A linear SLAM backend optimization algorithm based on virtual coordinate system and barycentric coordinates 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. The barycentric coordinates in 2D spaces

The barycentric coordinates are a geometric concept used to describe the position of a point relative to other points. Let three known points j, k, l in 2D space have Euclidean coordinates $p_j, p_k, p_l \in R^2$. Suppose the coordinates of point i satisfy the following equation.

$$p_i = a_{ij}p_j + a_{ik}p_k + a_{il}p_l$$

$$a_{ij} + a_{ik} + a_{il} = 1$$

Then, $\{a_{ij}, a_{ik}, a_{il}\}$ are the barycentric coordinates of node i relative to nodes i, j, k. In 2D space, the barycentric coordinates of point i can be computed using the signed area of triangles. Specifically, for the example shown in Figure 1, the barycentric coordinates $\{a_{il}, a_{ij}, a_{ik}\}$ of point i are calculated as follows.

$$a_{ij} = \frac{S_{ikl}}{S_{jkl}}, a_{ik} = \frac{S_{jil}}{S_{jkl}}, a_{il} = \frac{S_{jki}}{S_{jkl}}$$

The terms $S_{ikl}, S_{jil}, S_{jki}, S_{jkl}$ represent the signed areas of triangles ikl, jil, jki, jkl, respectively. If points j, k, l satisfy the right-hand rule, the sign of the signed area S_{jkl} is positive; otherwise, the sign of S_{jkl} is negative.

Assuming we know the coordinates of four points i, j, k, l in a two-dimensional space, denoted as $p_i, p_j, p_k, p_l \in R^2$, the areas of triangles $S_{ikl}, S_{jil}, S_{jki}, S_{jkl}$ can be calculated using determinants.

$$S_{ikl} = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ p_i & p_k & p_l \end{vmatrix}$$

$$S_{jil} = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ p_j & p_i & p_l \end{vmatrix}$$

$$S_{jki} = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ p_j & p_k & p_i \end{vmatrix}$$

$$S_{jkl} = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ p_j & p_k & p_l \end{vmatrix}$$

It is required that the area of triangle jkl is nonzero; otherwise, division by zero will occur when computing the barycentric coordinates a_{ij}, a_{ik}, a_{il}.

3. Representing pose transformation with barycentric coordinates

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

In SLAM algorithms, various methods can be used to obtain the pose transformation between two keyframes (in the 2D 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	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 frames of point clouds, various point cloud registration algorithms can be employed to calculate the pose transformation between the frames. 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 is always the pose transformation between two keyframes (nodes). The goal of back-end optimization is to estimate an accurate global map based on these pose transformation measurements. Let us denote a keyframe as node $i \in V$, where each node iii has 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. Assume that the position of node i in the 2D space is $p_i = [x_i, y_i]^T$. Let the set of all pose transformations T_{ij} be denoted as $E = \{T_{ij}, i, j \in V\}$. 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 transform the pose transformation $T_{ij} \in R^{3\times3}$ between any two nodes (node i and node j) into a linear equation constraint. Given that the coordinates of node i are $p_i \in R^2$, we define a virtual node i_x located on the x-axis of the local coordinate system Σ_i, with coordinates p_{ix}, such that $||p_{ix} - p_i|| = 1$. Similarly, we define a virtual node i_y located on the y-axis of the local coordinate system Σ_i, with coordinates p_{iy}, such that $||p_{iy} - p_i|| = 1$. In the local coordinate system Σ_i, node i serves as the origin of the coordinate system, while i_x and i_y represent the unit vector nodes along the two axes of the local coordinate system.

Similarly, for node j, whose coordinates are $p_j \in R^2$, we define a virtual node j_x located on the x-axis of the local coordinate system Σ_j, satisfying $||p_{jx} - p_j|| = 1$. Likewise, we define a virtual node j_y located on the y-axis of the local coordinate system Σ_j, satisfying $||p_{jy} - p_j|| = 1$. In the local coordinate system Σ_j, node j serves as the origin of the coordinate system, while j_x and j_y represent the unit vector nodes along the two axes of the local coordinate system.

Let the pose transformation from node i to node j be denoted as $T_{ij} \in R^{3\times3}$.

$$T_{ij} = \begin{bmatrix} R_{ij} & t_{ij} \\ 0 & 1 \end{bmatrix} \in R^{3\times3}, R_{ij} \in R^{2\times2}$$

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

The positions of node j and its virtual nodes j_x and j_y in the local coordinate system Σ_i are expressed as follows.

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

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

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

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

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

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

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

If the global coordinates of all points in the local coordinate system Σ_i are known, denoted as (p_i, p_{ix}, p_{iy}), it is necessary to represent the global coordinates of all points in the local coordinate system Σ_j, denoted as (p_j, p_{jx}, p_{jy}), using barycentric coordinates. Through this process, we can recursively compute the local coordinate systems for all nodes, starting from the first node. Furthermore, since node i and the virtual nodes i_x and i_y lie on different axes of the coordinate system, the area of the triangle i-ix-iy is non-zero, which satisfies the conditions for computing barycentric coordinates.

Because barycentric coordinates remain invariant under any rotation or translation applied to the set of nodes $V_{ij} = \{i, i_x, i_y, j, j_x, j_y\}$, we can compute the barycentric coordinates using the relative positions of the nodes in the local coordinate system Σ_i, denoted as $(p_i^i, p_{ix}^i, p_{iy}^i, p_j^i, p_{jx}^i, p_{jy}^i)$. The barycentric coordinates obtained in this manner remain valid for the global positions of the nodes $(p_i, p_ix, p_{iy}, p_j, p_{jx}, p_{jy})$ in the global coordinate system.

First, we construct the barycentric coordinates of node j using the local coordinate system Σ_i, which is defined by three nodes: i, i_x, and i_y.

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

$$a_{j,i} = \frac{S_{j,ix,iy}}{S_{i,ix,iy}} \quad (1.1)$$

$$a_{j,ix} = \frac{V_{i,j,iy}}{V_{i,ix,iy}} \quad (1.2)$$

$$a_{j,iy} = \frac{V_{i,ix,j}}{V_{i,ix,iy}} \quad (1.3)$$

$$S_{i,ix,iy} = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ p_i^i & p_{ix}^i & p_{iy}^i \end{vmatrix} \quad (1.4)$$

$$S_{j,ix,iy} = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ p_j^i & p_{ix}^i & p_{iy}^i \end{vmatrix} \quad (1.5)$$

$$S_{i,j,iy} = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ p_i^i & p_j^i & p_{iy}^i \end{vmatrix} \quad (1.6)$$

$$S_{i,ix,j} = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ p_i^i & p_{ix}^i & p_j^i \end{vmatrix} \quad (1.7)$$

Then, we construct the barycentric coordinates of node j_x using the same local coordinate system Σ_i, defined by the three nodes: i, i_x, and i_y.

$$p_{jx} = a_{jx,i}p_i + a_{jx,ix}p_{ix} + a_{jx,iy}p_{iy} \quad (2)$$

$$a_{jx,i} = \frac{S_{jx,ix,iy}}{S_{i,ix,iy}} \quad (2.1)$$

$$a_{jx,ix} = \frac{S_{i,jx,iy}}{S_{i,ix,iy}} \quad (2.2)$$

$$a_{jx,iy} = \frac{S_{i,ix,jx}}{S_{i,ix,iy}} \quad (2.3)$$

$$S_{i,ix,iy} = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ p_i^i & p_{ix}^i & p_{iy}^i \end{vmatrix} \quad (2.4)$$

$$S_{jx,ix,iy} = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ p_{jx}^i & p_{ix}^i & p_{iy}^i \end{vmatrix} \quad (2.5)$$

$$S_{i,jx,iy} = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ p_i^i & p_{jx}^i & p_{iy}^i \end{vmatrix} \quad (2.6)$$

$$S_{i,ix,jx} = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ p_i^i & p_{ix}^i & p_{jx}^i \end{vmatrix} \quad (2.7)$$

Finally, we construct the barycentric coordinates of node j_γ using the local coordinate system Σ_i, again defined by the three nodes: i, i_x, and i_y.

$$p_{jy} = a_{jy,i}p_i + a_{jy,ix}p_{ix} + a_{jy,iy}p_{iy} \quad (3)$$

$$a_{jy,i} = \frac{S_{jy,ix,iy}}{S_{i,ix,iy}} \quad (3.1)$$

$$a_{jy,ix} = \frac{S_{i,jy,iy}}{S_{i,ix,iy}} \quad (3.2)$$

$$a_{jy,iy} = \frac{S_{i,ix,jy}}{S_{i,ix,iy}} \quad (3.3)$$

$$S_{i,ix,iy} = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ p_i^i & p_{ix}^i & p_{iy}^i \end{vmatrix} \quad (3.4)$$

$$S_{jy,ix,iy} = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ p_{jy}^i & p_{ix}^i & p_{iy}^i \end{vmatrix} \quad (3.5)$$

$$S_{i,jy,iy} = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ p_i^i & p_{jy}^i & p_{iy}^i \end{vmatrix} \quad (3.6)$$

$$S_{i,ix,jy} = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ p_i^i & p_{ix}^i & p_{jy}^i \end{vmatrix} \quad (3.7)$$

If the global coordinates of all points in the local coordinate system Σ_i—denoted as (p_i, p_{ix}, p_{iy})—are known, we can use equations (1), (2), and (3) to compute the global coordinates of all points in the local coordinate system Σ_j, denoted as (p_j, p_{jx}, p_{jy}). However, when the problem is reversed—i.e., if the global coordinates of all points in the local coordinate system Σ_j (p_j, p_{jx}, p_{jy}) are given—it is not guaranteed that the global coordinates of all points in the local coordinate system Σ_i (p_i, p_{ix}, p_{iy}) can be computed using the above equations.

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

3.2 Representing nodes i,ix,iy with nodes j,jx,jy

In this section, we construct symmetric linear equations. We utilize the global coordinates of all points in the local coordinate system Σ_j—denoted as (p_j, p_{jx}, p_{jy})—to represent the global coordinates of all points in the local coordinate system Σ_i, expressed in barycentric coordinates as (p_i, p_{ix}, p_{iy}).

Given that the pose transformation from node i to node j is $T_{ij} \in R^{3\times3}$, the pose transformation from node j to node i is $T_{ji} = (T_{ij})^{-1}$.

$$T_{ji} = \begin{bmatrix} R_{ji} & t_{ji} \\ 0 & 1 \end{bmatrix} \in R^{3\times3}, R_{ji} \in R^{2\times2}$$

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

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

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

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

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

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

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

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

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

Given the global coordinates of all points in the local coordinate system Σ_j (p_j, p_{jx}, p_{jy}), we aim 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}). Since node j and the virtual nodes j_x and j_y lie on different axes of the coordinate system, the area of the triangle j-jx-jy is non-zero, thereby satisfying the conditions required for the computation of barycentric coordinates.

Since the barycentric coordinates in the node set $V_{ij} = \{i, i_x, i_y, j, j_x, j_y\}$ remain invariant under arbitrary rotations and translations, we can calculate the barycentric coordinates using the relative positions of the nodes in the local coordinate system Σ_j: $(p_i^j, p_{ix}^j, p_{iy}^j, p_j^j, p_{jx}^j, p_{jy}^j)$. The barycentric coordinates computed in this way are still valid for the positions of the nodes in the global coordinate system $(p_i, p_{ix}, p_{iy}, p_j, p_{jx}, p_{jy})$.

We first construct the barycentric coordinates of node i using the four nodes j, j_x, j_y in the local coordinate system Σ_j.

$$p_i = a_{i,j}p_j + a_{i,jx}p_{jx} + a_{i,jy}p_{jy} \tag{4}$$

$$a_{i,j} = \frac{S_{i,jx,jy}}{S_{j,jx,jy}} \tag{4.1}$$

$$a_{i,jx} = \frac{S_{j,i,jy}}{S_{j,jx,jy}} \tag{4.2}$$

$$a_{i,jy} = \frac{S_{j,jx,i}}{S_{j,jx,jy}} \tag{4.3}$$

$$S_{j,jx,jy} = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ p_j^j & p_{jx}^j & p_{jy}^j \end{vmatrix} \tag{4.4}$$

$$S_{i,jx,jy} = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ p_i^j & p_{jx}^j & p_{jy}^j \end{vmatrix} \tag{4.5}$$

$$S_{j,i,jy} = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ p_j^j & p_i^j & p_{jy}^j \end{vmatrix} \tag{4.6}$$

$$S_{j,jx,i} = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ p_j^j & p_{jx}^j & p_i^j \end{vmatrix} \tag{4.7}$$

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

$$p_{ix} = a_{ix,j}p_j + a_{ix,jx}p_{jx} + a_{ix,jy}p_{jy} \tag{5}$$

$$a_{ix,j} = \frac{S_{ix,jx,jy}}{S_{j,jx,jy}} \tag{5.1}$$

$$a_{ix,jx} = \frac{S_{j,ix,jy}}{S_{j,jx,jy}} \tag{5.2}$$

$$a_{ix,jy} = \frac{S_{j,jx,ix}}{S_{j,jx,jy}} \tag{5.3}$$

$$S_{j,jx,jy} = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ p_j^j & p_{jx}^j & p_{jy}^j \end{vmatrix} \tag{5.4}$$

$$S_{ix,jx,jy} = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ p_{ix}^j & p_{jx}^j & p_{jy}^j \end{vmatrix} \tag{5.5}$$

$$S_{j,ix,jy} = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ p_j^j & p_{ix}^j & p_{jy}^j \end{vmatrix} \tag{5.6}$$

$$S_{j,jx,ix} = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ p_j^j & p_{jx}^j & p_{ix}^j \end{vmatrix} \tag{5.7}$$

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

$$p_{iy} = a_{iy,j}p_j + a_{iy,jx}p_{jx} + a_{iy,jy}p_{jy} \tag{6}$$

$$a_{iy,j} = \frac{S_{iy,jx,jy}}{S_{j,jx,jy}} \tag{6.1}$$

$$a_{iy,jx} = \frac{S_{j,iy,jy}}{S_{j,jx,jy}} \tag{6.2}$$

$$a_{iy,jy} = \frac{S_{j,jx,iy}}{S_{j,jx,jy}} \tag{6.3}$$

$$S_{j,jx,jy} = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ p_j^j & p_{jx}^j & p_{jy}^j \end{vmatrix} \tag{6.4}$$

$$S_{iy,jx,jy} = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ p_{iy}^j & p_{jx}^j & p_{jy}^j \end{vmatrix} \tag{6.5}$$

$$S_{j,iy,jy} = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ p_j^j & p_{iy}^j & p_{jy}^j \end{vmatrix} \tag{6.6}$$

$$S_{j,jx,iy} = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ p_j^j & p_{jx}^j & p_{iy}^j \end{vmatrix} \tag{6.7}$$

If the global coordinates of all points in the local coordinate system Σ_j (denoted as p_j, p_{jx}, p_{jy}) are known, the global coordinates of all points in the local coordinate system Σ_i (denoted as p_i, p_{ix}, p_{iy}) can be calculated using equations (4), (5), and (6).

At the same time, by using equations (1)–(6), we can compute the coordinates of nodes j, j_x, j_y from the coordinates of nodes i, i_x, i_y, or vice versa, compute the coordinates of nodes i, i_x, i_y from the coordinates of nodes j, j_x, j_y. Therefore, for any pose transformation $T_{ij} \in E$, the aforementioned six linear equations can be used to represent this transformation. Additionally, these six linear equations remain valid when the graph formed by all nodes in the set $V_{ij} = \{i, i_x, i_y, j, j_x, j_y\}$ is scaled. Consequently, these linear equations can be broadly regarded as defining a constraint based on a similarity transformation.

Let the set of virtual nodes be denoted as $V_v = \{1_x, 1_y, \ldots, n_x, n_y\}$. Let the coordinate vector of all nodes $i \in V$ be represented as $p = [p_1^T, p_2^T, \ldots, p_n^T]^T \in R^{2n}$. Similarly, let the coordinate vector of all virtual nodes i_x, i_y for $i \in V$ be represented as $p_v = \left[p_{1x}^T, p_{1y}^T, \ldots, p_{nx}^T, p_{ny}^T\right]^T \in R^{4n}$. By combining the coordinate vectors p and p_v, we obtain the augmented coordinate vector $p_{acc} = [p^T, p_v^T]^T \in R^{6n}$. Each pose transformation $T_{ij} \in V$ can be expressed as six linear equations. By combining these linear equations, the following system of linear equations can be obtained.

$$Ap_{acc} = 0$$

The pseudocode for the algorithm is as follows.

Procedure 1: Representing pose transformation with linear equations

---

Input: Graph $G = [V, E]$, where V represents the set of n nodes, and E represents the set of all pose transformations

Output: Matrix A

Define the vector of all nodes as $p = [p_1^T, p_2^T, \ldots, p_n^T]^T \in R^{2n}$.

Define the virtual node vector of all nodes as $p_v = \left[p_{1x}^T, p_{1y}^T, \ldots, p_{nx}^T, p_{ny}^T\right]^T \in R^{4n}$.

Combine the coordinate vectors p and p_v to obtain p_{acc}, where $p_{acc} = [p^T, p_v^T]^T \in R^{6n}$.

For $T_{ij} \in V$:

//The coordinates of nodes i, i_x, i_y, j, j_x, and j_y in the local coordinate system Σ_i

Given the relative pose transformation from node i to node

j as $T_{ij} \in R^{3\times3}$, where $T_{ij} = \begin{bmatrix} R_{ij} & t_{ij} \\ 0 & 1 \end{bmatrix} \in R^{3\times3}, R_{ij} \in R^{2\times2}, t_{ij} = [x_{ij}, y_{ij}]^T \in R^2$

The positions of node j and its virtual nodes j_x and j_y in the local coordinate system Σ_i are given as follows.

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

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

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

The coordinates of node i and its virtual nodes i_x and i_y in the local coordinate system Σ_i are as follows: $p_i^i = [0,0]^T, p_{ix}^i = [1,0]^T, p_{iy}^i = [0,1]^T$;

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

Using three nodes i, i_x, and i_y in the local coordinate system Σ_i, the barycentric coordinates of node j are constructed as: $p_j = a_{j,i}p_i + a_{j,ix}p_{ix} + a_{j,iy}p_{iy}$, which is incorporated into the linear system of equations $Ap_{acc} = 0$. The parameters $a_{j,i}, a_{j,ix}$, and $a_{j,iy}$ are computed using the relative coordinates $p_i^i, p_{ix}^i, p_{iy}^i, p_j^i, p_{jx}^i$, and p_{jy}^i, as shown in equations (1.1)–(1.7).

Using three nodes i, i_x, and i_y in the local coordinate system Σ_i, the barycentric coordinates of node j_x are constructed as: $p_{jx} = a_{jx,i}p_i + a_{jx,ix}p_{ix} + a_{jx,iy}p_{iy}$, which is incorporated into the linear system of equations $Ap_{acc} = 0$. The parameters $a_{jx,i}, a_{jx,ix}$, and $a_{jx,iy}$ are calculated using the relative coordinates $p_i^i, p_{ix}^i, p_{iy}^i, p_j^i, p_{jx}^i$, and p_{jy}^i, as detailed in equations (2.1)–(2.7).

Using three nodes i, i_x, and i_y in the local coordinate system Σ_i, the barycentric coordinates of node j_y are constructed as: $p_{jy} = a_{jy,i}p_i + a_{jy,ix}p_{ix} + a_{jy,iy}p_{iy}$, which is incorporated into the linear system of equations $Ap_{acc} = 0$. The parameters $a_{jy,i}, a_{jy,ix}$, and $a_{jy,iy}$ are computed using the relative coordinates $p_i^i, p_{ix}^i, p_{iy}^i, p_j^i, p_{jx}^i$, and p_{jy}^i, as described in equations (3.1)–(3.7).

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

Given that the pose transformation from node j to node i is represented as $T_{ji} \in R^{3\times3}$, where: $T_{ji} = \begin{bmatrix} R_{ji} & t_{ji} \\ 0 & 1 \end{bmatrix} \in R^{3\times3}, R_{ji} \in R^{2\times2}, t_{ji} = [x_{ji}, y_{ji}]^T \in R^2$.

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

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

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

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

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

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

In the local coordinate system Σ_j, the barycentric coordinates of node i are constructed using the three nodes j, j_x, and j_y, expressed as: $p_i = a_{i,j}p_j + a_{i,jx}p_{jx} + a_{i,jy}p_{jy}$, and incorporated into the linear equation system $Ap_{acc} = 0$. The coefficients $a_{i,j}, a_{i,jx}$, and $a_{i,jy}$ are computed using the relative coordinates $p_i^j, p_{ix}^j, p_{iy}^j, p_j^j, p_{jx}^j$, and p_{jy}^j, as detailed in equations (4.1)–(4.7).

In the local coordinate system Σ_j, the barycentric coordinates of node ix are constructed using the three nodes j, j_x, and j_y, expressed as: $p_{ix} = a_{ix,j}p_j + a_{ix,jx}p_{jx} + a_{ix,jy}p_{jy}$, and incorporated into the linear equation system $Ap_{acc} = 0$. The coefficients $a_{ix,j}, a_{ix,jx}$, and $a_{ix,jy}$ are computed using the relative coordinates $p_i^j, p_{ix}^j, p_{iy}^j, p_j^j, p_{jx}^j$, and p_{jy}^j, as detailed in equations (5.1)–(5.7).

In the local coordinate system Σ_j, the barycentric coordinates of node i_y are constructed using the three nodes j, j_x, and j_y, expressed as: $p_{iy} = a_{iy,j}p_j + a_{iy,jx}p_{jx} + a_{iy,jy}p_{jy}$, and incorporated into the linear equation system $Ap_{acc} = 0$. The coefficients $a_{iy,j}, a_{iy,jx}$, and $a_{iy,jy}$ are computed using the relative coordinates $p_i^j, p_{ix}^j, p_{iy}^j, p_j^j, p_{jx}^j$, and p_{jy}^j, as detailed in equations (6.1)–(6.7).

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 odometers, and loop closure detection. The following subsections will discuss the pose transformations obtained from each type of sensor in detail.

4.1 Non-GPS sensors

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

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

Similarly, the loop closure detection module can also provide the relative pose transformation T_{ij} between two nodes (node i and node j). In this case, node i and node j are typically two adjacent nodes identified after the robot completes a loop and returns to a previously visited area. While node i and node j are temporally distant in the robot's motion history, they are spatially close to each other.

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

4.2 GPS sensors

The case of GPS sensors is relatively unique, as the data obtained from GPS sensors consists of latitude and longitude coordinates. Using specific formulas, these latitude and longitude values can be converted into measurable distances. Therefore, the positions detected by GPS sensors are always within a fixed latitude and longitude coordinate system, or in other words, GPS sensors operate 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 given nodes (node i and node j). To obtain the pose transformation between two nodes that are far apart (e.g., node i and node $i + n$), one must iteratively accumulate the transformations, such that: $T_{i,i+n} = T_{i,i+1}T_{i+1,i+2}\dots T_{i+n-1,i+n}$. However, this method of accumulating pose transformations $T_{i,i+n}$ introduces errors that gradually accumulate as well. In the case of GPS sensors, due to their fixed global coordinate system, the translation vector $t_{i,i+n}$ between two distant nodes (e.g., node i and node $i + n$) can still maintain very high accuracy. This characteristic of GPS data makes it particularly valuable for maintaining the overall shape and accuracy of large-scale maps.

Another special case of GPS sensors is that GPS data only provides displacement information without any rotational information. Suppose we obtain the GPS data of node i (its position in the latitude and longitude coordinate system) and then the GPS data of node j. In this case, we can only acquire the translational vector $t_{ij} \in R^2$ from node i to node j in the latitude and longitude coordinate system, without any rotation 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 a linear equation constraint. Although it is possible to construct a nonlinear pose graph constraint, 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 height information. Even when GPS sensors provide height data, the accuracy of the height measurements is significantly lower than that of the latitude and longitude data. If the GPS sensor data contains only latitude and longitude, it is possible to determine the planar translation $t_{ij} \in R^2$ between two nodes (node i and node j). By selecting three nodes (nodes i,j, and k), we can construct a system of linear equations using planar triangles.

For the processing of GPS sensor data, we adopt the method presented in [1], which uses similarity triangle constraints to construct linear equations. To avoid excessive verbosity in this paper, we directly present the linear system of equations derived from GPS data as follows. The specific process for constructing these linear equations is detailed in [1].

$$Cp_{acc} = 0$$

This includes constructing linear equations using adjacent GPS data and constructing linear equations using cross-node GPS data.

3.3 Variance of pose transformation

We can obtain the relative pose transformation between two frames through various sensors, such as cameras, LiDAR, IMU sensors, GPS sensors, wheel encoders, and loop closure detection. Typically, the pose transformations obtained from different sensors exhibit different variances. These variances can either be preset according to the characteristics of each sensor or derived during the computation of 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_{ij}^2 to represent the variance of the entire pose transformation matrix.

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

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

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

The error vector $v_1 \sim N(0, \Sigma_1), v_2 \sim N(0, \Sigma_2)$; if we ignore the correlations between the individual 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 corresponding to the pose transformation T_{ij} in the linear equation as the variance of that equation. For the error vector v_2, each linear equation is constructed using two pose transformations (e.g., T_{ij}, T_{ik}). For simplicity, we set 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), where the set V contains n nodes. The coordinate vector of all nodes is denoted as $p = [p_1^T, \dots, p_n^T]^T \in R^{2n}$. In the local coordinate system Σ_i of each node $i \in V$, two virtual nodes i_x and i_y are defined. The set of virtual nodes is denoted as $V_v = \{i_x, i_y, i \in V\}$, and the coordinate vector of all virtual nodes is $p_v = [p_{1x}^T, p_{1y}^T, \dots, p_{nx}^T, p_{ny}^T] \in R^{4n}$. By combining the coordinate vectors p and p_v, we obtain the augmented coordinate vector $p_{acc} = [p^T, p_v^T]^T \in R^{6n}$.

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 (i and j) can be expressed as six 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 of the first node, Σ_1, as the global coordinate system Σ_g, and calculate the coordinates of other nodes accordingly. If we choose a different coordinate system Σ_i as the global coordinate system Σ_g, then using the corresponding pose transformation T_{1i}, all coordinates can be transformed into the coordinate system Σ_i. Assuming the local coordinate system Σ_1 is the global coordinate system Σ_g, the coordinates of node 1 are $p_1 = [0,0]^T$, the coordinates of the virtual node 1_x are $p_{1x} = [1,0]^T$, and the coordinates of the virtual node 1_y are $p_{1y} = [0,1]^T$. According to the construction method of the linear equation system $Ap_{acc} = 0$, as long as the coordinates of any one node in its local coordinate system (e.g., p_1, p_{1x}, p_{1y}) are determined, 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 of the system remains scalable.

If the coordinates of other nodes in p_{acc} are calculated using the coordinates of p_1, p_{1x}, and p_{1y}, this is equivalent to determining the scale of the graph using the coordinates of p_1, p_{1x}, and p_{1y}. As this scale information propagates outward from node 1 to other nodes, the error will gradually accumulate and amplify. For example, if we define the unit vector lengths of the two axes in the local coordinate system Σ_1 as 1 ($\rho_{i,ix} = 1, \rho_{i,iy} = 1$), the calculated coordinates of node n may result in the vector lengths of the two axes in the local coordinate system Σ_n (formed by node n, node n_x, and node n_y) not necessarily maintaining a magnitude of 1 ($\rho_{n,nx}, \rho_{n,ny}$). To address this, we can adopt the following method to determine a more accurate map scale.

Let the coordinates of node 1 be $p_1 = [0,0]^T$, the coordinates of the virtual node 1_x be $p_{1x} = [\rho, 0]^T$, and the coordinates of the virtual node 1_y be $p_{1y} = [0, \rho]^T$. Let $p_{sub} \in R^{6n-6}$ represent the coordinate vector obtained by removing p_1, p_{1x}, and p_{1y} from the coordinate vector p_{acc}. Substituting the coordinates p_1, p_{1x}, and p_{1y} into the equation, the linear system $Ap_{acc} = 0 + v_1$ is transformed as follows.

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

Where $v_{sub1} \sim N(0, \Sigma_{sub1})$ represents the noise associated with each linear equation. The variance of each linear equation is represented by the variance σ_{ij}^2 of the corresponding pose transformation T_{ij}. The number of linear equations in the system remains unchanged, and the variance corresponding to each linear equation does not change. Thus, the error vector remains $v_{sub1} = v_1$, and the covariance matrix is $\Sigma_{sub1} = \Sigma_1$. The least squares solution of this linear system is 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 component of each coordinate (e.g., p_i, p_{ix}, p_{iy}) is a linear function of the unknown variable ρ.

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

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

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

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

At the same time, the coordinates of node 1 and the virtual nodes 1_x and 1_y are included in the coordinate vector p_{acc}, where each component is also a linear function of the unknown variable ρ. Subsequently, we define the following error function J_1. This error function ensures that, in the coordinate system Σ_i of each node i, the distance from node i to its virtual node i_x is 1, and the distance from node i to its virtual node i_y is also 1.

$$J_1 = \sum_{i \in V} \left(\left(||p_i - p_{ix}|^2 - 1 \right)^2 + \left(\left| |p_i - p_{iy}| \right|^2 - 1 \right)^2 \right)$$

$$= \sum_{i \in V} + \left(\left((a_i - a_{iz})\rho + (b_i - b_{iz}) \right)^2 + \left((c_i - c_{iz})\rho + (d_i - d_{iz}) \right)^2 + \left((e_i - e_{iz})\rho + (f_i - f_{iz}) \right)^2 - 1 \right)^2 +$$

$$\left(\left((a_i - a_{ix})\rho + (b_i - b_{ix}) \right)^2 + \left((c_i - c_{ix})\rho + (d_i - d_{ix}) \right)^2 + \left((e_i - e_{ix})\rho + (f_i - f_{ix}) \right)^2 - 1 \right)^2$$

$$= A_1\rho^4 + B_1\rho^3 + C_1\rho^2 + D_1\rho + E_1$$

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 translation component is $t_{ij} \in R^2$, then the distance between node i and node j is $\rho_{ij} = \left|\left|t_{ij}\right|\right|$. Meanwhile, the magnitude can be computed (as a linear function of the unknown ρ) based on the solved coordinates p_i of node i and p_j of node j. The difference between these two magnitudes is used to define the error function.

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

$$= \sum_{T_{ij} \in E} \left(\left((a_i - a_j)\rho + (b_i - b_j) \right)^2 + \left((c_i - c_j)\rho + (d_i - d_j) \right)^2 - \rho_{ij}^2 \right)^2$$

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

By adding the error function J_1 and the error function J_2, the composite error function J is obtained.

$$J = J_1 + J_2$$

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

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 formula of the cubic equation is used to calculate its three roots, denoted as ρ_1, ρ_2, and ρ_3. After discarding any

complex roots among these, the second derivative of the function J is employed to evaluate the remaining real roots.

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

If the second derivative of J at root ρ_i is positive, then ρ_i corresponds to the minimum value of the function J. Conversely, if the second derivative of J at root ρ_i is negative, then ρ_i corresponds to a local maximum of J. Once the optimized map scale ρ is determined, it can be substituted into the coordinates $p_{acc} \in R^{6n}$. The pseudocode for computing the map scale and the node coordinates is as follows.

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

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

Output: Coordinate vector $p_{acc} \in R^{6n}$, map scale ρ

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

Let the virtual node coordinate vector be $p_v = [p_{1x}^T, p_{1y}^T, \ldots, p_{nx}^T, p_{ny}^T] \in R^{4n}$.

Let the combined coordinate vector p_{acc} of p and p_v be $p_{acc} = [p^T, p_v^T]^T \in R^{6n}$.

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

Let p_{acc} exclude the coordinates of p_1, p_{1x}, and p_{1y}, forming a sub-coordinate vector $p_{sub} \in R^{6n-6}$.

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

The planar components of virtual nodes p_{sub} are then solved using the linear equation system $A_{sub}p_{sub} = b_{sub} + v_{sub1}$. The least-squares solution of this linear 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)

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

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

The total error function J is obtained by combining J_1 and J_2: $J = J_1 + J_2 = A\rho^4 + B\rho^3 + C\rho^2 + D\rho + E$.

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

Among the three roots ρ_1, ρ_2, ρ_3, any complex roots are discarded. For the remaining real roots, the second derivative of the error function J is used to determine which root corresponds to the minimum value of J. The optimal root is denoted as ρ_i.

The optimal parameter ρ is substituted into the coordinate vector p_{acc}.

4.2 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 as follows.

$$Ap_{acc} = 0 + v_1 \quad (7)$$

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 augmented as follows.

$$Cp_{acc} = 0 + v_2 \quad (8)$$

Then, equations (7) and (8) 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 $v_3 \sim N(0, \Sigma_3)$. By ignoring the correlations between the individual linear equations, the covariance matrix Σ_3 becomes a diagonal matrix and satisfies the following relationship: $\Sigma_3 = \mathrm{diag}\{\Sigma_1, \Sigma_2\}$.

We define the following coordinates: $p_1 = [0,0]^T, p_{1x} = [\rho, 0]^T, p_{1y} = [0, \rho]^T$. 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 $p_{sub} \in R^{6n-6}$ is the vector p_{acc} with p_1, p_{1x}, and p_{1y} removed. The error vector $v_{sub3} \sim N(0, \Sigma_{sub3})$ represents the noise corresponding to each of the linear equations. The number of linear equations in the linear system remains unchanged, and the variance associated with each linear equation also remains unchanged. Thus, the error vector satisfies $v_{sub3} = v_3$, and the covariance matrix satisfies $\Sigma_{sub3} = \Sigma_3$. The least-squares solution to this linear equation 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 systems of equations: $Ap_{acc} = 0, Cp_{acc} = 0$; covariance matrices: Σ_1, Σ_2

Output: Coordinate vector $p_{acc} \in R^{6n}$; map scale ρ

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

Let the virtual node coordinate vector be $p_v = [p_{1x}^T, p_{1y}^T, \ldots, p_{nx}^T, p_{ny}^T] \in R^{4n}$.

By combining p and p_v, the full coordinate vector is $p_{acc} = [p^T, p_v^T]^T \in R^{6n}$.

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

The coordinate vector p_{acc}, with p_1, p_{1x}, and p_{1y} removed, becomes $p_{sub} \in R^{6n-6}$.

The linear equations $Ap_{acc} = 0$ and $Cp_{acc} = 0$ are combined into a single linear equation: $Ep_{acc} = 0$.

Substituting the coordinates of p_1, p_{1x}, and p_{1y} into the equations, the system $Ep_{acc} = 0$ is transformed into $E_{sub}p_{sub} = F_{sub}$.

The coordinate vector p_{sub} is computed using the linear system $E_{sub}p_{sub} = F_{sub} + v_{sub3}$. The least-squares solution to this 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 ρ)

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

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

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 find the parameter ρ that minimizes the 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, calculate the three roots ρ_1, ρ_2, ρ_3. Discard any complex roots, and use the second derivative of J to select the root ρ_i that corresponds to the minimum value of J.

Substitute the parameter ρ 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 $e(T_{ij}) = h(T_{ij})^2$, 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.

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

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, $e(T_{ij}) = h(T_{ij})^2$, 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.

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

The probability p_{ij} is used in the outlier removal algorithm to find 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 been provided in [1]. However, for the sake of completeness in the SLAM framework, we explicitly present the calculation method for the local coordinate system here. We have already computed the coordinates of each node in the global coordinate system Σ_g for the graph $G = [V, E]$, represented as $p = [p_1^T, p_2^T, \dots, p_n^T] \in R^{2n}$. Let the global coordinates of node i be $p_i = [x_i, y_i]^T$. Denote the neighboring nodes of node i by $j \in N_i$, 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 expressed as $p_j^r = [x_j - x_i, y_j - y_i]^T$. The relative coordinate vectors of these neighboring nodes in the global coordinate system are represented by $P_i^n = [p_{j1}^r, \dots, p_{jn}^r] \in R^{2\times|N_i|}$, where $j_1, \dots, j_n \in N_i$. Let the pose transformation from node i to its neighboring node $j \in N_i$ be denoted as $T_{ij} = \begin{bmatrix} R_{ij} & t_{ij} \\ 0 & 1 \end{bmatrix} \in R^{3\times3}$, where the translation vector is $t_{ij} = [x_{ij}, y_{ij}]^T$, for $j \in N_i$. Then, the relative position vectors of the neighboring nodes $j \in N_i$ in the local coordinate system Σ_i are expressed as $Q_i^n = [t_{ij1}, \dots, t_{ijn}] = [q_{ij1}, \dots, q_{ijn}] \in R^{2\times|N_i|}, j_1, \dots, j_n \in N_i$.

Then, we have calculated the coordinates of each node $i \in V$ and its 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 coordinate of node i is $p_i = [x_i, y_i]^T$. The relative coordinates of the virtual nodes i_x and i_y with respect to node i are defined 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 $P_i^v = [p_{ix}^r, p_{iy}^r]$. The coordinates of the virtual nodes i_x and i_y in the local coordinate system Σ_i 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 concatenated to form $P_i = [P_i^n, P_i^v] \in R^{2|N_i|+2}$, and similarly, Q_i^n and Q_i^v are concatenated to form $Q_i = [Q_i^n, Q_i^v] \in R^{2|N_i|+2}$. Using these, the rotation matrix R_i of the local coordinate system of node i can be computed through point set registration methods.

Here, the coordinate vectors P_i and Q_i are constructed using 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, we have opted not to include them for the following reasons. First, the rotation matrix R_i can already be determined as long as at least one node (e.g., the virtual nodes i_x and i_y) is included. Second, the information of the neighboring nodes $j \in N_i$ already provides a sufficient approximation of their positions. Including the virtual coordinates j_x and j_y of these nodes may unnecessarily increase computational complexity.

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

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

We seek the rotation matrix $R_i \in R^{2\times2}$ that minimizes the error function $E(R_i)$. This problem belongs to the class of point cloud registration problems, and we can compute the matrix R_i using the Singular Value Decomposition (SVD)

method 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 relative to the global coordinate system Σ_g is $(R_i)^{-1}$. Using the SVD method in the ICP algorithm, the rotation matrix R that minimizes the error function E(R) is determined as follows.

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

For the matrix $H \in R^{2\times2}$, perform singular value decomposition (SVD) to obtain the orthogonal matrices U and V. The rotation matrix is defined as $R_i = VR^T$. However, if the point set P_i or Q_i is degenerate, the resulting matrix R_i might become a reflection matrix ($\det(R_i) = -1$). To address this, we can adjust the reflection matrix R_i into a proper rotation matrix using the following steps.

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

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

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

Process 4: Calculation of local coordinate system

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

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

The relative coordinates of a neighbor node $j \in N_i$ with respect to node i are given by: $p_j^r = [x_j - x_i, y_j - y_i]^T$.

The relative coordinate vector of all neighbor nodes $j \in N_i$ in the global coordinate system Σ_g is $P_i^n = [p_{j1}^r, \ldots, p_{jn}^r] \in R^{2\times|N_i|}, j_1, \ldots, j_n \in N_i$.

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

The relative coordinates of the virtual nodes i_x, i_y with respect to node i are: $p_{ix}^r = [x_{ix} - x_i, y_{ix} - y_i]^T, p_{iy} = [x_{iy} - x_i, y_{iy} - y_i]^T$; combining these, the vector of virtual nodes' relative coordinates is $P_i^v = [p_{ix}^r, p_{iy}^r] \in R^{2\times2}$.

The coordinates of the virtual nodes i_x, i_y in the local coordinate system Σ_i are $p_{ix}^i = [1,0]^T, p_{iy}^i = [0,1]^T$; combining these, the vector of virtual nodes' local coordinates is $Q_i^v = [p_{ix}^i, p_{iy}^i] \in R^{2\times2}$.

Combine the coordinate vectors P_i^n and P_i^v to form: $P_i = [P_i^n, P_i^v] \in R^{2\times(2n+2)}$; combine Q_i^n and Q_i^v to form: $Q_i = [Q_i^n, Q_i^v] \in R^{2\times(2n+2)}$.

Compute the rotation matrix R_i using point-set registration. First, compute the matrix H: $H = P_i Q_i^T$.

Perform Singular Value Decomposition (SVD) on H: $H = U\Sigma V^T$.

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

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