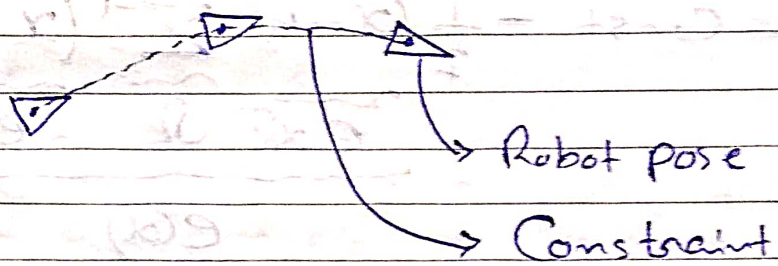
(9)

Graph Based Slam

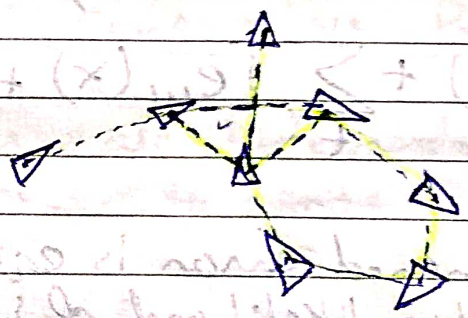
(A Least square approach to SLAM)
Using pose graph



⇒ Constraints connect the poses of the robot while it is moving.

⇒ Constraints are inherently uncertain.

⇒ Observing previously seen areas generate constraints between non successive poses.



★ Idea of Graph-Based SLAM

⇒ Use a graph to represent the problem

⇒ Every node in the graph corresponds to a pose of the robot during mapping.

⇒ Every edge between two nodes corresponds to a spatial constraint between them.

Graph based SLAM

↳ Build the graph and find a node configuration that minimize the error introduced by the constraints.

★ Graph-based slam in a Nutshell

⇒ Every node in the graph corresponds to a robot position and a laser measurement.

⇒ An edge between two nodes represents a spatial constraint between the nodes.

⇒ Once we have the graph, we determine the most likely map by correcting the nodes.

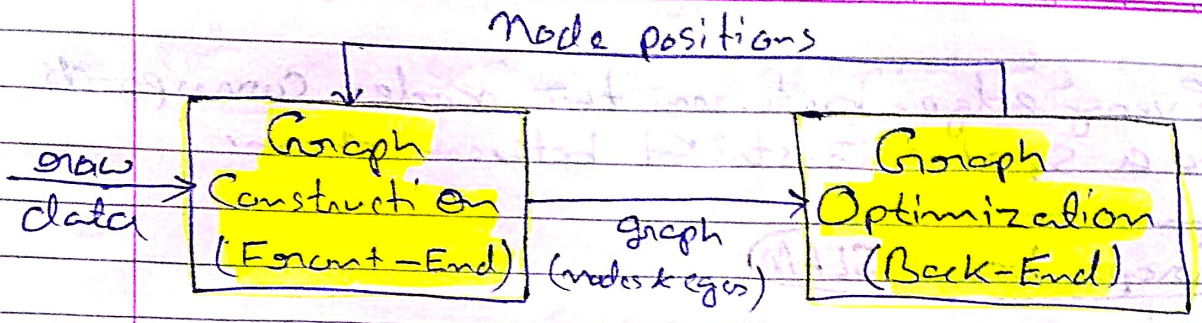
⇒ Then, we can render a map based on the known poses.

★ The overall SLAM System

⇒ Interplay of front end and backend.

Graph Construction

Graph Optimization



★ The Graph

⇒ It consists of n nodes $x = x_{1:n}$

⇒ Each x_i is a pose of the robot at time t_i

⇒ A constraint/edge exists between the nodes x_i and x_j if

① The robot moves from x_i to x_{i+1}
   ↳ Edge corresponding to odometry.

② The robot observes the same part of the environment from x_i and from x_j.
   ↳ Edge corresponding to virtual measurement.

★ Transformations

⇒ Transformations can be expressed using homogeneous coordinates.

① Odometry-Based edge

$$X_i^{-1} X_{i+1}$$

② Observation-Based edge

$$X_i^{-1} X_j$$

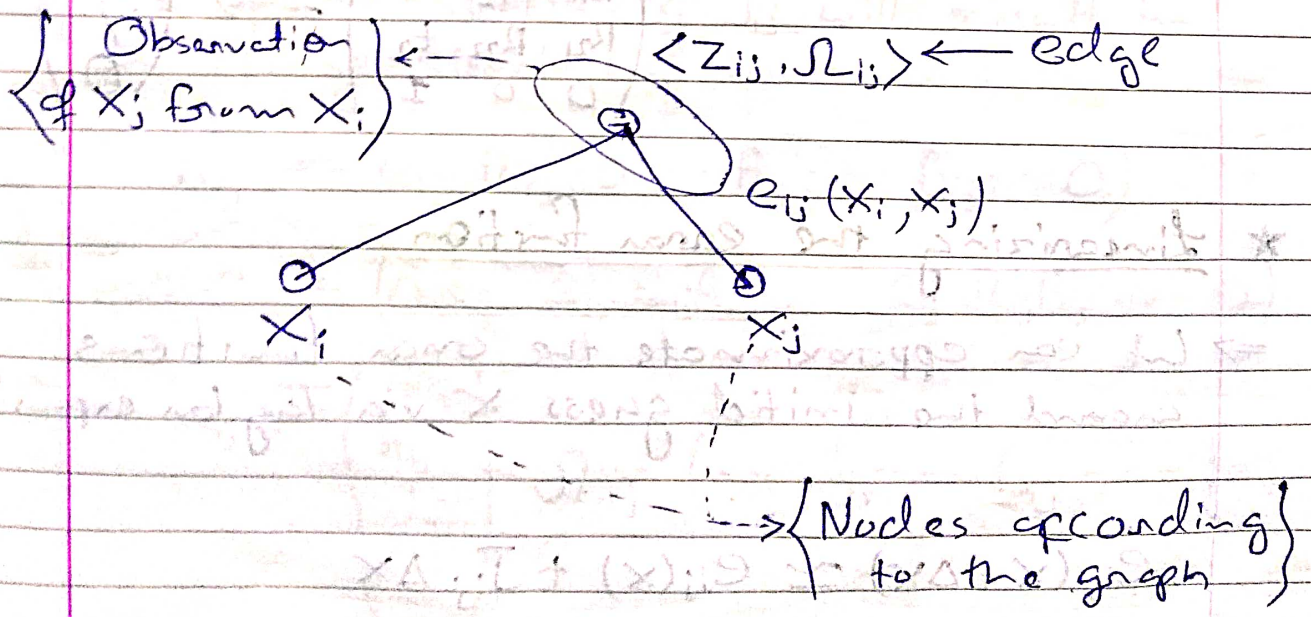
★ The Edge Information matrices

⇒ Observations are affected by noise.

⇒ Information matrix Ω_{ij} for each edge to encode its uncertainty.

⇒ The "bigger" Ω_{ij}, the more the edge "matters" in the optimization.

★ Pose Graph



Goal: $X^* = \underset{X}{\operatorname{argmin}} \sum_{ij} e_{ij}^T \Omega_{ij} e_{ij}$

★ Least Square SLAM

⇒ State vector:

$$X^T = (X_1^T \; X_2^T \; \cdots \; X_n^T)$$

★ The Error Function

⇒ Error function for a single constraint

$$e_{ij}(X_i, X_j) = t2v\left(Z_{ij}^{-1}\left(X_i^{-1} X_j\right)\right)$$

→ function that maps transformation matrix to vector

$$\begin{pmatrix} R_{11} & R_{12} & t_1 \\ R_{21} & R_{22} & t_2 \\ 0 & 0 & 1 \end{pmatrix} \longrightarrow \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$$

★ Linearizing the error function

⇒ We can approximate the error functions around the initial guess X via Taylor expansion

$$e_{ij}(X + \Delta X) \simeq e_{ij}(X) + J_{ij} \Delta X$$

$$\text{with } J_{ij} = \frac{\delta e_{ij}(X)}{\delta X}$$

$\Rightarrow$ J_{ij} will be non-zero only in the rows corresponding to x_i or x_j.

$$\frac{\delta e_{ij}(x)}{\delta x} = \left(0 \cdots \frac{\delta e_{ij}(x_i)}{\delta x_i} \cdots \frac{\delta e_{ij}(x_j)}{\delta x_j} \cdots 0\right)$$

$$J_{ij} = \left(0 \cdots A_{ij} \cdots B_{ij} \cdots 0\right)$$

★ Consequences of the Sparsity

$\Rightarrow$ We need to compute the coefficient vector b and matrix H.

$$b^T = \sum_{ij} b_{ij}^T = \sum_{ij} e_{ij}^T \Omega_{ij} J_{ij}$$

$$H = \sum_{ij} H_{ij} = \sum_{ij} J_{ij}^T \Omega_{ij} J_{ij}$$

$\Rightarrow$ The sparse structure of J_{ij} will result in a sparse structure of H.

$$b_{ij}^T = e_{ij}^T \Omega_{ij} \left(0 \cdots A_{ij} \cdots B_{ij} \cdots 0\right)$$

$$= \left(0 \cdots e_{ij}^T \Omega_{ij} A_{ij} \cdots e_{ij}^T \Omega_{ij} B_{ij} \cdots 0\right)$$

$$H_{ij} = \begin{pmatrix} \vdots \\ A_{ij}^T \\ \vdots \\ B_{ij}^T \\ \vdots \end{pmatrix} \Omega_{ij} \left(\cdots A_{ij} \cdots B_{ij} \cdots\right)$$

$$H_{ij} = \begin{pmatrix} \cdots & A_{ij}^T \Omega_{ij} A_{ij} & \cdots & A_{ij}^T \Omega_{ij} B_{ij} & \cdots \\ \cdots & B_{ij}^T \Omega_{ij} A_{ij} & \cdots & B_{ij}^T \Omega_{ij} B_{ij} & \cdots \end{pmatrix}$$

⇒ Non-zero only in the blocks relating i, j.

⇒ Efficient solvers can be used:

① Sparse Cholesky decomposition

② Conjugate gradients

etc...

★ The Linear system

⇒ The vector of the state increments:

$$\Delta x^T = (\Delta x_1^T \; \Delta x_2^T \; \cdots \; \Delta x_n^T)$$

⇒ Coefficients vector:

$$b^T = (\bar{b}_1^T \; \bar{b}_2^T \; \cdots \; \bar{b}_n^T)$$

⇒ Normal equation matrix:

$$H = \begin{pmatrix} \bar{H}^{11} & \bar{H}^{12} & \cdots & \bar{H}^{1n} \\ \bar{H}^{21} & \bar{H}^{22} & \cdots & \bar{H}^{2n} \\ \vdots & \vdots & & \vdots \\ \bar{H}^{n1} & \bar{H}^{n2} & \cdots & \bar{H}^{nn} \end{pmatrix}$$

★ Building the Linear System

⇒ For each constraint:

① Compute error

$$e_{ij} = t2v\left(Z_{ij}^{-1}\left(x_i^{-1} x_j\right)\right)$$

② Compute the blocks of the Jacobian:

$$A_{ij} = \frac{\delta e(x_i, x_j)}{\delta x_i} \qquad B_{ij} = \frac{\delta e(x_i, x_j)}{\delta x_j}$$

③ Update the coefficient vector:

$$b_i^T \mathrel{+}= e_{ij}^T \Omega_{ij} A_{ij} \qquad b_j^T \mathrel{+}= e_{ij}^T \Omega_{ij} B_{ij}$$

④ Update the normal equation matrix:

$$H^{ii} \mathrel{+}= A_{ij}^T \Omega_{ij} A_{ij} \qquad H^{ij} \mathrel{+}= A_{ij}^T \Omega_{ij} B_{ij}$$

$$H^{ji} \mathrel{+}= B_{ij}^T \Omega_{ij} A_{ij} \qquad H^{jj} \mathrel{+}= B_{ij}^T \Omega_{ij} B_{ij}$$

★ Algorithm

```
1. Optimize(x):
2.     while (!Converged)
3.         (H,b) = buildLinearSystem(x)
4.         Δx = solveSparse(HΔx = -b)
5.         x = x + Δx
6.     end
7.     return x;
```

★ Role of the Prior

⇒ Fixing the global reference frame is strongly related to the prior $P(X_0)$.

⇒ A Gaussian estimate about X_0 results in a additional constraint.

Eg: first pose in the origin:

$$e(X_0) = t2v(X_0)$$

★ Fixing a Subset of Variables

⇒ Assume that the value of certain variables during the optimization is known a priori.

↳ We may want to optimize all others and keep these fixed.

⇒ Construct the full system:

↳ Suppress the rows and the columns corresponding to the variables to fix.

★ Uncertainty

⇒ H represents the information matrix given the linearization point.

⇒ Inverting H gives the (dense) covariance matrix.

⇒ The diagonal blocks of the covariance matrix represent the uncertainties of the corresponding variables.

★ Relative Uncertainty

⇒ To determine the relative uncertainty between X_i and X_j

- Construct the full matrix H
- Suppress the rows and the columns of X_i
- Compute the block j, j of the inverse
- This block will contain the covariance matrix of X_j wrt X_i which has been fixed.
- Useful to decide loop closure.