University of Bonn, IGG
Prof. Dr. Cyrill Stachniss
Graph-SLAM Tutorial

Graph-Based SLAM Exercise

Exercise

Implement a least-squares method to address SLAM in its graph-based formulation. To support this task, we provided a small *Matlab/Octave* framework. The framework contains the following folders:

data contains several datasets, each gives the measurements of one SLAM problem

matlab contains the Matlab/Octave framework with stubs to complete.

plots this folder is used to store images.

The below mentioned tasks should be implemented inside the framework in the directory `matlab` by completing the stubs. After implementing the missing parts, you can run the optimizer by typing `lsSLAM`. The script will produce a plot showing the positions of the robot and (if available) the positions of the landmarks in each iteration. These plots will be saved in the `plots` directory.

1. Implement the missing code in `compute_global_error.m` for computing the current error value for a graph with constraints.
2. Implement the missing code in `linearize_pose_pose_constraint.m` for computing the error and the Jacobian of a pose-pose constraint. Test your implementation with `test_jacobian_pose_pose`.
3. Implement the missing code in `linearize_and_solve.m` for constructing and solving the linear approximation.
4. Implement the missing code in `linearize_pose_landmark_constraint.m` for computing the error and the Jacobian of a pose-landmark constraint. Test your implementation with `test_jacobian_pose_landmark`.
5. Implement the update of the state vector in `lsSLAM.m`.
6. Additional homework: For a deeper understanding, verify the error function and Jacobian for pose-pose as well as pose-landmark constraints as shown in the remainder of this document.

Expected Results

Figure 1 depicts the result that you should obtain after convergence for each dataset. Additionally, the initial and the final error for each dataset should approximately take the values in the table below.

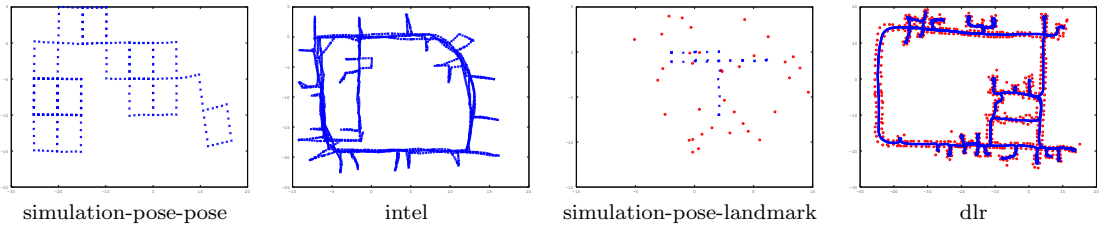

Figure 1: Result for each dataset.

dataset	initial error	final error
simulation-pose-pose.dat	138862234	8269
intel.dat	1795139	360
simulation-pose-landmark.dat	3030	474
dlr.dat	369655336	56860

Hints

- The key elements in the graph datastructure `g` are the node poses/locations stored in `g.x` as well as the edge information `g.edges` consisting of the measurement (`measurement`) between the two nodes and the corresponding information matrix (`information`).
- The state vector (`g.x`) consists of a concatenation of robot poses (and possibly landmark positions), i.e., $\mathbf{x}_{1:N}$ with
  - pose of the robot: $\mathbf{x}_i = (x_i \; y_i \; \theta_i)^T$
    Hint: Use the function v2t($\cdot$) and t2v($\cdot$) that transform a vector into a transformation matrix (v2t($\cdot$)) and vice versa (t2v($\cdot$)). The function invt($\cdot$) inverts a transformation.

$$\text{v2t}(\mathbf{x}_i) = \begin{pmatrix} R_i & \mathbf{t}_i \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} \cos(\theta_i) & -\sin(\theta_i) & x_i \\ \sin(\theta_i) & \cos(\theta_i) & y_i \\ 0 & 0 & 1 \end{pmatrix} = X_i \qquad \text{t2v}(X_i) \quad = \mathbf{x}_i$$

  - position of a landmark: $\mathbf{x}_l = (x_l \; y_l)^T$
- We consider the following error functions:
  - pose-pose constraint: $\mathbf{e}_{ij} = \text{t2v}(Z_{ij}^{-1}(X_i^{-1}X_j))$, where $Z_{ij} = \text{v2t}(\mathbf{z}_{ij})$ is the transformation matrix of the measurement $\mathbf{z}_{ij}^T = (\mathbf{t}_{ij}^T, \theta_{ij})$.
  - pose-landmark constraint: $\mathbf{e}_{il}(\mathbf{x}) = X_i^{-1}\mathbf{x}_l - \mathbf{z}_{il}$

Further Information on the Error Functions and Jacobians for 2D Graph-Based SLAM

In the following we will provide the definitions and the derivations for the Jacobians to implement 2D graph-based SLAM.

Error Functions and Jacobians for a Pose-Pose Constraint

The basic entities of the poses of the robot and the constraint are defined as

$$\begin{aligned} \mathbf{x}_i^\top &= (\mathbf{t}_i^\top, \theta_i) &= (x_i, y_i, \theta_i) &(1)\\ \mathbf{x}_j^\top &= (\mathbf{t}_j^\top, \theta_j) &= (x_j, y_j, \theta_j) &(2)\\ \mathbf{z}_{ij}^\top &= (\mathbf{t}_{ij}^\top, \theta_{ij}) &= (x_{ij}, y_{ij}, \theta_{ij}) &(3) \end{aligned}$$

where $\mathbf{t}_i$ and $\mathbf{t}_{ij}$ are 2D vectors and θ_i and θ_{ij} are rotation angles which are normalized to $[-\pi, \pi)$.

Error Function

The error function is

$$\mathbf{e}_{ij}(\mathbf{x}) = \text{t2v}(Z_{ij}^{-1}(X_i^{-1}X_j)) \quad (4)$$

Jacobian

To derive the Jacobian, it is advantageous to start from:

$$\mathbf{e}_{ij}(\mathbf{x}) = \mathbf{e}_{ij}(\mathbf{x}_i, \mathbf{x}_j) = \begin{pmatrix} R_{ij}^\top(R_i^\top(\mathbf{t}_j - \mathbf{t}_i) - \mathbf{t}_{ij}) \\ \theta_j - \theta_i - \theta_{ij} \end{pmatrix}, \quad (5)$$

where R_i and R_{ij} are the 2×2 rotation matrices of θ_i and θ_{ij} with the following structure

$$R_i = \begin{pmatrix} \cos(\theta_i) & -\sin(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \end{pmatrix} \qquad R_{ij} = \begin{pmatrix} \cos(\theta_{ij}) & -\sin(\theta_{ij}) \\ \sin(\theta_{ij}) & \cos(\theta_{ij}) \end{pmatrix}. \quad (6)$$

The Jacobians of the error function become

$$\mathbf{A}_{ij} = \frac{\partial \mathbf{e}_{ij}(\mathbf{x})}{\partial \mathbf{x}_i} = \begin{pmatrix} -R_{ij}^\top R_i^\top & R_{ij}^\top \frac{\partial R_i^\top}{\partial \theta_i}(\mathbf{t}_j - \mathbf{t}_i) \\ \mathbf{0} & -1 \end{pmatrix} \quad (7)$$

$$\mathbf{B}_{ij} = \frac{\partial \mathbf{e}_{ij}(\mathbf{x})}{\partial \mathbf{x}_j} = \begin{pmatrix} R_{ij}^\top R_i^\top & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix}. \quad (8)$$

And for $\frac{\partial R_i^\top}{\partial \theta_i}$ we obtain

$$\frac{\partial R_i^\top}{\partial \theta_i} = \begin{pmatrix} -\sin(\theta_i) & \cos(\theta_i) \\ -\cos(\theta_i) & -\sin(\theta_i) \end{pmatrix}. \quad (9)$$

Error Functions and Jacobians for a Pose-Landmark Constraint

The basic entities of the poses of the robot and the constraint are defined as

$$\mathbf{x}_i^\top = (\mathbf{t}_i^\top, \theta_i) \tag{10}$$
$$\mathbf{x}_l^\top = (x_l, y_l)^\top \tag{11}$$
$$\mathbf{z}_{il} = (x_{il}, y_{il})^\top, \tag{12}$$

where $\mathbf{t}_i$ is a 2D vector, θ_i is a rotation angle which is normalized to $[-\pi, \pi)$, and $(x_{il}, y_{il})^\top$ is the measurement of $\mathbf{x}_l$ in the local frame of $\mathbf{x}_i$.

Error Function

The error function is

$$\mathbf{e}_{il}(\mathbf{x}) = X_i^{-1}\mathbf{x}_l - \mathbf{z}_{il} \tag{13}$$

Jacobian

To derive the Jacobian, it is advantageous to start from:

$$\mathbf{e}_{il}(\mathbf{x}) = R_i^\top(\mathbf{x}_l - \mathbf{t}_i) - \mathbf{z}_{il}, \tag{14}$$

where R_i is the 2×2 rotation matrices of θ_i with the following structure

$$R_i = \begin{pmatrix} \cos(\theta_i) & -\sin(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \end{pmatrix}. \tag{15}$$

The Jacobians of the error function becomes

$$\mathbf{A}_{il} = \frac{\partial \mathbf{e}_{il}(\mathbf{x})}{\partial \mathbf{x}_i} = \begin{pmatrix} -R_i^\top & \frac{\partial R_i^\top}{\partial \theta_i}(\mathbf{x}_l - \mathbf{t}_i) \end{pmatrix} \tag{16}$$
$$\mathbf{B}_{il} = \frac{\partial \mathbf{e}_{il}(\mathbf{x})}{\partial \mathbf{x}_j} = R_i^\top \tag{17}$$

And for $\frac{\partial R_i^\top}{\partial \theta_i}$ we obtain

$$\frac{\partial R_i^\top}{\partial \theta_i} = \begin{pmatrix} -\sin(\theta_i) & \cos(\theta_i) \\ -\cos(\theta_i) & -\sin(\theta_i) \end{pmatrix}. \tag{18}$$