathrm{p_j, p_k, p_l, p_h} \in R^3$. Assume that the coordinates of point i satisfy the following equation.

$$\mathrm{p_i = a_{ij}p_j + a_{ik}p_k + a_{il}p_l + a_{ih}p_h}$$

$$\mathrm{a_{ij} + a_{ik} + a_{il} + a_{ih} = 1}$$

Then, $\{\mathrm{a_{ij}, a_{ik}, a_{il}, a_{ih}}\}$ are the barycentric coordinates of node iii with respect to nodes $\mathrm{i, j, k, h}$. In 3D space, the barycentric coordinates of point i can be computed using the signed volumes of tetrahedra. Specifically, for the example shown in Figure 1, the barycentric coordinates $\{\mathrm{a_{il}, a_{ij}, a_{ik}, a_{ih}}\}$ of point i are calculated as follows.

$$\mathrm{a_{ij}} = \frac{\mathrm{V_{iklh}}}{\mathrm{V_{jklh}}}, \mathrm{a_{ik}} = \frac{\mathrm{V_{jilh}}}{\mathrm{V_{jklh}}}, \mathrm{a_{il}} = \frac{\mathrm{V_{jkih}}}{\mathrm{V_{jklh}}}, \mathrm{a_{ih}} = \frac{\mathrm{V_{jkli}}}{\mathrm{V_{jklh}}}$$

The variables $\mathrm{V_{iklh}, V_{jilh}, V_{jkih}, V_{jkli}, V_{jklh}}$ represent the oriented volumes of the tetrahedra iklh, jilh, jkih, jkli, jklh, respectively. If the points $\mathrm{j, k, l, h}$ satisfy the right-hand rule, the sign of the oriented volume $\mathrm{V_{jklh}}$ is positive; otherwise, the sign of $\mathrm{V_{jklh}}$ is negative.

Assuming the coordinates of five points $\mathrm{i, j, k, l. h}$ in a three-dimensional space are given as $\mathrm{p_i, p_j, p_k, p_l, p_h} \in \mathrm{R}^3$, the volumes of the tetrahedra $\mathrm{V_{iklh}, V_{jilh}, V_{jkih}, V_{jkli}, V_{jklh}}$ can be calculated using determinants.

$$\mathrm{V_{iklh}} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_i & p_k & p_l & p_h \end{vmatrix}$$

$$\mathrm{V_{jilh}} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_j & p_i & p_l & p_h \end{vmatrix}$$

$$\mathrm{V_{jkih}} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_j & p_k & p_i & p_h \end{vmatrix}$$

$$\mathrm{V_{jkli}} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_j & p_k & p_l & p_i \end{vmatrix}$$

$$\mathrm{V_{jklh}} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_j & p_k & p_l & p_h \end{vmatrix}$$

Here, it is required that the volume of tetrahedron jklh must not be zero. Otherwise, a division by zero would occur when calculating the barycentric coordinates $\mathrm{a_{ij}, a_{ik}, a_{il}, a_{ih}}$.

3. Representing pose transformation with barycentric coordinates

3.1 Representing nodes j,jx,jy,jz with nodes i,ix,iy,iz

In SLAM algorithms, various methods can be used to obtain the pose transformation between two keyframes (in 3D, represented as a rotation-translation matrix $T \in R^{4\times4}$). The table below lists the 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	Transformation algorithm
2d-2d	epipolar geometry
2d-2d(planar)	homography matrix
2d-3d	DLT、EpnP
3d-3d	ICP

If a LiDAR sensor is used to obtain two consecutive point clouds, various point cloud registration algorithms can be employed to determine the pose transformation. Additionally, the pose transformation between the two frames can also be obtained using 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 is the pose transformation between two keyframes (nodes). The purpose of backend optimization is to use this pose transformation information to estimate an accurate global map. Here, we denote a keyframe as node $i \in V$, where each node i has its own local coordinate system Σ_i. There exists an unknown rotation matrix R_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 three-dimensional space be $p_i = [x_i, y_i, z_i]^T$. Let the set of all pose transformations T_{ij} be $E = \{T_{ij}, i, j \in V\}$. We can 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.

Then, we transform the pose transformation $T_{ij} \in R^{4\times4}$ between any two nodes (node i and node j) into a set of linear equation constraints. Given the coordinates of node i as $p_i \in R^3$, we define a virtual node i_x along the x-axis of the local coordinate frame Σ_i, with coordinates p_{ix} such that $||p_{ix} - p_i|| = 1$. Similarly, we define a virtual node i_y along the y-axis of Σ_i, with coordinates p_{iy} such that $||p_{iy} - p_i|| = 1$, and a virtual node i_z along the z-axis of Σ_i, with coordinates p_{iz} such that $||p_{iz} - p_i|| = 1$. Therefore, in the local coordinate frame Σ_i, node i serves as the origin, and i_x, i_y, and i_z represent the unit vector nodes along the three respective axes of the local coordinate frame.

Similarly, the coordinates of node j are given as $p_j \in R^3$. We define a virtual node p_{jx} along the x-axis of the local coordinate frame Σ_j, such that $||p_{jx} - p_j|| = 1$. Similarly, we define a virtual node p_{jy} along the y-axis with $||p_{jy} - p_j|| = 1$, and a virtual node p_{jz} along the z-axis with $||p_{jz} - p_j|| = 1$. Therefore, in the local coordinate frame Σ_j, node j serves as the origin, and p_{jx}, p_{jy}, and p_{jz} represent the unit coordinate vectors along the three respective axes of the local coordinate frame.

Let the pose transformation from node i to node j be denoted as $T_{ij} \in R^{4\times4}$.

$$T_{ij} = \begin{bmatrix} R_{ij} & t_{ij} \\ 0 & 1 \end{bmatrix} \in R^{4\times4}, R_{ij} \in R^{3\times3}$$

$$t_{ij} = \left[x_{ij}, y_{ij}, z_{ij}\right]^T \in R^3$$

Then, the positions of node j and the virtual nodes j_x, j_y, and j_z in the local coordinate system Σ_i are as follows.

$$p_j^i = t_{ij}$$

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

$$p_{jy}^i = R_{ij}[0,1,0]^T + t_{ij}$$

$$p_{jz}^i = R_{ij}[0,0,1]^T + t_{ij}$$

The coordinates of node i and the virtual nodes i_x, i_y, and i_z in the local coordinate system Σ_i are as follows.

$$p_i^i = [0,0,0]^T$$

$$p_{ix}^i = [1,0,0]^T$$

$$p_{iy}^i = [0,1,0]^T$$

$$p_{iz}^i = [0,0,1]^T$$

If the global coordinates of all points in the local coordinate system Σ_i (denoted as $p_i, p_{ix}, p_{iy}, p_{iz}$) are known, we need to represent the global coordinates of all points in the local coordinate system Σ_j (denoted as $p_j, p_{jx}, p_{jy}, p_{jz}$) using barycentric coordinates. By doing so, we can recursively determine the local coordinate system for all nodes starting from the first node. Since node i and the virtual nodes i_x, i_y, i_z represent points on different axes of the coordinate system, the tetrahedron formed by i-ix-iy-iz has a non-zero volume, satisfying the conditions for computing barycentric coordinates.

Because barycentric coordinates remain invariant under arbitrary rotations and translations of the node set $V_{ij} = \{i, i_x, i_y, i_z, j, j_x, j_y, j_z\}$, we can compute the barycentric coordinates using the relative positions of the nodes in the local coordinate system Σ_i (denoted as $p_i^i, p_{ix}^i, p_{iy}^i, p_{iz}^i, p_j^i, p_{jx}^i, p_{jy}^i, p_{jz}^i$). The barycentric coordinates calculated in this way remain valid for the positions of the nodes in the global coordinate system (denoted as $p_i, p_i x, p_{iy}, p_{iz}, p_j, p_{jx}, p_{jy}, p_{jz}$).

First, we construct the barycentric coordinates of node j using the four nodes i, i_x, i_y, i_z in the local coordinate system Σ_i.

$$p_j = a_{j,i}p_i + a_{j,ix}p_{ix} + a_{j,iy}p_{iy} + a_{j,iz}p_{iz} \qquad (1)$$

$$\mathrm{a}_{\mathrm{j,i}} = \frac{\mathrm{V}_{\mathrm{j,ix,iy,iz}}}{\mathrm{V}_{\mathrm{i,ix,iy,iz}}} \tag{1.1}$$

$$\mathrm{a}_{\mathrm{j,ix}} = \frac{\mathrm{V}_{\mathrm{i,j,iy,iz}}}{\mathrm{V}_{\mathrm{i,ix,iy,iz}}} \tag{1.2}$$

$$\mathrm{a}_{\mathrm{j,iy}} = \frac{\mathrm{V}_{\mathrm{i,ix,j,iz}}}{\mathrm{V}_{\mathrm{i,ix,iy,iz}}} \tag{1.3}$$

$$\mathrm{a}_{\mathrm{j,iz}} = \frac{\mathrm{V}_{\mathrm{i,ix,iy,j}}}{\mathrm{V}_{\mathrm{i,ix,iy,iz}}} \tag{1.4}$$

$$\mathrm{V}_{\mathrm{i,ix,iy,iz}} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_i^i & p_{ix}^i & p_{iy}^i & p_{iz}^i \end{vmatrix} \tag{1.5}$$

$$\mathrm{V}_{\mathrm{j,ix,iy,iz}} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_j^i & p_{ix}^i & p_{iy}^i & p_{iz}^i \end{vmatrix} \tag{1.6}$$

$$\mathrm{V}_{\mathrm{i,j,iy,iz}} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_i^i & p_j^i & p_{iy}^i & p_{iz}^i \end{vmatrix} \tag{1.7}$$

$$\mathrm{V}_{\mathrm{i,ix,j,iz}} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_i^i & p_{ix}^i & p_j^i & p_{iz}^i \end{vmatrix} \tag{1.8}$$

$$\mathrm{V}_{\mathrm{i,ix,iy,j}} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_i^i & p_{ix}^i & p_{iy}^i & p_j^i \end{vmatrix} \tag{1.9}$$

Next, we construct the barycentric coordinates of node $\mathrm{j_x}$ using the four nodes $\mathrm{i, i_x, i_y, i_z}$ in the local coordinate system Σ_i.

$$\mathrm{p_{jx}} = \mathrm{a_{jx,i}p_i} + \mathrm{a_{jx,ix}p_{ix}} + \mathrm{a_{jx,iy}p_{iy}} + \mathrm{a_{jx,iz}p_{iz}} \tag{2}$$

$$\mathrm{a}_{\mathrm{jx,i}} = \frac{\mathrm{V}_{\mathrm{jx,ix,iy,iz}}}{\mathrm{V}_{\mathrm{i,ix,iy,iz}}} \tag{2.1}$$

$$\mathrm{a}_{\mathrm{jx,ix}} = \frac{\mathrm{V}_{\mathrm{i,jx,iy,iz}}}{\mathrm{V}_{\mathrm{i,ix,iy,iz}}} \tag{2.2}$$

$$\mathrm{a}_{\mathrm{jx,iy}} = \frac{\mathrm{V}_{\mathrm{i,ix,jx,iz}}}{\mathrm{V}_{\mathrm{i,ix,iy,iz}}} \tag{2.3}$$

$$\mathrm{a}_{\mathrm{jx,iz}} = \frac{\mathrm{V}_{\mathrm{i,ix,iy,jx}}}{\mathrm{V}_{\mathrm{i,ix,iy,iz}}} \tag{2.4}$$

$$\mathrm{V}_{\mathrm{i,ix,iy,iz}} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_i^i & p_{ix}^i & p_{iy}^i & p_{iz}^i \end{vmatrix} \tag{2.5}$$

$$\mathrm{V}_{\mathrm{jx,ix,iy,iz}} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_{jx}^i & p_{ix}^i & p_{iy}^i & p_{iz}^i \end{vmatrix} \tag{2.6}$$

$$\mathrm{V}_{\mathrm{i,jx,iy,iz}} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_i^i & p_{jx}^i & p_{iy}^i & p_{iz}^i \end{vmatrix} \tag{2.7}$$

$$\mathrm{V}_{\mathrm{i,ix,jx,iz}} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_i^i & p_{ix}^i & p_{jx}^i & p_{iz}^i \end{vmatrix} \tag{2.8}$$

$$\mathrm{V}_{\mathrm{i,ix,iy,jx}} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_i^i & p_{ix}^i & p_{iy}^i & p_{jx}^i \end{vmatrix} \tag{2.9}$$

Then, we construct the barycentric coordinates of node $\mathrm{j_y}$ using the four nodes $\mathrm{i, i_x, i_y, i_z}$ in the local coordinate system Σ_i.

$$\mathrm{p_{jy}} = \mathrm{a_{jy,i}p_i} + \mathrm{a_{jy,ix}p_{ix}} + \mathrm{a_{jy,iy}p_{iy}} + \mathrm{a_{jy,iz}p_{iz}} \tag{3}$$

$$\mathrm{a}_{\mathrm{jy,i}} = \frac{\mathrm{V}_{\mathrm{jy,ix,iy,iz}}}{\mathrm{V}_{\mathrm{i,ix,iy,iz}}} \tag{3.1}$$

$$\mathrm{a}_{\mathrm{jy,ix}} = \frac{\mathrm{V}_{\mathrm{i,jy,iy,iz}}}{\mathrm{V}_{\mathrm{i,ix,iy,iz}}} \tag{3.2}$$

$$\mathrm{a}_{\mathrm{jy,iy}} = \frac{\mathrm{V}_{\mathrm{i,ix,jy,iz}}}{\mathrm{V}_{\mathrm{i,ix,iy,iz}}} \tag{3.3}$$

$$\mathrm{a}_{\mathrm{jy,iz}} = \frac{\mathrm{V}_{\mathrm{i,ix,iy,jy}}}{\mathrm{V}_{\mathrm{i,ix,iy,iz}}} \tag{3.4}$$

$$\mathrm{V}_{\mathrm{i,ix,iy,iz}} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_i^i & p_{ix}^i & p_{iy}^i & p_{iz}^i \end{vmatrix} \tag{3.5}$$

$$\mathrm{V}_{\mathrm{jy,ix,iy,iz}} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_{jy}^i & p_{ix}^i & p_{iy}^i & p_{iz}^i \end{vmatrix} \tag{3.6}$$

$$\mathrm{V}_{\mathrm{i,jy,iy,iz}} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_i^i & p_{jy}^i & p_{iy}^i & p_{iz}^i \end{vmatrix} \tag{3.7}$$

$$\mathrm{V}_{\mathrm{i,ix,jy,iz}} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_i^i & p_{ix}^i & p_{jy}^i & p_{iz}^i \end{vmatrix} \tag{3.8}$$

$$\mathrm{V}_{\mathrm{i,ix,iy,jy}} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_i^i & p_{ix}^i & p_{iy}^i & p_{jy}^i \end{vmatrix} \tag{3.9}$$

Then, we use the four nodes $\mathrm{i, i_x, i_y, i_z}$ in the local coordinate system Σ_i to construct the barycentric coordinates of node $\mathrm{j_z}$.

$$\mathrm{p_{jz}} = \mathrm{a_{jz,i}p_i} + \mathrm{a_{jz,ix}p_{ix}} + \mathrm{a_{jz,iy}p_{iy}} + \mathrm{a_{jz,iz}p_{iz}} \tag{4}$$

$$\mathrm{a}_{\mathrm{jz,i}} = \frac{\mathrm{V}_{\mathrm{jz,ix,iy,iz}}}{\mathrm{V}_{\mathrm{i,ix,iy,iz}}} \tag{4.1}$$

$$\mathrm{a}_{\mathrm{jz,ix}} = \frac{\mathrm{V}_{\mathrm{i,jz,iy,iz}}}{\mathrm{V}_{\mathrm{i,ix,iy,iz}}} \tag{4.2}$$

$$\mathrm{a}_{\mathrm{jz,iy}} = \frac{\mathrm{V}_{\mathrm{i,ix,jz,iz}}}{\mathrm{V}_{\mathrm{i,ix,iy,iz}}} \tag{4.3}$$

$$\mathrm{a}_{\mathrm{jz,iz}} = \frac{\mathrm{V}_{\mathrm{i,ix,iy,jz}}}{\mathrm{V}_{\mathrm{i,ix,iy,iz}}} \tag{4.4}$$

$$\mathrm{V}_{\mathrm{i,ix,iy,iz}} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_i^i & p_{ix}^i & p_{iy}^i & p_{iz}^i \end{vmatrix} \tag{4.5}$$

$$\mathrm{V}_{\mathrm{jz,ix,iy,iz}} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_{jz}^i & p_{ix}^i & p_{iy}^i & p_{iz}^i \end{vmatrix} \tag{4.6}$$

$$\mathrm{V}_{\mathrm{i,jz,iy,iz}} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_i^i & p_{jz}^i & p_{iy}^i & p_{iz}^i \end{vmatrix} \tag{4.7}$$

$$\mathrm{V}_{\mathrm{i,ix,jz,iz}} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_i^i & p_{ix}^i & p_{jz}^i & p_{iz}^i \end{vmatrix} \tag{4.8}$$

$$\mathrm{V}_{\mathrm{i,ix,iy,jz}} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_i^i & p_{ix}^i & p_{iy}^i & p_{jz}^i \end{vmatrix} \tag{4.9}$$

If the global coordinates of all points in the local coordinate system Σ_i ($\mathrm{p_i, p_{ix}, p_{iy}, p_{iz}}$) are known, the global coordinates of all points in the local coordinate system Σ_j ($\mathrm{p_j, p_{jx}, p_{jy}, p_{jz}}$) can be calculated using Equations (1), (2), (3), and (4). However, if the problem is reversed and the global coordinates of all points in the local coordinate system Σ_j (p_j, $\mathrm{p_j, p_{jx}, p_{jy}, p_{jz}}$) are known, it is not always possible to calculate the global coordinates of all points in the local coordinate system Σ_i ($\mathrm{p_i, p_{ix}, p_{iy}, p_{iz}}$) using the aforementioned equations.

Therefore, in the next section, we propose the construction of symmetric linear equations. Using the global coordinates of all points in the local coordinate system Σ_j ($\mathrm{p_j, p_{jx}, p_{jy}, p_{jz}}$), we can construct barycentric coordinates to represent the global coordinates of all points in the local coordinate system Σ_i $(p_i, p_{ix}, p_{iy}, p_{iz})$.

2.2 Representing nodes i,ix,iy,iz with nodes j,jx,jy,jz

In this section, we construct symmetric linear equations. Using the global coordinates of all points in the local coordinate system Σ_j—represented as $(p_j, p_{jx}, p_{jy}, p_{jz})$—we can express the global coordinates of all points in the local coordinate system Σ_i—represented as $(p_i, p_{ix}, p_{iy}, p_{iz})$—using barycentric coordinates.

Given that the pose transformation from node i to node j is $T_{ij} \in R^{4\times4}$, the pose transformation from node j to node i is $T_{ji} = \left(T_{ij}\right)^{-1}$.

$$T_{ji} = \begin{bmatrix} R_{ji} & t_{ji} \\ 1 & 0 \end{bmatrix} \in R^{4\times4}, R_{ji} \in R^{3\times3}$$

$$t_{ji} = \left[x_{ji}, y_{ji}, z_{ji}\right]^T \in R^3$$

Thus, the positions of node i and the virtual nodes i_x, i_y, and i_z in the local coordinate system Σ_j are as follows.

$$p_i^j = t_{ji}$$

$$p_{ix}^j = R_{ji}[1,0,0]^T + t_{ji}$$

$$p_{iy}^j = R_{ji}[0,1,0]^T + t_{ji}$$

$$p_{iz}^j = R_{ji}[0,0,1]^T + t_{ji}$$

The coordinates of node j and virtual nodes j_x, j_y, and j_z in the local coordinate system Σ_j are as follows.

$$p_j^j = [0,0,0]^T$$

$$p_{jx}^j = [1,0,0]^T$$

$$p_{jy}^j = [0,1,0]^T$$

$$p_{jz}^j = [0,0,1]^T$$

Given the global coordinates of all points in the local coordinate system Σ_j $(p_j, p_{jx}, p_{jy}, p_{jz})$, we need to use barycentric coordinates to represent the global coordinates of all points in the local coordinate system Σ_i $(p_i, p_{ix}, p_{iy}, p_{iz})$. Since node j and the virtual nodes j_x, j_y, and j_z lie on different axes of the coordinate system, the volume of the tetrahedron j-jx-jy-jz is non-zero, satisfying the conditions for barycentric coordinate computation.

Since the barycentric coordinates in the node set $V_{ij} = \{i, i_x, i_y, i_z, j, j_x, j_y, j_z\}$ remain invariant under any arbitrary rotation and translation, we can calculate the barycentric coordinates using the relative positions of the nodes in the local coordinate system Σ_j (i.e., $p_i^j, p_{ix}^j, p_{iy}^j, p_{iz}^j, p_j^j, p_{jx}^j, p_{jy}^j, p_{jz}^j$). The barycentric coordinates obtained in this way remain valid for the positions of the nodes in the global coordinate system (i.e., $p_i, p_{ix}, p_{iy}, p_{iz}, p_j, p_{jx}, p_{jy}, p_{jz}$).

Next, we will construct the barycentric coordinates of node i using four nodes (j, j_x, j_y, j_z) in the local coordinate system Σ_j.

$$p_i = a_{i,j}p_j + a_{i,jx}p_{jx} + a_{i,jy}p_{jy} + a_{i,jz}p_{jz} \quad (5)$$

$$a_{i,j} = \frac{V_{i,jx,jy,jz}}{V_{j,jx,jy,jz}} \quad (5.1)$$

$$a_{i,jx} = \frac{V_{j,i,jy,jz}}{V_{j,jx,jy,jz}} \quad (5.2)$$

$$a_{i,jy} = \frac{V_{j,jx,i,jz}}{V_{j,jx,jy,jz}} \quad (5.3)$$

$$a_{i,jz} = \frac{V_{j,jx,jy,i}}{V_{j,jx,jy,jz}} \quad (5.4)$$

$$V_{j,jx,jy,jz} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_j^j & p_{jx}^j & p_{jy}^j & p_{jz}^j \end{vmatrix} \quad (5.5)$$

$$V_{i,jx,jy,jz} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_i^j & p_{jx}^j & p_{jy}^j & p_{jz}^j \end{vmatrix} \quad (5.6)$$

$$V_{j,i,jy,jz} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_j^j & p_i^j & p_{jy}^j & p_{jz}^j \end{vmatrix} \quad (5.7)$$

$$V_{j,jx,i,jz} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_j^j & p_{jx}^j & p_i^j & p_{jz}^j \end{vmatrix} \quad (5.8)$$

$$V_{j,jx,jy,i} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_j^j & p_{jx}^j & p_{jy}^j & p_i^j \end{vmatrix} \quad (5.9)$$

Then, we use the local coordinate system Σ_j and the four nodes j, j_x, j_y, and j_z to construct the barycentric coordinates of node i_x.

$$p_{ix} = a_{ix,j}p_j + a_{ix,jx}p_{jx} + a_{ix,jy}p_{jy} + a_{ix,jz}p_{jz} \quad (6)$$

$$a_{ix,j} = \frac{V_{ix,jx,jy,jz}}{V_{j,jx,jy,jz}} \quad (6.1)$$

$$a_{ix,jx} = \frac{V_{j,ix,jy,jz}}{V_{j,jx,jy,jz}} \quad (6.2)$$

$$a_{ix,jy} = \frac{V_{j,jx,ix,jz}}{V_{j,jx,jy,jz}} \quad (6.3)$$

$$a_{ix,jz} = \frac{V_{j,jx,jy,ix}}{V_{j,jx,jy,jz}} \quad (6.4)$$

$$V_{j,jx,jy,jz} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_j^j & p_{jx}^j & p_{jy}^j & p_{jz}^j \end{vmatrix} \quad (6.5)$$

$$V_{ix,jx,jy,jz} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_{ix}^j & p_{jx}^j & p_{jy}^j & p_{jz}^j \end{vmatrix} \quad (6.6)$$

$$V_{j,ix,jy,jz} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_j^j & p_{ix}^j & p_{jy}^j & p_{jz}^j \end{vmatrix} \quad (6.7)$$

$$V_{j,jx,ix,jz} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_j^j & p_{jx}^j & p_{ix}^j & p_{jz}^j \end{vmatrix} \quad (6.8)$$

$$V_{j,jx,jy,ix} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_j^j & p_{jx}^j & p_{jy}^j & p_{ix}^j \end{vmatrix} \quad (6.9)$$

Similarly, we use the local coordinate system Σ_j and the four nodes j, j_x, j_y, and j_z to construct the barycentric coordinates of node i_y.

$$p_{iy} = a_{iy,j}p_j + a_{iy,jx}p_{jx} + a_{iy,jy}p_{jy} + a_{iy,jz}p_{jz} \quad (7)$$

$$a_{iy,j} = \frac{V_{iy,jx,jy,jz}}{V_{j,jx,jy,jz}} \quad (7.1)$$

$$a_{iy,jx} = \frac{V_{j,iy,jy,jz}}{V_{j,jx,jy,jz}} \quad (7.2)$$

$$a_{iy,jy} = \frac{V_{j,jx,iy,jz}}{V_{j,jx,jy,jz}} \quad (7.3)$$

$$a_{iy,jz} = \frac{V_{j,jx,jy,iy}}{V_{j,jx,jy,jz}} \quad (7.4)$$

$$V_{j,jx,jy,jz} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_j^j & p_{jx}^j & p_{jy}^j & p_{jz}^j \end{vmatrix} \quad (7.5)$$

$$V_{iy,jx,jy,jz} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_{iy}^j & p_{jx}^j & p_{jy}^j & p_{jz}^j \end{vmatrix} \quad (7.6)$$

$$V_{j,iy,jy,jz} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_j^j & p_{iy}^j & p_{jy}^j & p_{jz}^j \end{vmatrix} \quad (7.7)$$

$$V_{j,jx,iy,jz} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_j^j & p_{jx}^j & p_{iy}^j & p_{jz}^j \end{vmatrix} \quad (7.8)$$

$$V_{j,jx,jy,iy} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_j^j & p_{jx}^j & p_{jy}^j & p_{iy}^j \end{vmatrix} \quad (7.9)$$

Finally, we use the local coordinate system Σ_j and the four nodes j, j_x, j_y, and j_z to construct the barycentric coordinates of node i_z.

$$p_{iz} = a_{iz,j}p_j + a_{iz,jx}p_{jx} + a_{iz,jy}p_{jy} + a_{iz,jz}p_{jz} \quad (8)$$

$$a_{iz,j} = \frac{V_{iz,jx,jy,jz}}{V_{j,jx,jy,jz}} \quad (8.1)$$

$$a_{iz,jx} = \frac{V_{j,iz,jy,jz}}{V_{j,jx,jy,jz}} \quad (8.2)$$

$$a_{iz,jy} = \frac{V_{j,jx,iz,jz}}{V_{j,jx,jy,jz}} \quad (8.3)$$

$$a_{iz,jz} = \frac{V_{j,jx,jy,iz}}{V_{j,jx,jy,jz}} \quad (8.4)$$

$$V_{j,jx,jy,jz} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_j^j & p_{jx}^j & p_{jy}^j & p_{jz}^j \end{vmatrix} \quad (8.5)$$

$$V_{iz,jx,jy,jz} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_{iz}^j & p_{jx}^j & p_{jy}^j & p_{jz}^j \end{vmatrix} \quad (8.6)$$

$$V_{j,iz,jy,jz} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_j^j & p_{iz}^j & p_{jy}^j & p_{jz}^j \end{vmatrix} \quad (8.7)$$

$$V_{j,jx,iz,jz} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_j^j & p_{jx}^j & p_{iz}^j & p_{jz}^j \end{vmatrix} \quad (8.8)$$

$$V_{j,jx,jy,iz} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_j^j & p_{jx}^j & p_{jy}^j & p_{iz}^j \end{vmatrix} \quad (8.9)$$

If the global coordinates of all points in the local coordinate system Σ_j (denoted as $p_j, p_{jx}, p_{jy}, p_{jz}$) are known, we can compute the global coordinates of all points in the local coordinate system Σ_i (denoted as $p_i, p_{ix}, p_{iy}, p_{iz}$) using equations (5), (6), (7), and (8).

Meanwhile, by employing equations (1) through (8), we can calculate the coordinates of nodes j, j_x, j_y, j_z from the coordinates of nodes i, i_x, i_y, i_z, and vice versa. Therefore, for any given pose transformation $T_{ij} \in E$, it can be represented using the aforementioned eight linear equations. Furthermore, these eight linear equations remain valid when the graph, formed by all nodes in the node set $V_{ij} = \{i, i_x, i_y, i_z, j, j_x, j_y, j_z\}$, undergoes scaling. Consequently, these linear equations can be approximately regarded as establishing a constraint of similarity transformation.

Let the set of virtual nodes be defined as $V_v = \{1_x, 1_y, 1_z, \ldots, n_x, n_y, n_z\}$. Denote the coordinate vector of all nodes $i \in V$ as $p = [p_1^T, p_2^T, \ldots, p_n^T]^T \in R^{3n}$. Similarly, let the coordinate vector of all virtual nodes i_x, i_y, i_z corresponding to $i \in V$ be $p_v = \left[p_{1x}^T, p_{1y}^T, p_{1z}^T, \ldots, p_{nx}^T, p_{ny}^T, p_{nz}^T\right]^T \in R^{9n}$. By combining the coordinate vectors p and p_v, we obtain the augmented coordinate vector $p_{acc} = [p^T, p_v^T]^T \in R^{12n}$. Each pose transformation $T_{ij} \in V$ can be converted into 8 linear equations. By assembling these linear equations, we can construct the following system of linear equations.

$$Ap_{acc} = 0$$

The pseudocode for 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 n nodes, and E represents the set of all pose transformations
Output: Matrix A
Define the vector of all nodes as $p = [p_1^T, p_2^T, \ldots, p_n^T]^T \in R^{3n}$.
Define the virtual node vector for all nodes as $p_v = \left[p_{1x}^T, p_{1y}^T, p_{1z}^T, \ldots, p_{nx}^T, p_{ny}^T, p_{nz}^T\right]^T \in R^{9n}$.
Combine the coordinate vector p and the virtual node vector $p_v$ to obtain $p_{acc}$, where: $p_{acc} = [p^T, p_v^T]^T \in R^{12n}$.
For $T_{ij} \in V$:
    // Computing the coordinates of nodes $i, i_x, i_y, i_z, j, j_x, j_y, j_z$ in the local coordinate system $\Sigma_i$
    Given that the pose transformation from node i to node j is represented as $T_{ij} \in R^{4\times4}$, where $T_{ij} = \begin{bmatrix} R_{ij} & t_{ij} \\ 0 & 1 \end{bmatrix} \in R^{4\times4}, R_{ij} \in R^{3\times3}, t_{ij} = \left[x_{ij}, y_{ij}, z_{ij}\right]^T \in R^3$.
    The positions of node j and its virtual nodes $j_x, j_y, j_z$ in the local coordinate system $\Sigma_i$ are given as:
    $p_j^i = t_{ij}$
    $p_{jx}^i = R_{ij}[1,0,0]^T + t_{ij}$
    $p_{jy}^i = R_{ij}[0,1,0]^T + t_{ij}$
    $p_{jz}^i = R_{ij}[0,0,1]^T + t_{ij}$
    The coordinates of node i and its virtual nodes $i_x, i_y, i_z$ in the local coordinate system $\Sigma_i$ are as follows. $p_i^i =$
```

$[0,0,0]^T, p_{ix}^i = [1,0,0]^T, p_{iy}^i = [0,1,0]^T, p_{iz}^i = [0,0,1]^T$.

// Using nodes i, i_x, i_y, and i_z to represent node j, j_x, j_y, and j_z

Using the four nodes i, i_x, i_y, and i_z in the local coordinate system Σ_i, the barycentric coordinates of node j are constructed as: $p_j = a_{j,i}p_i + a_{j,ix}p_{ix} + a_{j,iy}p_{iy} + a_{j,iz}p_{iz}$. This expression is incorporated into the linear system of equations $Ap_{acc} = 0$. The coefficients $a_{j,i}, a_{j,ix}, a_{j,iy}$, and $a_{j,iz}$ are computed using the relative coordinates $p_i^i, p_{ix}^i, p_{iy}^i, p_{iz}^i, p_j^i, p_{jx}^i, p_{jy}^i$, and p_{jz}^i, as detailed in equations (1.1)–(1.9).

Similarly, using the four nodes i, i_x, i_y, and i_z in the local coordinate system Σ_i, the barycentric coordinates of node jx are constructed as: $p_j = a_{j,i}p_i + a_{j,ix}p_{ix} + a_{j,iy}p_{iy} + a_{j,iz}p_{iz}$. This expression is also incorporated into the linear system of equations $Ap_{acc} = 0$. The coefficients $a_{j,i}, a_{j,ix}, a_{j,iy}$, and $a_{j,iz}$ are calculated using the relative coordinates $p_i^i, p_{ix}^i, p_{iy}^i, p_{iz}^i, p_j^i, p_{jx}^i, p_{jy}^i$, and p_{jz}^i, as detailed in equations (2.1)–(2.9).

Likewise, using the four nodes i, i_x, i_y, and i_z in the local coordinate system Σ_i, the barycentric coordinates of node j_y are constructed as: $p_{jy} = a_{jy,i}p_i + a_{jy,ix}p_{ix} + a_{jy,iy}p_{iy} + a_{jy,iz}p_{iz}$. This expression is incorporated into the linear system of equations $Ap_{acc} = 0$. The coefficients $a_{jy,i}, a_{jy,ix}, a_{jy,iy}$, and $a_{jy,iz}$ are calculated using the relative coordinates $p_i^i, p_{ix}^i, p_{iy}^i, p_{iz}^i, p_j^i, p_{jx}^i, p_{jy}^i$, and p_{jz}^i, as detailed in equations (3.1)–(3.9).

Finally, using the four nodes i, i_x, i_y, and i_z in the local coordinate system Σ_i, the barycentric coordinates of node j_z are constructed as: $p_{jz} = a_{jz,i}p_i + a_{jz,ix}p_{ix} + a_{jz,iy}p_{iy} + a_{jz,iz}p_{iz}$. This expression is incorporated into the linear system of equations $Ap_{acc} = 0$. The coefficients $a_{jz,i}, a_{jz,ix}, a_{jz,iy}$, and $a_{jz,iz}$ are calculated using the relative coordinates $p_i^i, p_{ix}^i, p_{iy}^i, p_{iz}^i, p_j^i, p_{jx}^i, p_{jy}^i$, and p_{jz}^i, as detailed in equations (4.1)–(4.9).

// Calculating the coordinates of nodes $i, i_x, i_y, i_z, j, j_x, j_y$, and j_z in the local coordinate system Σ_j

Given that the pose transformation from node j to node i is denoted as $T_{ji} \in R^{4\times4}$, where: $T_{ji} = \begin{bmatrix} R_{ji} & t_{ji} \\ 0 & 1 \end{bmatrix} \in R^{4\times4}, R_{ji} \in R^{3\times3}, t_{ji} = [x_{ji}, y_{ji}, z_{ji}]^T \in R^3$.

The positions of node i and its associated virtual nodes i_x, i_y, and i_z in the local coordinate system Σ_j are as follows.

$$p_i^j = t_{ji}$$

$$p_{ix}^j = R_{ji}[1,0,0]^T + t_{ji}$$

$$p_{iy}^j = R_{ji}[0,1,0]^T + t_{ji}$$

$$p_{iz}^j = R_{ji}[0,0,1]^T + t_{ji}$$

The coordinates of node j and its virtual nodes j_x, j_y, and j_z in the local coordinate system Σ_j are defined as: $p_j^j = [0,0,0]^T, p_{jx}^j = [1,0,0]^T, p_{jy}^j = [0,1,0]^T, p_{jz}^j = [0,0,1]^T$.

// Using nodes j, j_x, j_y, and j_z to represent node i, i_x, i_y, and i_z

Using the four nodes j, j_x, j_y, and j_z in the local coordinate system Σ_j, the barycentric coordinates of node i are constructed as follows: $p_i = a_{i,j}p_j + a_{i,jx}p_{jx} + a_{i,jy}p_{jy} + a_{i,jz}p_{jz}$. This relationship is incorporated into the linear system of equations $Ap_{acc} = 0$. The coefficients $a_{i,j}, a_{i,jx}, a_{i,jy}, a_{i,jz}$ are calculated using the relative coordinates $p_i^j, p_{ix}^j, p_{iy}^j, p_{iz}^j, p_j^j, p_{jx}^j, p_{jy}^j, p_{jz}^j$, as defined by Equations (5.1)–(5.9).

Similarly, the barycentric coordinates of node i_x are constructed in the local coordinate system Σ_j using the four nodes j, j_x, j_y, j_z, as follows: $p_{ix} = a_{ix,j}p_j + a_{ix,jx}p_{jx} + a_{ix,jy}p_{jy} + a_{ix,jz}p_{jz}$. This relationship is also incorporated into the linear system $Ap_{acc} = 0$. The coefficients $a_{ix,j}, a_{ix,jx}, a_{ix,jy}, a_{ix,jz}$ are computed based on the relative coordinates $p_i^j, p_{ix}^j, p_{iy}^j, p_{iz}^j, p_j^j, p_{jx}^j, p_{jy}^j, p_{jz}^j$, as defined by Equations (6.1)–(6.9).

The barycentric coordinates of node i_y are constructed in the local coordinate system Σ_j using the four nodes j, j_x, j_y, j_z, as follows: $p_{iy} = a_{iy,j}p_j + a_{iy,jx}p_{jx} + a_{iy,jy}p_{jy} + a_{iy,jz}p_{jz}$. This relationship is included in the linear system $Ap_{acc} = 0$. The coefficients $a_{iy,j}, a_{iy,jx}, a_{iy,jy}, a_{iy,jz}$ are determined using the relative coordinates $p_i^j, p_{ix}^j, p_{iy}^j, p_{iz}^j, p_j^j, p_{jx}^j, p_{jy}^j, p_{jz}^j$, as shown in Equations (7.1)–(7.9).

The barycentric coordinates of node i_z are constructed in the local coordinate system Σ_j using the four nodes j, j_x, j_y, j_z, as follows: $p_{iz} = a_{iz,j}p_j + a_{iz,jx}p_{jx} + a_{iz,jy}p_{jy} + a_{iz,jz}p_{jz}$. This relationship is also added to the linear system $Ap_{acc} = 0$. The coefficients $a_{iz,j}, a_{iz,jx}, a_{iz,jy}, a_{iz,jz}$ are computed from the relative coordinates $p_i^j, p_{ix}^j, p_{iy}^j, p_{iz}^j, p_j^j, p_{jx}^j, p_{jy}^j, p_{jz}^j$, as detailed in Equations (8.1)–(8.9).

4. Pose 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 encoders, and loop closure detection. The pose transformations obtained from each type of sensor will be discussed in detail below.

4.1 Non-GPS sensors

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

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

In the case of loop closure detection, it is also possible to obtain the pose transformation T_{ij} between two nodes (node i and node j). Typically, node i and node j are two adjacent nodes identified when the robot completes a loop and returns to its starting point. While node i and node j are temporally far apart in the robot's movement timeline, they are spatially close in distance.

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

3.2 GPS sensors

The case of GPS sensors is relatively unique. The data obtained by GPS sensors consists of latitude and longitude values. Using specific formulas, these latitude and longitude coordinates can be converted into measurable distance values. This means that the position detected by the 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 LiDAR) can only acquire the relative pose transformation T_{ij} between two specific nodes (node i and node j). If the pose transformation between two distant nodes (e.g., node i and node $i + n$) is required, it must be obtained through successive accumulation: $T_{i,i+n} = T_{i,i+1}T_{i+1,i+2}\ldots T_{i+n-1,i+n}$. However, the error in the accumulated pose transformation $T_{i,i+n}$ will inevitably increase over time. For GPS sensors, since they operate within a fixed latitude-longitude coordinate system, the translation $t_{i,i+n}$ between two distant nodes (e.g., node i and node $i + n$) can still maintain high accuracy. Therefore, GPS sensor data plays a crucial role in preserving the overall shape accuracy of large-scale maps.

Another unique characteristic of GPS sensors is that GPS data provides only translational information, without any rotational information. Suppose we have the GPS data for node i (its position in the latitude-longitude coordinate system) and the GPS data for node j. In this case, we can only determine the translation $t_{ij} \in R^2$ from node iii to node jjj in the latitude-longitude coordinate system. However, no rotational information is available. As a result, the translation vector $t_{ij} \in R^2$ obtained from the GPS sensor cannot be combined with the pose transformation $T_{ij} \in R^{4\times4}$ obtained from other sensors to construct linear equation constraints. While it is possible to construct non-linear pose graph constraints, this approach differs from our goal of building an optimization algorithm based on linear equations.

The data obtained from GPS sensors typically includes only latitude and longitude, without altitude information. Even when altitude data is available, its accuracy is significantly lower compared to the latitude and longitude data. If the data from the GPS sensor contains only latitude and longitude, the planar translation $t_{ij} \in R^2$ between two nodes (node i and node j) can be obtained. In this case, we can select three nodes (i, j, k) to construct a system of linear equations based on planar triangles. Conversely, if the data obtained from the GPS sensor includes latitude, longitude, and altitude, the planar translation $t_{ij} \in R^3$ between two nodes (node i and node j) can be obtained. Similarly, three nodes (i, j, k) can be selected to construct a system of linear equations based on spatial triangles.

For processing GPS sensor data, we adopt the method proposed in [1], which uses constraints from similar triangles to construct the system of linear equations. To avoid unnecessary verbosity in the description, we directly present the system of linear equations built using GPS data as follows. The detailed process of constructing the linear equations can be found in [1].

$$Cp_{acc} = 0$$

These equations include those constructed using adjacent GPS data and those constructed using cross-node GPS data.

3.3 Variance of pose transformation

The relative pose transformation between two frames can be obtained using cameras, LiDAR, IMU sensors, GPS sensors, wheel odometers, and loop closure detection. Typically, the pose transformations derived from different sensors exhibit varying levels of variance. We can preset the variance for each type of sensor based on its intrinsic characteristics or determine the variance of the pose transformation T_{ij} during the computation process in the front-end module of the SLAM algorithm. The pose transformation matrix T_{ij} is a 4×4 matrix. For simplicity, we use a single variance σ_{ij}^2 to represent the overall variance of the entire pose transformation matrix.

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

$$Ap_{acc} = 0 + v_1$$

$$Cp_{acc} = 0 + v_2$$

The error vectors $v_1 \sim N(0, \Sigma_1), v_2 \sim N(0, \Sigma_2)$. If we ignore the correlations between different linear equations, the covariance matrix Σ_1, Σ_2 becomes a diagonal matrix.

For the variance Σ_1 of the error vector v_1, we use the variance σ_{ij}^2 of the corresponding pose transformation T_{ij} in the linear equation as the variance of that linear equation. For the error vector v_2, each linear equation is constructed using two pose transformations (e.g., T_{ij} and T_{ik}). For simplicity, we define the variance of the corresponding linear equation as $\sigma^2 = \sigma_{ij}^2 + \sigma_{ik}^2$.

5. Map scale calculation

5.1 Without using GPS sensors

In the graph $G = [V, E]$, V represents the set of all nodes (keyframes), and suppose the set V contains n nodes. The coordinate vector of all nodes is denoted as $p = [p_1^T, \ldots, p_n^T]^T \in R^{3n}$. For each node $i \in V$, three virtual nodes i_x, i_y, i_z are defined in its local coordinate system Σ_i. The set of virtual nodes is $V_v = \{i_x, i_y, i_z, i \in V\}$, and the coordinate vector of the virtual nodes is $p_v = \left[p_{1x}^T, p_{1y}^T, p_{1z}^T, \ldots, p_{nx}^T, p_{ny}^T, p_{nz}^T\right] \in R^{9n}$. By combining the coordinate vectors p and p_v, we obtain $p_{acc} = [p^T, p_v^T]^T \in R^{12n}$.

The set of all pose transformations T_{ij} is given 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 eight linear equations. We define the system of linear equations corresponding to all pose transformations $T_{ij} \in E$ as follows.

$$Ap_{acc} = 0$$

We define the local coordinate system Σ_1 of the first node as the global coordinate system Σ_g, and calculate the coordinates of other nodes. If we use another coordinate system Σ_i as the global coordinate system Σ_g, then the corresponding pose transformation T_{1i} can be used to transform all coordinates into the coordinate system Σ_i. When the local coordinate system Σ_1 is set as the global coordinate system Σ_g, the coordinates of node 1 are $p_1 = [0,0,0]^T$, the coordinates of the virtual node 1_x are $p_{1x} = [1,0,0]^T$, the coordinates of the virtual node 1_y are $p_{1y} = [0,1,0]^T$, and the coordinates of the virtual node 1_z are $p_{1z} = [0,0,1]^T$. According to the method for constructing the linear equation system $Ap_{acc} = 0$, as long as the coordinates of any local coordinate system for one node are determined (e.g., $p_1, p_{1x}, p_{1y}, p_{1z}$), the coordinates of all nodes p_{acc} can be computed. The linear equation system $Ap_{acc} = 0$ constrains the relationships between the local coordinate systems of different nodes, but the scale remains adjustable.

If the coordinates of other nodes in p_{acc} are calculated using the coordinates of p_1, p_{1x}, p_{1y}, and p_{1z}, it essentially means that the scale of the graph is determined by these coordinates. When this scale information propagates outward from node 1 to other nodes, the error will gradually accumulate. For instance, if we define the unit vector magnitudes of the three axes in the local coordinate system Σ_1 as 1 (i.e., $\rho_{i,ix} = 1, \rho_{i,iy} = 1, \rho_{i,iz} = 1$), the magnitudes of the vectors in the local coordinate system Σ_n (formed by node n and its associated nodes n_x, n_y, and n_z) may not necessarily remain equal to 1 ($\rho_{n,nx}, \rho_{n,ny}, \rho_{n,nz}$) after calculating the coordinates of node n. 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 $p_1 = [0,0,0]^T$, the coordinates of the virtual node 1_x as $p_{1x} = [\rho, 0,0]^T$, the coordinates of the virtual node 1_y as $p_{1y} = [0, \rho, 0]^T$, and the coordinates of the virtual node 1_z as $p_{1z} = [0,0, \rho]^T$. Let p_{acc} represent the coordinate vector, where the coordinates of p_1, p_{1x}, p_{1y}, and p_{1z} are removed, resulting in the sub-coordinate vector $p_{sub} \in R^{12n-12}$. By substituting the coordinates of p_1, p_{1x}, p_{1y}, and p_{1z} into the equations, the linear system $Ap_{acc} = 0 + v_1$ is transformed into the following.

$$A_{sub} p_{sub} = b_{sub} + v_{sub1}$$

Here, $v_{sub1} \sim N(0, \Sigma_{sub1})$ represents the noise corresponding to each linear equation. The variance of each linear equation is represented by the variance σ_{ij}^2 associated with the corresponding pose transformation T_{ij}. The number of linear equations in the system remains unchanged, and the variance associated with each linear equation also remains unchanged. Thus, the error vector satisfies $v_{sub1} = v_1$, and the covariance matrix satisfies $\Sigma_{sub1} = \Sigma_1$. The least-squares solution of this linear system is given as follows.

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

The coordinate vector p_{sub} represents the coordinates from node 2 to node n, where each coordinate (e.g., $p_i, p_{ix}, p_{iy}, p_{iz}$) consists of components that are linear functions of the unknown variable ρ.

$$p_i = [a_i\rho + b_i, c_i\rho + d_i, e_i\rho + f_i]^T$$

$$p_{ix} = [a_{ix}\rho + b_{ix}, c_{ix}\rho + d_{ix}, e_{ix}\rho + f_{ix}]^T$$

$$p_{iy} = \left[a_{iy}\rho + b_{iy}, c_{iy}\rho + d_{iy}, e_{iy}\rho + f_{iy}\right]^T$$

$$p_{iz} = [a_{iz}\rho + b_{iz}, c_{iz}\rho + d_{iz}, e_{iz}\rho + f_{iz}]^T$$

Similarly, the coordinates of node 1 and the virtual nodes $1_x, 1_y$, and 1_z are included in the coordinate vector p_{acc}, where each component is also a linear function of the unknown variable ρ. We then 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 its virtual node i_x should be 1, the distance from node i to i_y should be 1, and the distance from node i to i_z should also be 1.

$$\begin{aligned} J_1 &= \sum_{i\in V}\left(\left(\left\|p_i - p_{ix}\right\|^2 - 1\right)^2 + \left(\left\|p_i - p_{iy}\right\|^2 - 1\right)^2 + \left(\left\|p_i - p_{iz}\right\|^2 - 1\right)^2\right) \\ &= \sum_{i\in V} + \left(\left((a_i - a_{iz})\rho + (b_i - b_{iz})\right)^2 + \left((c_i - c_{iz})\rho + (d_i - d_{iz})\right)^2 + \left((e_i - e_{iz})\rho + (f_i - f_{iz})\right)^2 - 1\right)^2 + \\ &\quad \left(\left((a_i - a_{ix})\rho + (b_i - b_{ix})\right)^2 + \left((c_i - c_{ix})\rho + (d_i - d_{ix})\right)^2 + \left((e_i - e_{ix})\rho + (f_i - f_{ix})\right)^2 - 1\right)^2 + \\ &\quad \left(\left((a_i - a_{iy})\rho + (b_i - b_{iy})\right)^2 + \left((c_i - c_{iy})\rho + (d_i - d_{iy})\right)^2 + \left((e_i - e_{iy})\rho + (f_i - f_{iy})\right)^2 - 1\right)^2 \\ &= A_1\rho^4 + B_1\rho^3 + C_1\rho^2 + D_1\rho + E_1 \end{aligned}$$

The error function J_2 is defined as follows: If there exists a pose transformation $T_{ij} \in R^{4\times4}$ between node i and node j, where the translational component is $t_{ij} \in R^3$, then the distance between node i and node j can be expressed as $\rho_{ij} = \left|\left|t_{ij}\right|\right|$. Meanwhile, the magnitude can also be calculated using the solved coordinates p_j of node i and p_j of node j (a linear function of the unknown ρ). The difference between these two magnitudes is used to define the error function.

$$J_2 = \sum\nolimits_{T_{ij}\in E}\left(\left|\left|p_i - p_j\right|\right|^2 - \rho_{ij}^2\right)^2$$

$$= \sum\nolimits_{T_{ij}\in E}\left(\left((a_i - a_j)\rho + (b_i - b_j)\right)^2\right.$$
$$+ \left((c_i - c_j)\rho + (d_i - d_j)\right)^2 + \left((e_i - e_j)\rho + (f_i - f_j)\right)^2$$
$$\left. - \rho_{ij}^2\right)^2$$

$$= A_2\rho^4 + B_2\rho^3 + C_2\rho^2 + D_2\rho + E_2$$

By summing the error function J_1 and the error function J_2, we obtain the overall 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, denoted as 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 the function J is used to evaluate the remaining roots.

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

If the second derivative at a root ρ_i is positive, ρ_i corresponds to the minimum of the function J. Conversely, if the second derivative at ρ_i is negative, ρ_i corresponds to a local maximum of the function J. Once the optimized map scale ρ is determined, it can be substituted into the coordinates $p_{acc} \in R^{12n}$. The pseudocode for computing the map scale and node coordinates is as follows.

Process 2: Calculation of map scale and node coordinates (without using GPS data)

Input: Linear system of equations $Ap_{acc} = 0$, variance matrix Σ_1

Output: Coordinate vector $p_{acc} \in R^{12n}$ and map scale ρ

Let the coordinate vector of node i ∈ V be defined as: $p = [p_1^T, p_2^T, \dots, p_n^T]^T \in R^{3n}$.

Let the coordinate vector of virtual nodes be defined as: $p_v = \left[p_{1x}^T, p_{1y}^T, p_{1z}^T, \dots, p_{nx}^T, p_{ny}^T, p_{nz}^T\right] \in R^{9n}$.

The combined coordinate vector of nodes p and virtual nodes p_v is then: $p_{acc} = [p^T, p_v^T]^T \in R^{12n}$.

Assume the coordinates of node 1 are fixed as $p_1 = [0,0,0]^T$, and the coordinates of its associated virtual nodes are defined as: $p_{1x} = [\rho, 0,0]^T$, $p_{1y} = [0, \rho, 0]^T$, $p_{1z} = [0,0, \rho]^T$.

Let $p_{sub} \in R^{12n-12}$ denote the coordinate vector p_{acc} with the coordinates of p_1, p_{1x}, p_{1y}, and p_{1z} removed.

Substituting the coordinates of p_1, p_{1x}, p_{1y}, and p_{1z} into the system of equations, the linear system $Ap_{acc} = 0$ reduces to $A_{sub}p_{sub} = b_{sub}$.

The planar components of the virtual nodes, p_{sub}, can then be computed by solving the linear system $A_{sub}p_{sub} = b_{sub} + v_{sub1}$. The least-squares solution to this system is given by $p_{sub} = \left(A_{sub}^T\Sigma_1^{-1}A_{sub}\right)^{-1}A_{sub}^T\Sigma_1^{-1}b_{sub}$. (This solution is a linear function of ρ)

Define the error function J_1 as $J1 = \sum_{i\in V}\left(\left(\left|\left|p_1 - p_{1x}\right|\right|^2 - 1\right)^2 + \left(\left|\left|p_1 - p_{1y}\right|\right|^2 - 1\right)^2 + \left(\left|\left|p_1 - p_{1z}\right|\right|^2 - 1\right)^2\right) = A_1\rho^4 + B_1\rho^3 + C_1\rho^2 + D_1\rho + E_1$.

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

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

To compute the parameter ρ that minimizes the error function, calculate 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 the three, and select the root ρ_i that corresponds to the minimum value of J by evaluating the second derivative of the function J.

Substitute the parameter ρ into the coordinate vector p_{acc}.

5.2 Using GPS sensors

When not using GPS sensors, the linear equations between each node i ∈ V and the virtual nodes i ∈ V are as follows.

$$Ap_{acc} = 0 + v_1 \tag{9}$$

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

$$Cp_{acc} = 0 + v_2 \tag{10}$$

Subsequently, equations (9) and (10) are combined as follows.

$$Ep_{acc} = 0 + v_3$$

The error vector is defined as $v_3 = [v_1^T, v_2^T]^T$, where $v_3 \sim N(0, \Sigma_3)$. If we ignore the correlations between different linear equations, the covariance matrix Σ_3 becomes a diagonal matrix. Furthermore, it satisfies the following relationship: $\Sigma_3 = \mathrm{diag}\{\Sigma_1, \Sigma_2\}$.

We define the following coordinates: $p_1 = [0,0,0]^T, p_{1x} = [\rho, 0,0]^T, p_{1y} = [0, \rho, 0]^T, p_{1z} = [0,0, \rho]^T$. Substituting these coordinates into the equation $Ep_{acc} = 0 + v_3$, the equation transforms into the following form.

$$E_{sub}p_{sub} = F_{sub} + v_{sub3}$$

The coordinate vector $p_{sub} \in R^{12n-12}$ is the vector p_{acc} with components p_1, p_{1x}, p_{1y}, and p_{1z} removed. The error vector $v_{sub3} \sim N(0, \Sigma_{sub3})$ 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 $v_{sub3} = v_3$, and the covariance matrix $\Sigma_{sub3} = \Sigma_3$. The least squares solution of this linear system is as follows.

$$p_{sub} = \left(E_{sub}^T \Sigma_3^{-1} E_{sub}\right)^{-1} E_{sub}^T \Sigma_3^{-1} F_{sub}$$

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

Process 3: Calculation of map scale and node coordinates (using GPS data)

```
Input: Linear equations $Ap_{acc} = 0, Cp_{acc} = 0$, variance matrices $\Sigma_1, \Sigma_2$

Output: Coordinate vector $p_{acc} \in R^{12n}$, map scale $\rho$

Define the coordinate vector of node $i \in V$ as $p = [p_1^T, p_2^T, \dots, p_n^T]^T \in R^{3n}$.

Define the virtual node coordinate vector as $p_v = \left[p_{1x}^T, p_{1y}^T, p_{1z}^T, \dots, p_{nx}^T, p_{ny}^T, p_{nz}^T\right] \in R^{9n}$.

Combine the coordinate vectors p and $p_v$ to obtain $p_{acc} = [p^T, p_v^T]^T \in R^{12n}$.

Assume the coordinates of node 1 are $p_1 = [0,0,0]^T$, the coordinates of the virtual node $1_x$ are $p_{1x} = [\rho, 0,0]^T$, the coordinates of the virtual node $1_y$ are $p_{1y} = [0, \rho, 0]^T$, and the coordinates of the virtual node $1_z$ are $p_{1z} = [0,0, \rho]^T$.

By removing $p_1, p_{1x}, p_{1y}$, and $p_{1z}$ from the coordinate vector $p_{acc}$, derive the reduced coordinate vector $p_{sub} \in R^{12n-12}$.

Combine the linear equations $Ap_{acc} = 0$ and $Cp_{acc} = 0$ into linear equations $Ep_{acc} = 0$.

Substituting the coordinates of $p_1, p_{1x}, p_{1y}$, and $p_{1z}$ into the equation, the linear equation system $Ep_{acc} = 0$ becomes $E_{sub}p_{sub} = F_{sub}$.

The coordinate vector $p_{sub}$ is calculated using the linear equation system $E_{sub}p_{sub} = F_{sub} + v_{sub3}$. The least-squares solution of this linear equation system is given by: $p_{sub} = \left(E_{sub}^T \Sigma_3^{-1} E_{sub}\right)^{-1} E_{sub}^T \Sigma_3^{-1} F_{sub}$. (This solution is a linear function of $\rho$)

Define the error function $J_1$ as follows: $J1 = \sum_{i \in V}\left(\left(||p_1 - p_{1x}||^2 - 1\right)^2 + \left(||p_1 - p_{1y}||^2 - 1\right)^2 + \left(||p_1 - p_{1z}||^2 - 1\right)^2\right) = A_1\rho^4 + B_1\rho^3 + C_1\rho^2 + D_1\rho + E_1$.

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

By summing the two error functions, 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 determine the parameter rho that minimizes the error function, we compute the derivative of J with respect to $\rho$: $\frac{dJ}{d\rho} = 4A\rho^3 + 3B\rho^2 + 2C\rho + D = 0$.

Using the cubic equation formula, we compute the three roots $\rho_1, \rho_2, \rho_3$. Among these roots, we discard any complex roots and use the second derivative of J to select the root $\rho_i$ that corresponds to the minimum value of the error function.

The parameter $\rho$ is substituted into the coordinate vector $p_{acc}$.
```

6. Outlier removal algorithm

In this paper, the linear equations are constructed using the pose transformations T_{ij} between nodes. If there are outliers in the set of pose transformations $E = \{T_{ij}, i, j \in V\}$ within the graph $G = [V, E]$, 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 a residual term. The presence of outliers can cause significant deviations in the solution of the equations.

In nonlinear optimization-based SLAM approaches, each pose transformation T_{ij} is used to construct a residual function $h(T_{ij})$ through an observation function. The residual terms are then formed as $e(T_{ij}) = h\left(T_{ij}\right)^2$, and the overall cost function JJJ is obtained by summing these residual terms. However, it is common practice to apply a robust function to the residual 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 grows much slower beyond a certain range. After introducing the robust function, the error function J is expressed as follows.

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

Due to the presence of the robust function, even if a specific pose transformation T_{ij} is classified as an outlier, the corresponding error term will not be excessively large. Consequently, outliers have a limited impact on the minimization of the overall error function. Furthermore, the original error term, $e(T_{ij}) = h\left(T_{ij}\right)^2$, is a nonlinear function. After incorporating the robust function, the new error term,

$\rho\left(h(T_{ij})^2\right)$, remains a nonlinear function. Therefore, we still solve for the minimum of the error function using numerical optimization methods.

However, for the linear equations discussed in this paper, the robust function approach cannot be applied. If a robust function is added to the error term of each linear equation, the resulting function would become nonlinear. The purpose of formulating pose transformations as linear equations is to reduce computational complexity; if nonlinear equations were used instead, this objective could not be achieved.

Paper [2] presents a method for removing outliers in pose transformations. Since the method involves numerous detailed steps, we provide a brief overview of its 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 a set of pose transformations $E' = \{T_{ij}, i, j \in V\}$ with outliers removed. By constructing a system of linear equations using all pose transformations in E' E'E' , the influence of outliers on the equations can be effectively avoided.

This algorithm requires assigning a probability p_{ij} to each pose transformation T_{ij} in the input set $E = \{T_{ij}, i, j \in V\}$, representing the likelihood of it being an outlier. It is not necessary for this probability to be an exact value; an approximate estimation can suffice. For instance, the variance σ_{ij}^2 corresponding to the pose transformation T_{ij} can be used as a rough measure. A potential scheme for determining this is as follows.

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

The probability p_{ij} is used in the outlier removal algorithm to find a route 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 consisting of edges with fewer outliers to achieve the transformation from node i to node j.

The general steps of the outlier removal algorithm are as follows: Suppose there exists a graph $G = [V, E]$ containing two nodes $p_1, p_2 \in V$. There are three rotation and translation matrices $T_{12}^1, T_{12}^2, T_{12}^3$ representing the transformations between nodes p_1 and p_2.

At the same time, nodes p_1 and p_2 can also be connected 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 another pair of 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 then 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 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 (set E) contain no outliers, then these pose transformations T_{12}^i should be very similar to one another. However, if any of the pose transformations used is an outlier, the corresponding pose transformation T_{12}^i will differ significantly from the others. We can decompose the pose transformation T_{12}^i into a translation vector and a rotation vector. Each component of these vectors is then processed using a statistical outlier removal method based on the Interquartile Range (IQR) to eliminate outliers.

7. Calculation of local coordinate system

The method for calculating the local coordinate system has already been presented in [1]. However, for the sake of completeness in this SLAM framework, we will explicitly describe the method for calculating the local coordinate system here. We have already determined the coordinates of each node in the graph $G = [V, E]$ within the global coordinate system Σ_g, denoted as $p = [p_1^T, p_2^T, \ldots, p_n^T] \in R^{3n}$. Let the global coordinate of node i be $p_i = [x_i, y_i, z_i]^T$, and let the neighboring nodes of node i be denoted as $j \in N_i$, where the global coordinate of node j in the global coordinate system Σ_g is $p_j = \left[x_j, y_j, z_j\right]^T$. The relative coordinate of node j with respect to node i is given as $p_j^r = \left[x_j - x_i, y_j - y_i, z_j - z_i\right]^T$. The relative coordinate vectors of all neighboring nodes in the global coordinate system can then be represented as $P_i^n = \left[p_{j1}^r, \ldots, p_{jn}^r\right] \in R^{3\times|N_i|}$, 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 $T_{ij} = \begin{bmatrix} R_{ij} & t_{ij} \\ 0 & 1 \end{bmatrix} \in R^{4\times4}$, where the translation vector is $t_{ij} = \left[x_{ij}, y_{ij}, z_{ij}\right]^T, j \in N_i$. Then, the relative position vectors of the neighboring nodes $j \in N_i$ in the local coordinate system Σ_i are given as $Q_i^n = \left[t_{ij1}, \ldots, t_{ijn}\right] = \left[q_{ij1}, \ldots, q_{ijn}\right] \in R^{3\times|N_i|}$, where $j_1, \ldots, j_n \in N_i$.

We have computed the coordinates of the virtual nodes $i_x, i_y, i_z \in V_v$ corresponding to each node $i \in V$ in the global coordinate system Σ_g. The coordinates of these virtual nodes in the global coordinate system are expressed as $p_{ix} = [x_{ix}, y_{ix}, z_{ix}]^T, p_{iy} = \left[x_{iy}, y_{iy}, z_{iy}\right]^T$, and $p_{iz} = [x_{iz}, y_{iz}, z_{iz}]^T$, respectively. The global coordinate of node i is represented as $p_i = [x_i, y_i, z_i]^T$. The relative coordinates of the virtual nodes i_x, i_y, and i_z with respect to node i are as follows.

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

$$p_{iy}^r = \left[x_{iy} - x_i, y_{iy} - y_i, z_{iy} - z_i\right]^T$$

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

The relative coordinate vectors of virtual nodes i_x, i_y, and i_z in the global coordinate system are denoted as $P_i^v =$

$[p^r_{ix}, p^r_{iy}, p^r_{iz}]$. The coordinates of the virtual nodes i_x, i_y, and i_z in the local coordinate system Σ_i are $p^i_{ix} = [1,0,0]^T, p^i_{iy} = [0,1,0]^T$, and $p^i_{iz} = [0,0,1]^T$, respectively. Let the coordinate vectors of the virtual nodes i_x, i_y, and i_z in the local coordinate system Σ_i be represented as $Q^v_i = [p^i_{ix}, p^i_{iy}, p^i_{iz}]$. The coordinate vectors P^n_i and P^v_i are concatenated to form $P_i = [P^n_i, P^v_i] \in R^{3\times|N_i|+3}$; similarly, Q^n_i and Q^v_i are concatenated to form $Q_i = [Q^n_i, Q^v_i] \in R^{3|N_i|+3}$. Using these, the rotation matrix R_i of the local coordinate system at node i can be computed via point set registration methods.

Here, the coordinate vectors P_i and Q_i are constructed using the neighboring nodes $j \in N_i$ and the virtual nodes i_x, i_y, i_z. Although we could also include the virtual coordinates of the neighboring nodes j_x, j_y, j_z for $j \in N_i$ in the coordinate vectors, we opted not to do so for the following reasons: First, as long as three nodes (e.g., the virtual nodes i_x, i_y, i_z) are present, the rotation matrix R_i can be determined. Second, the information of the neighboring nodes $j \in N_i$ already sufficiently represents their positions. Including the virtual coordinates j_x, j_y, j_z for $j \in N_i$ may unnecessarily increase computational complexity.

Given a set of points with position vectors $P_i \in R^{3|N_i|+3}$ in the global coordinate system Σ_g, and their corresponding position vectors $Q_i \in R^{3|N_i|+3}$ in the local coordinate system Σ_i, we aim to find a rotation matrix R_i by constructing the following error function.

$$E(R) = \sum\nolimits_{j\in N_i} \left| \left| R_i p^r_{ij} - q_{ij} \right| \right|^2$$

We need to determine the rotation matrix $R_i \in R^{3\times3}$ that minimizes the error function $E(R_i)$. This problem falls under the category of point cloud registration. To compute the matrix R_i, we can employ the Singular Value Decomposition (SVD) method as part of the Iterative Closest Point (ICP) algorithm. The matrix R_i represents the rotation that transforms 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 given by $(R_i)^{-1}$. Using the SVD method within the ICP algorithm, the rotation matrix R_i that minimizes the error function $E(R)$ is determined as follows.

$$\begin{aligned} H &= P_i Q_i^T \\ &= U\Sigma V^T \\ R_i &= VU^T \end{aligned}$$

The singular value decomposition (SVD) of the matrix $H \in R^{3\times3}$ is computed, yielding the orthogonal matrices U and V. The rotation matrix is then defined as $R_i = VR^T$. However, if the point sets P_i or Q_i are degenerate, the resulting matrix R_i may become a reflection matrix ($\det(R_i) = -1$). To address this, the reflection matrix R_i can be adjusted to a proper rotation matrix using the following steps.

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

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

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

Process 4: Calculation of local coordinate system

Input: The coordinates of node i: $p_i \in R^3$; neighboring nodes $j \in N_i$ of node i; the coordinates of neighboring nodes $j \in N_i$: $p_j \in R^3$; the relative transformation of neighboring nodes $j \in N_i$: T_{ij}; the coordinates of virtual nodes i_x, i_y, i_z: $p_{ix}, p_{iy}, p_{iz} \in R^3$

Output: The rotation matrix of the local coordinate system Σ_i relative to the global coordinate system Σ_g, denoted as $(R_i)^{-1}$

The relative coordinates of neighboring node $j \in N_i$ with respect to node i are given by: $p^r_j = \left[x_j - x_i, y_j - y_i, z_j - z_i\right]^T$.

The relative coordinate vectors of all neighboring nodes $j_1, \ldots, j_n \in N_i$ in the global coordinate system Σ_g form the matrix: $P^n_i = \left[p^r_{j1}, \ldots, p^r_{jn}\right] \in R^{3\times|N_i|}$.

In the local coordinate system Σ_i, the coordinates of these neighboring nodes are represented as: $Q^n_i = \left[t_{ij1}, \ldots, t_{ijn}\right] = \left[q_{ij1}, \ldots, q_{ijn}\right] \in R^{3\times|N_i|}, j_1, \ldots, j_n \in N_i$.

The relative coordinates of virtual nodes i_x, i_y, i_z with respect to node i are: $p^r_{ix} = [x_{ix} - x_i, y_{ix} - y_i, z_{ix} - z_i]^T, p_{iy} = \left[x_{iy} - x_i, y_{iy} - y_i, z_{iy} - z_i\right]^T, p_{iz} = [x_{iz} - x_i, y_{iz} - y_i, z_{iz} - z_i]^T$. These vectors are combined into the matrix: $P^v_i = \left[p^r_{ix}, p^r_{iy}, p^r_{iz}\right] \in R^{3\times3}$.

In the local coordinate system Σ_i, the coordinates of virtual nodes i_x, i_y, i_z are defined as: $p^i_{ix} = [1,0,0]^T, p^i_{iy} = [0,1,0]^T, p^i_{iz} = [0,0,1]^T$. These vectors form the matrix: $Q^v_i = \left[p^i_{ix}, p^i_{iy}, p^i_{iz}\right] \in R^{3\times3}$.

The coordinate matrices for neighboring nodes and virtual nodes are combined as follows: $P_i = [P^n_i, P^v_i] \in R^{3\times(3n+3)}$, $Q_i = [Q^n_i, Q^v_i] \in R^{3\times(3n+3)}$.

To compute the rotation matrix R_i, the matrix H is defined as: $H = P_i Q_i^T$.

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

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

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

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