Basic SLAM Tutorials for Everyone
Lec. #6. Filtering vs Smoothing (2): Understanding Graph Optimization from a MAP Perspective

Hyungtae Lim, Ph.D.

<shapelim [at] mit [dot] edu> / <fudxo5143+slam [at] gmail [dot] com>

Filtering vs Smoothing

Filtering (e.g., Kalman Filter)

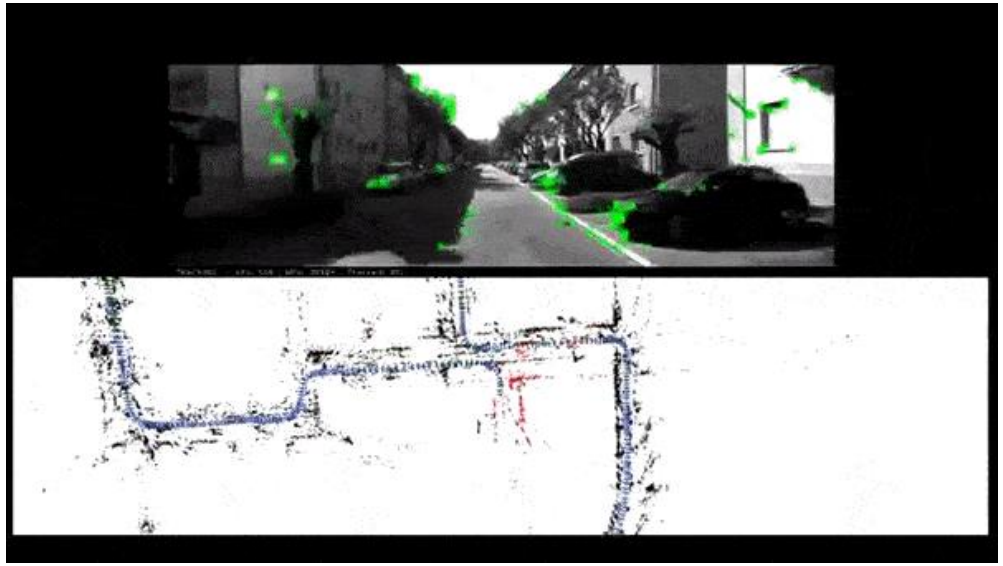
- ▶ Processes data **sequentially**.
- ▶ Maintains only current state $(\hat{x}_k, P_k)$.
- ▶ **Marginalizes** out all past states.
- ⊕ Constant memory and time per step.
- ⊖ Past states cannot be revised.
- ⊖ Linearization errors are permanent.

Smoothing (e.g., Graph SLAM)

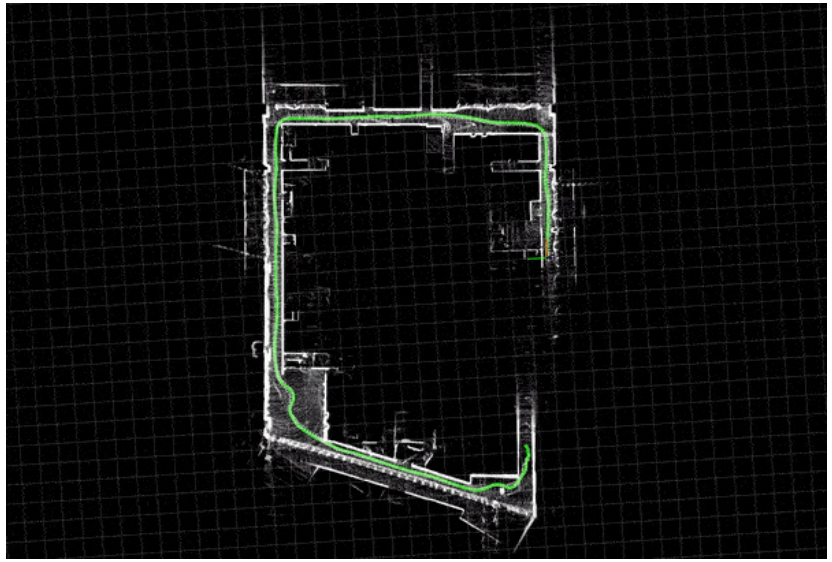
- ▶ Optimizes **all states jointly**.
- ▶ Retains the full trajectory.
- ▶ **Relinearizes** at each solver iteration.
- ⊕ Loop closures improve all poses.
- ⊕ Better accuracy for nonlinear models.
- ⊖ Cost grows with trajectory (mitigated by sparsity).

Graph SLAM: Factor Graph Optimization-Based SLAM

- Represent data in a form of graph structure
  - also called *pose graph*
- Enable to correct all the poses via *pose graph optimization*
- *Loop closing*: once revisit happens, all trajectories are corrected



Example of Camera-based SLAM
(called *Visual SLAM*)

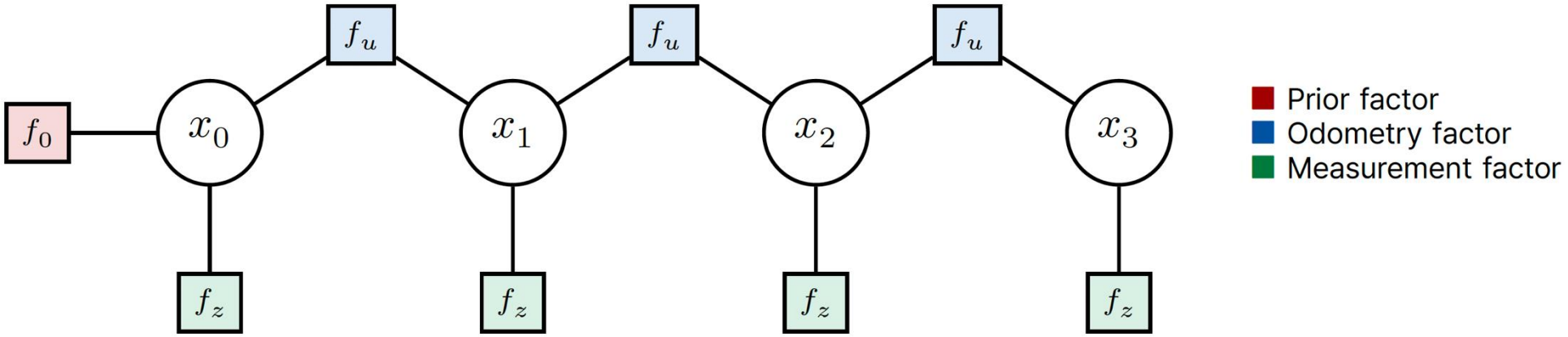


Example of LiDAR-based SLAM
(called *LiDAR SLAM*)

1D Graph Optimization: Batch Formulation (MAP)

- Instead of processing sequentially, optimize all states at once

Factor graph representation:



Batch objective over all states $\boldsymbol{x} = [x_0, x_1, \dots, x_K]^\top$:

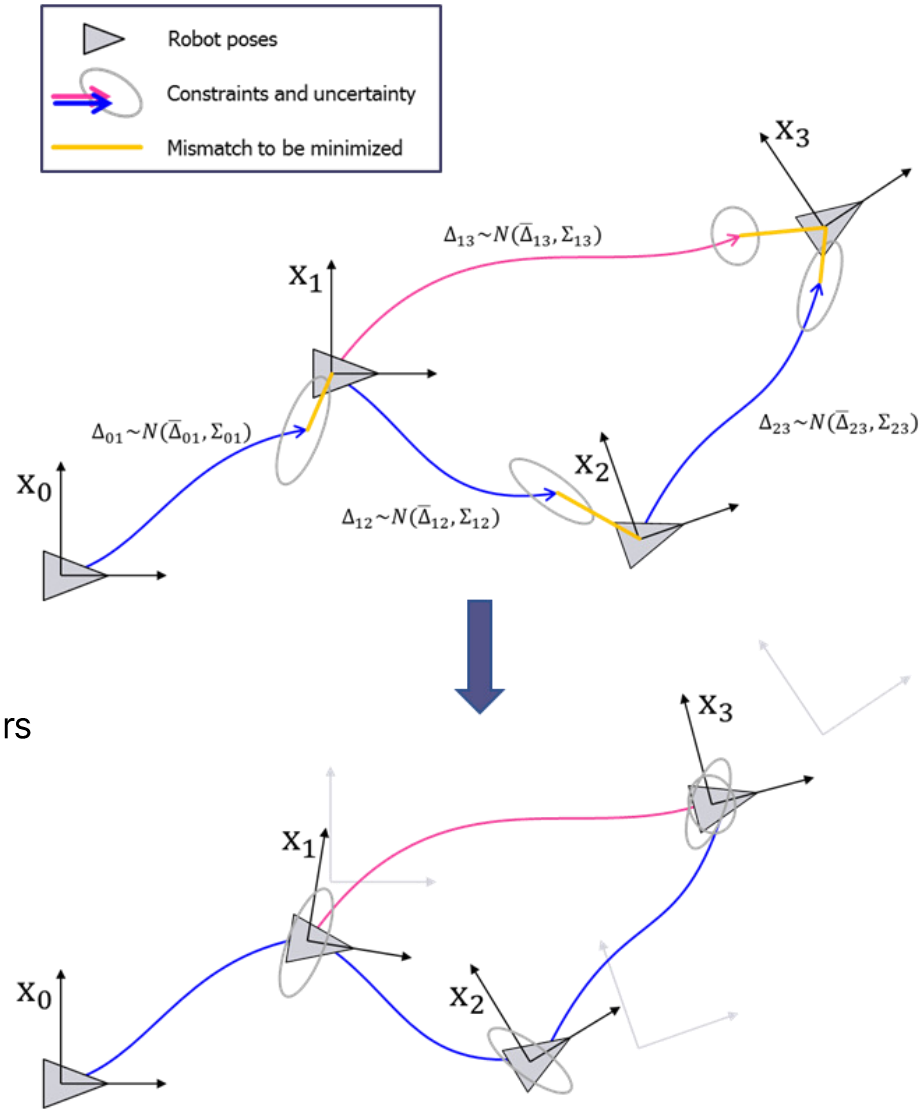
$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \underbrace{\frac{(x_0 - \mu_0)^2}{2\sigma_0^2}}_{\text{Prior}} + \underbrace{\sum_{k=1}^{K} \frac{(x_k - x_{k-1} - u_k)^2}{2\sigma_w^2}}_{\text{Odometry factors}} + \underbrace{\sum_{k=0}^{K} \frac{(z_k - x_k)^2}{2\sigma_v^2}}_{\text{Measurement factors}}$$

Procedure of Graph SLAM

- Pose graphs
  1. Represent data in a form of graph structure
     - **Nodes**
       - Poses, which is the state of interest
       - Sensor data (for loop closing)
     - **Edges** (or constraints)
       - Relative pose between two nodes
       - Consists of
         - Odometry (between $t-1$ and t frames)
         - Loop closing information using camera or LiDAR sensors (between i and j frames where $i+1 \neq j$)
  2. Define cost function

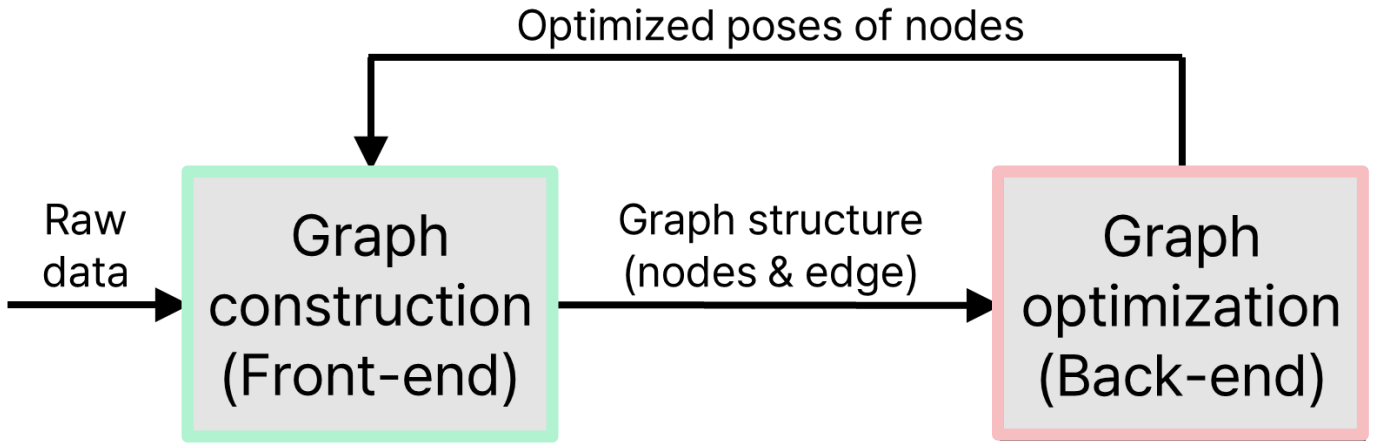
$$\mathcal{L}(\mathrm{x}) = \sum_{\{i,j\} \in P} \mathrm{r}_{ij}(\mathrm{x})^{\mathrm{T}} \Sigma_{ij}{}^{-1} \mathrm{r}_{ij}(\mathrm{x})$$

  3. Optimization
     - (weighted) least squares



Graph-Based SLAM Framework

- Graph-based SLAM
  - = Front-End (Graph construction)
  - + Back-End (Optimization)



$$\mathcal{X}^{\text{MAP}} = \underset{\mathbf{X}\in\mathcal{X}}{\operatorname{argmin}} \sum_{i} \left\| h\left(\mathbf{X}_i, \mathbf{X}_{i+1}\right) \boxminus \mathbf{z}_i^{\text{odom}} \right\|^2_{\Sigma_{\text{odom}_i}} + \sum_{(j,k)\in\mathcal{I}} \left\| s_{jk}\left(h\left(\mathbf{X}_j, \mathbf{X}_k\right) \boxminus \mathbf{z}_{jk}^{\text{loop}}\right) \right\|^2_{\Sigma_{\text{loop}_{jk}}}$$

where $j+1 \neq k$

$\mathbf{X}$: State $\in$ SE(n)

Σ: $n \times n$ covariance

$\mathcal{I}$: An indices pair by loop detection

$\mathbf{z}^{\text{odom}}$: Measurements from odometry

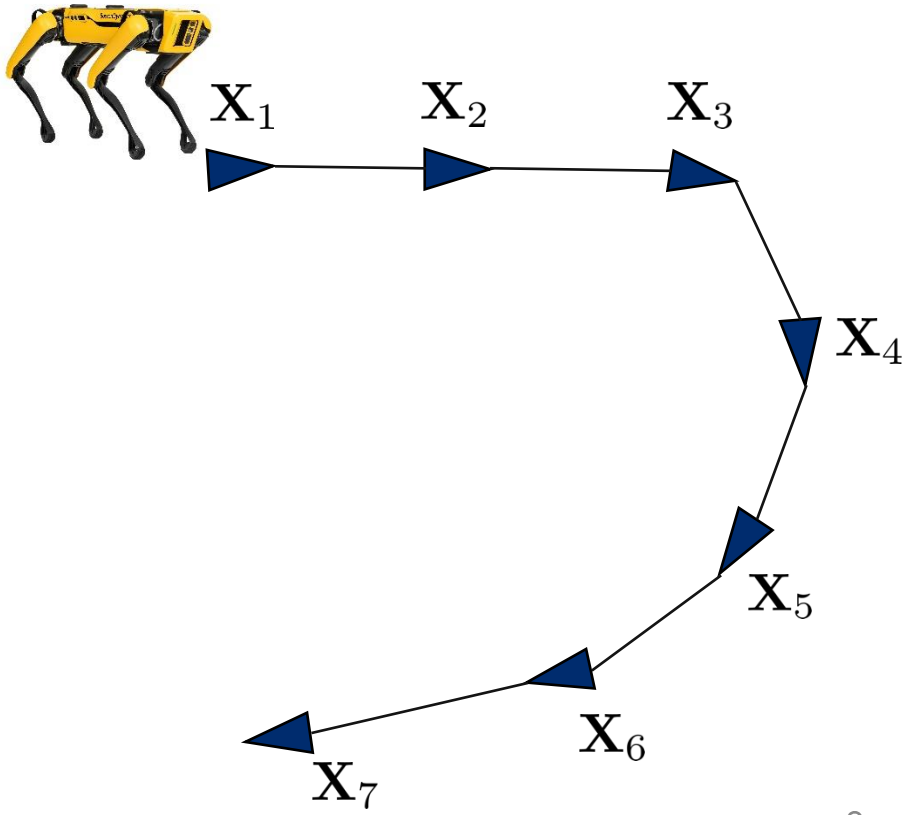
$\mathbf{z}^{\text{loop}}$: Measurements from loop closure

$s_{jk}(\cdot)$: Scaling factor (or robust kernel)

$\boxminus$: Minus operation in SE(n)

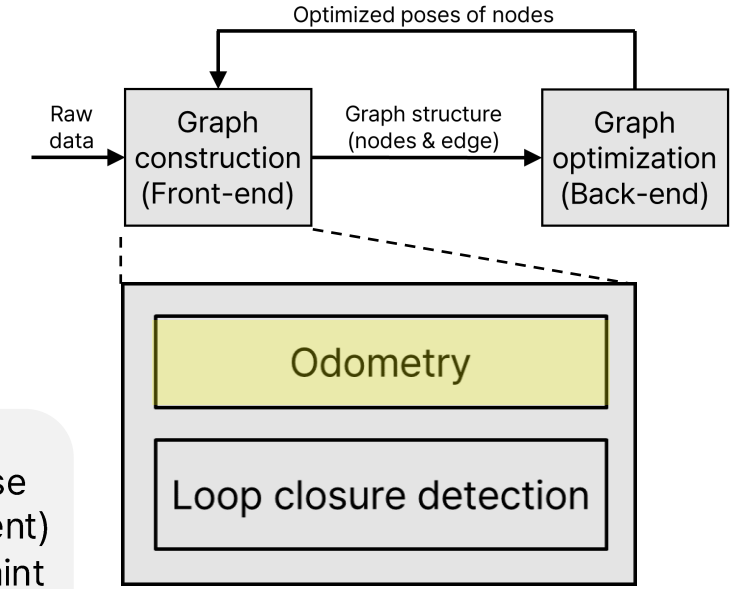
Graph-Based SLAM Framework (Cont'd)

$$\mathcal{X}^{\text{MAP}} = \underset{\mathbf{X}\in\mathcal{X}}{\operatorname{argmin}} \sum_i \left\| h\left(\mathbf{X}_i, \mathbf{X}_{i+1}\right) \boxminus \mathbf{z}_i^{\text{odom}} \right\|^2_{\Sigma_{\text{odom}_i}} + \sum_{(j,k)\in\mathcal{I}} \left\| s_{jk} \left(h\left(\mathbf{X}_j, \mathbf{X}_k\right) \boxminus \mathbf{z}_{jk}^{\text{loop}} \right) \right\|^2_{\Sigma_{\text{loop}_{jk}}}$$



Node (i.e., Robot pose w/ measurement)

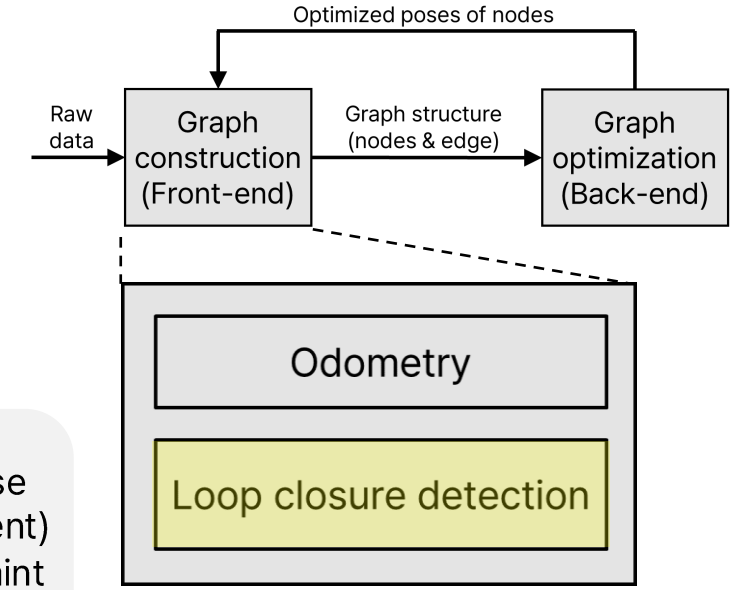
Odom. constraint

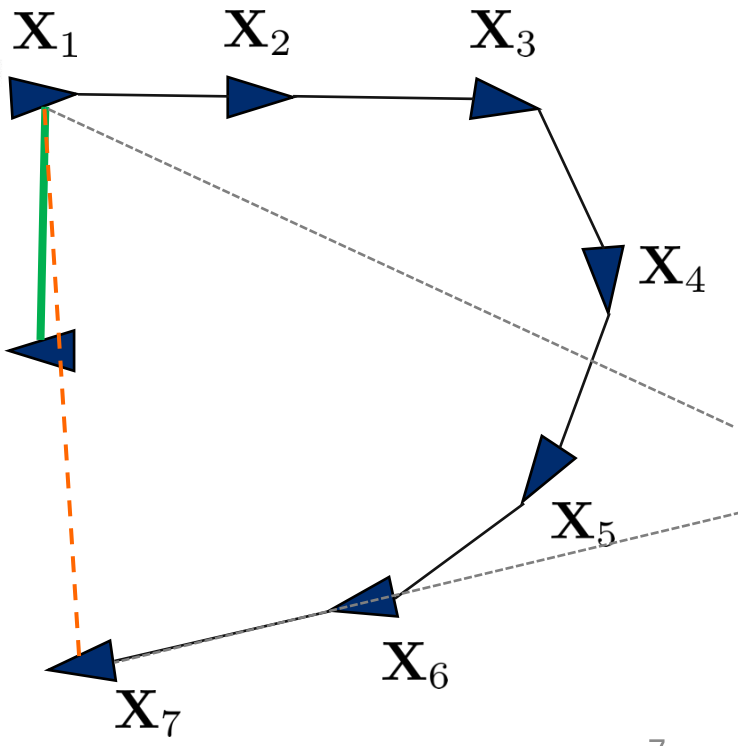


Graph-Based SLAM Framework (Cont'd)

$$\mathcal{X}^{\text{MAP}} = \underset{\mathbf{X}\in\mathcal{X}}{\operatorname{argmin}} \sum_i \left\| h\left(\mathbf{X}_i, \mathbf{X}_{i+1}\right) \boxminus \mathbf{z}_i^{\text{odom}} \right\|^2_{\Sigma_{\text{odom}_i}}$$

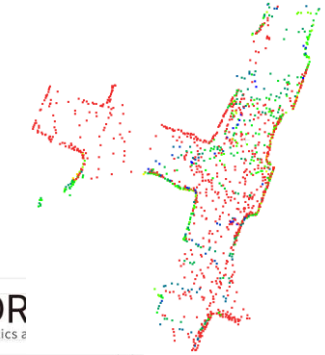
$$+ \sum_{(j,k)\in\mathcal{I}} \left\| s_{jk} \left(h\left(\mathbf{X}_j, \mathbf{X}_k\right) \boxminus \mathbf{z}_{jk}^{\text{loop}} \right) \right\|^2_{\Sigma_{\text{loop}_{jk}}}$$

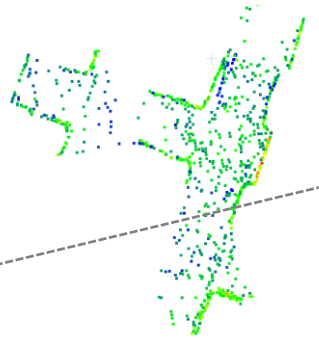




Node (i.e., Robot pose w/ measurement)
Odom. constraint
Loop constraint

Estimated $\mathbf{X}_7$ by loop closure detection





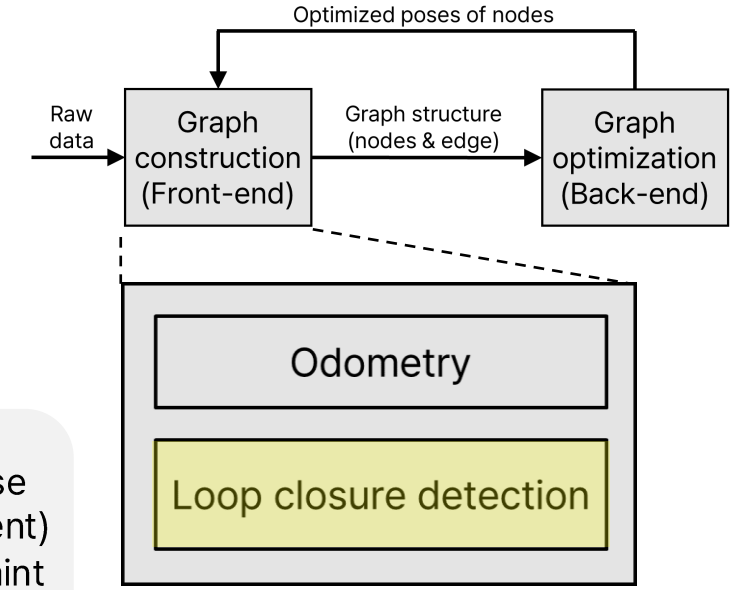
Measurement corresponding to $\mathbf{X}_1$

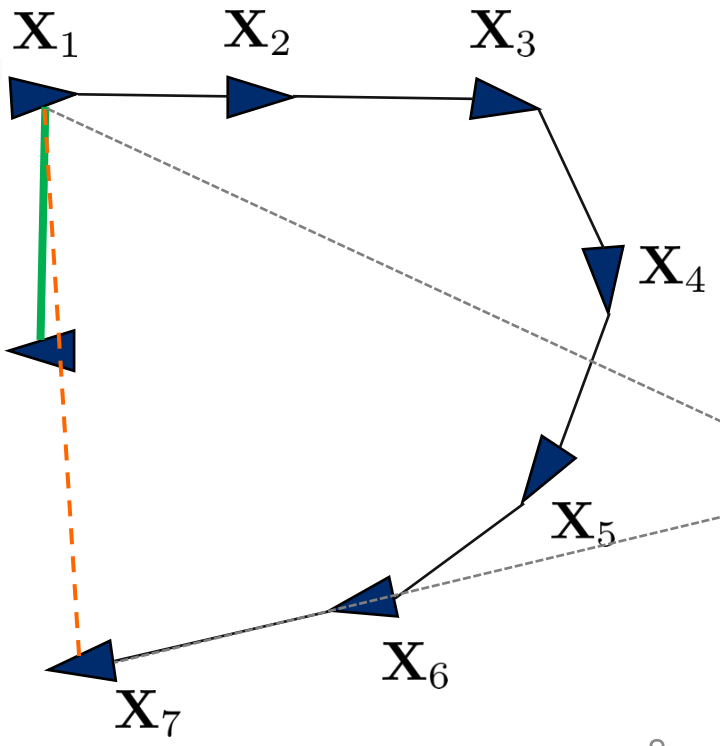
Measurement corresponding to $\mathbf{X}_7$

Graph-Based SLAM Framework (Cont'd)

$$\mathcal{X}^{\text{MAP}} = \underset{\mathbf{X}\in\mathcal{X}}{\operatorname{argmin}} \sum_i \left\| h\left(\mathbf{X}_i, \mathbf{X}_{i+1}\right) \boxminus \mathbf{z}_i^{\text{odom}} \right\|^2_{\Sigma_{\text{odom}_i}}$$

$$+ \sum_{(j,k)\in\mathcal{I}} \left\| s_{jk} \left(h\left(\mathbf{X}_j, \mathbf{X}_k\right) \boxminus \mathbf{z}_{jk}^{\text{loop}} \right) \right\|^2_{\Sigma_{\text{loop}_{jk}}}$$



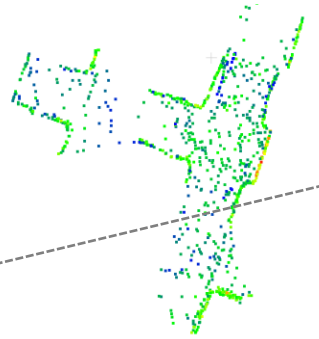


Node (i.e., Robot pose w/ measurement)

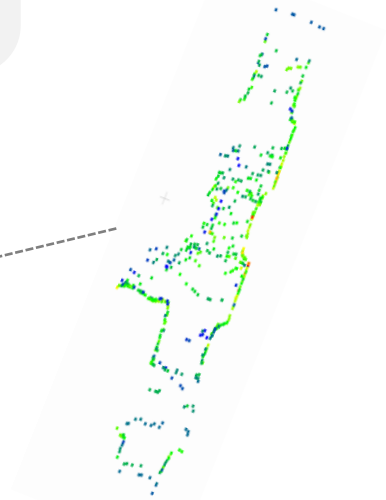
Odom. constraint

Loop constraint

Estimated $\mathbf{X}_7$ by loop closure detection



Measurement corresponding to $\mathbf{X}_1$

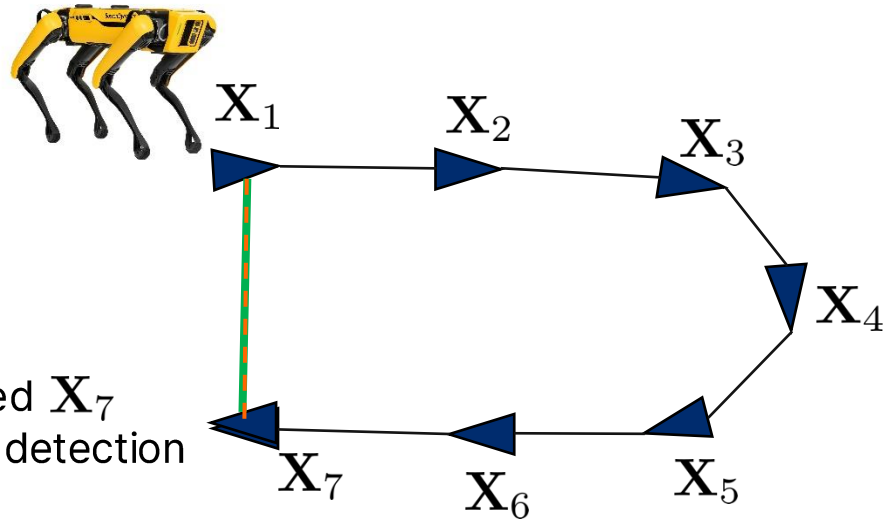


Measurement corresponding to $\mathbf{X}_7$

Graph-Based SLAM Framework (Cont'd)

$$\mathcal{X}^{\text{MAP}} = \underset{\mathbf{X}\in\mathcal{X}}{\operatorname{argmin}} \sum_i \left\| h\left(\mathbf{X}_i, \mathbf{X}_{i+1}\right) \boxminus \mathbf{z}_i^{\text{odom}} \right\|^2_{\Sigma_{\text{odom}_i}}$$

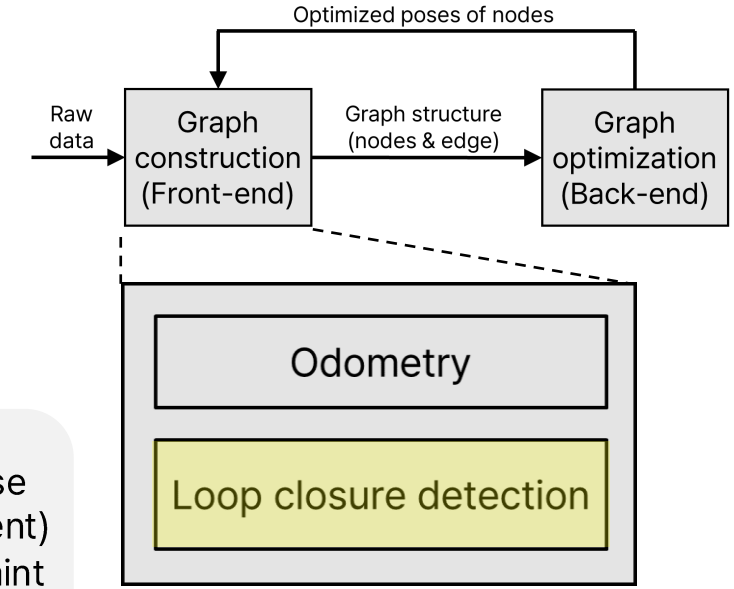
$$+ \sum_{(j,k)\in\mathcal{I}} \left\| s_{jk} \left(h\left(\mathbf{X}_j, \mathbf{X}_k\right) \boxminus \mathbf{z}_{jk}^{\text{loop}} \right) \right\|^2_{\Sigma_{\text{loop}_{jk}}}$$



Node
(i.e., Robot pose
w/ measurement)

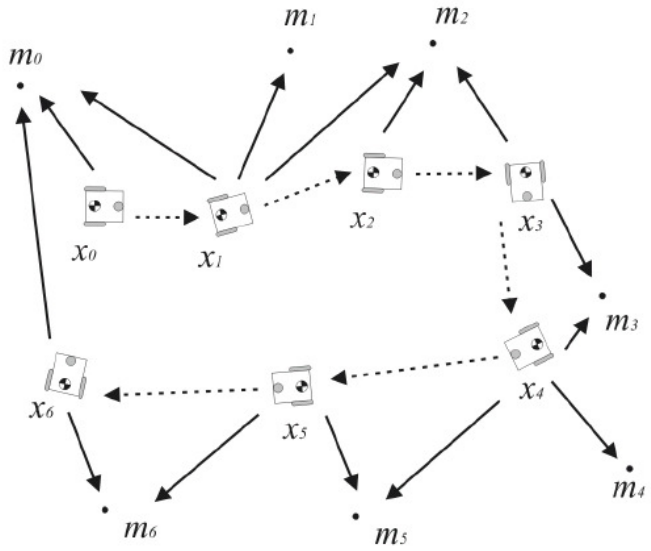
Odom. constraint

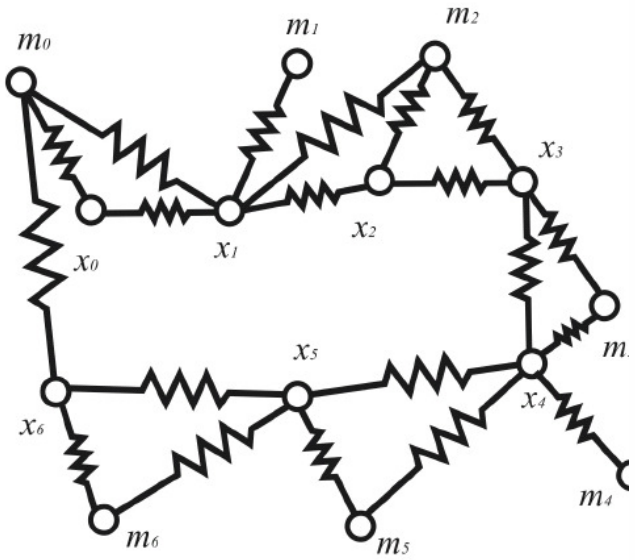
Loop constraint

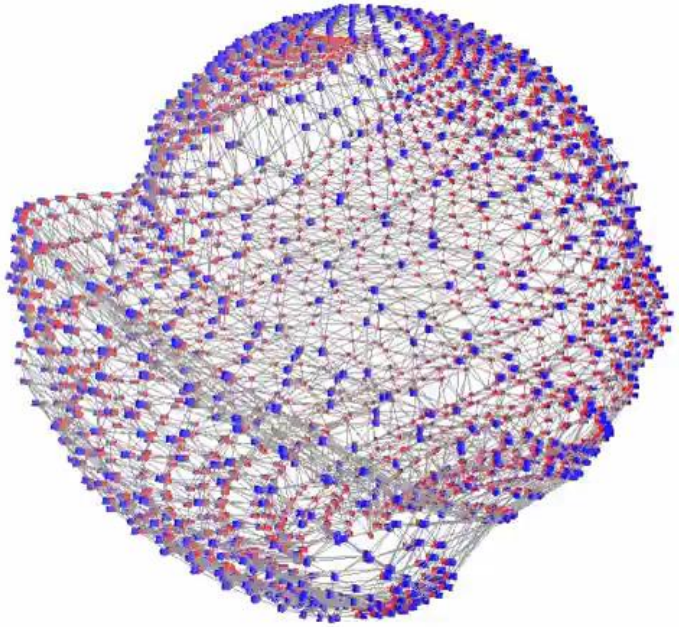


Intuitive Understanding of Graph SLAM

- It's analogous to the behavior of mass-spring systems
- With strong and weak springs connecting nodes,
  - Graph optimization is equivalent to energy minimization

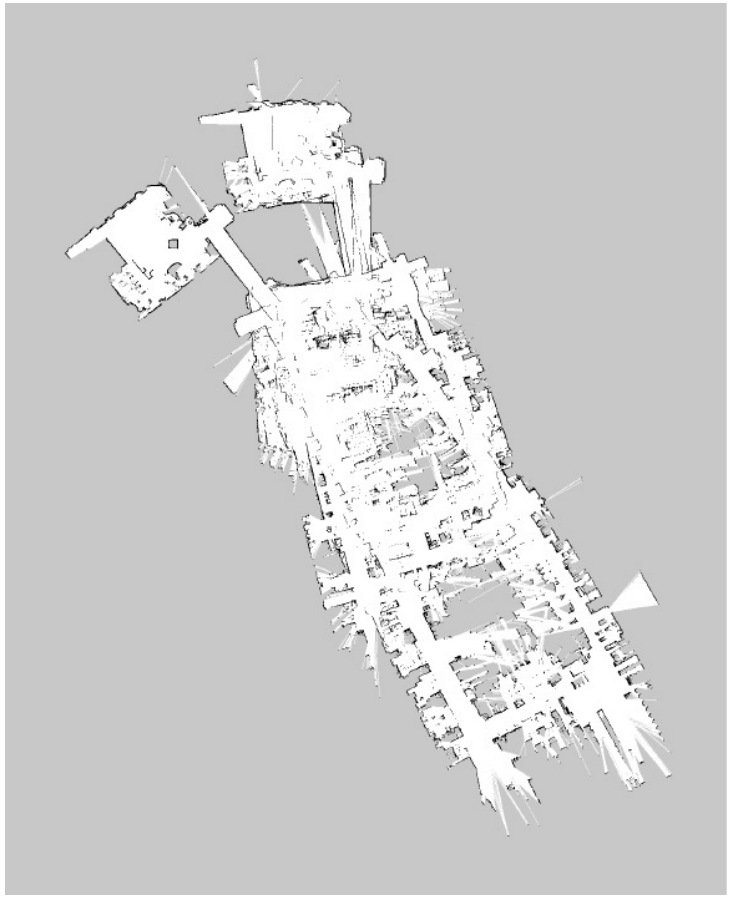






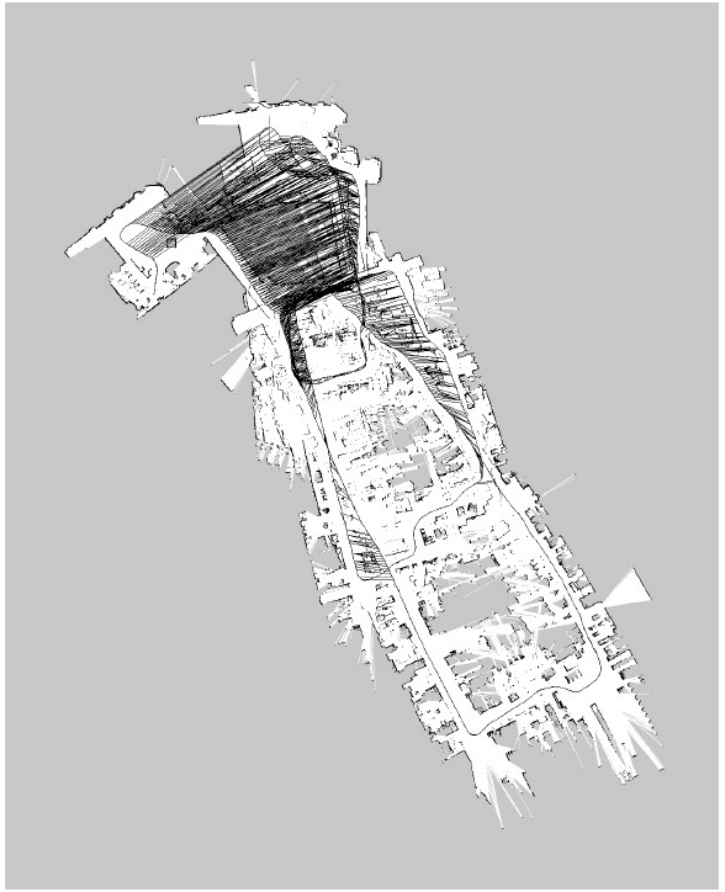
Example: Graph SLAM Using a 2D LiDAR Sensor

- In a large-scale env., drift of odometry becomes more severe
- It results in an imprecise map



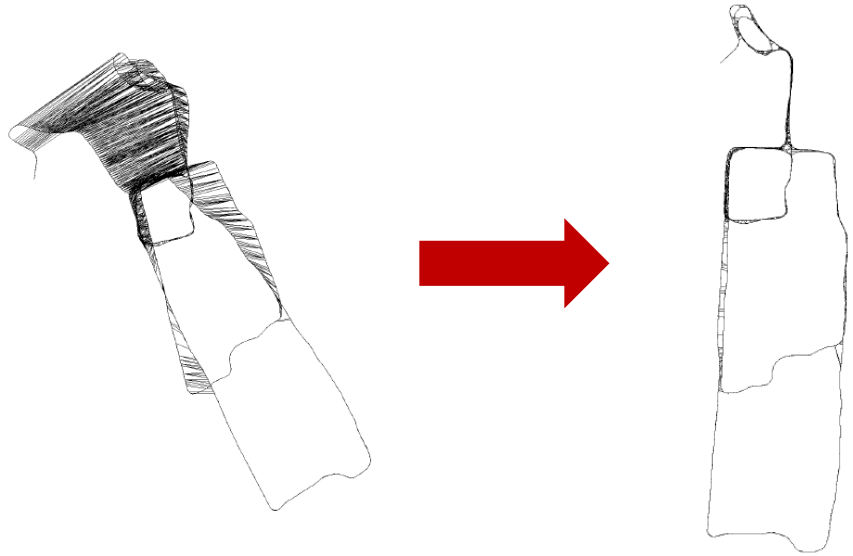
Example: Graph SLAM Using a 2D LiDAR Sensor (Cont'd)

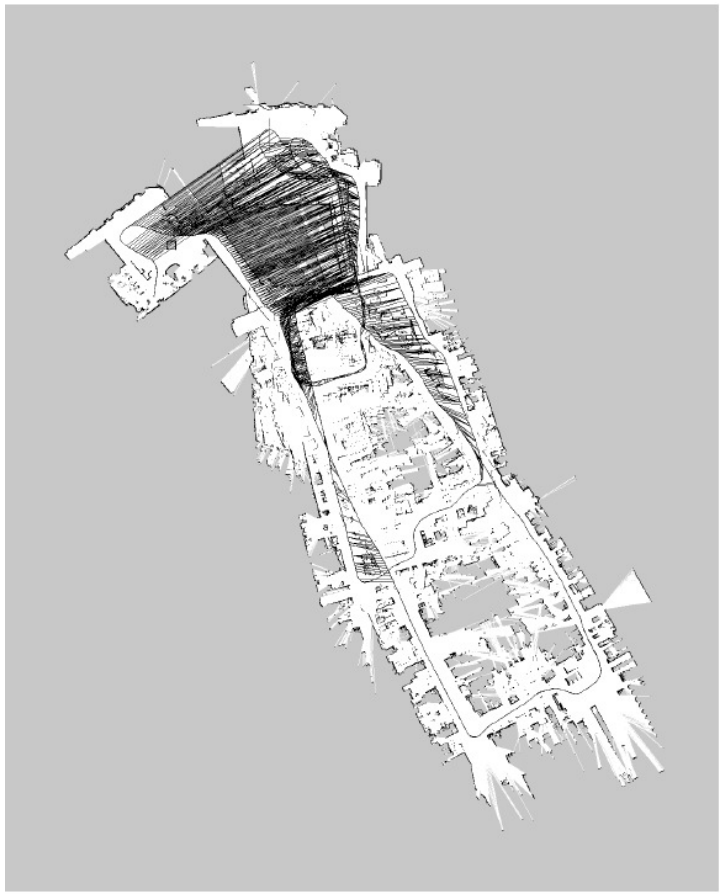
- In a large-scale env., drift of odometry becomes more severe
- It results in an imprecise map
- Add *strong springs* by checking similarity of acquired data
  - high similarity → high prob. of revisit



Example: Graph SLAM Using a 2D LiDAR Sensor (Cont'd)

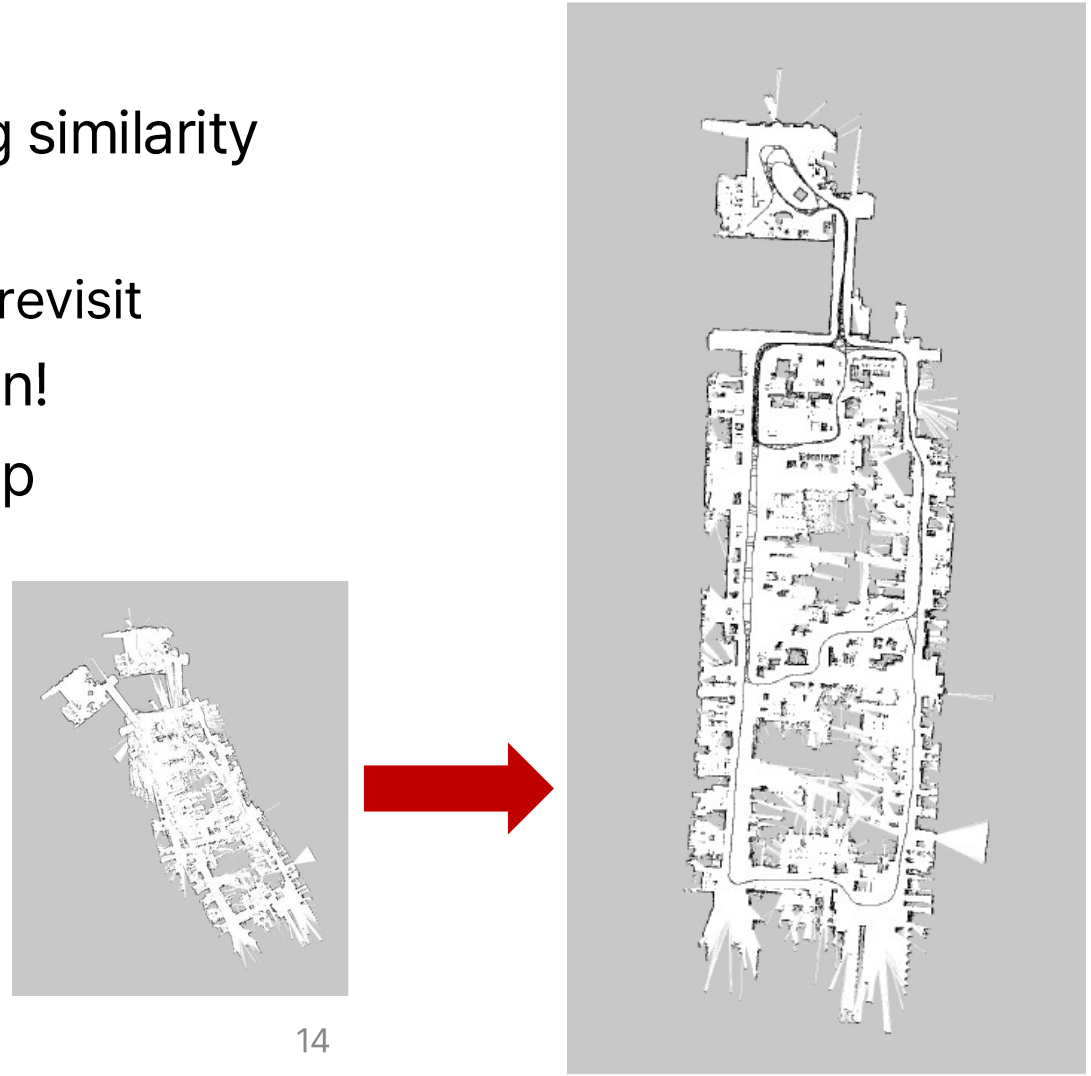
- In a large-scale env., drift of odometry becomes more severe
- It results in an imprecise map
- Add *strong springs* by checking similarity of acquired data
  - high similarity → high prob. of revisit
- Then, minimize the cost function!





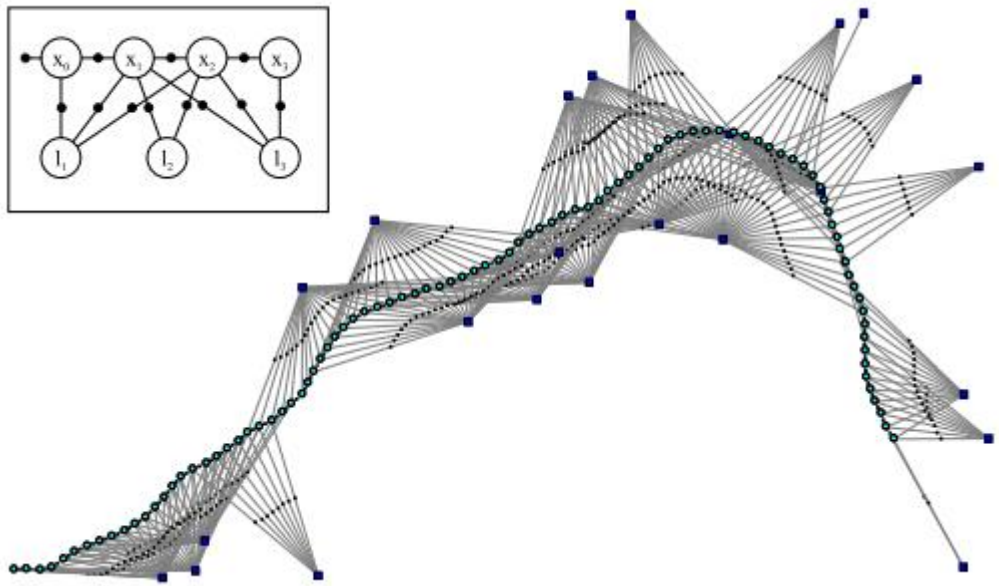
Example: Graph SLAM Using a 2D LiDAR Sensor (Cont'd)

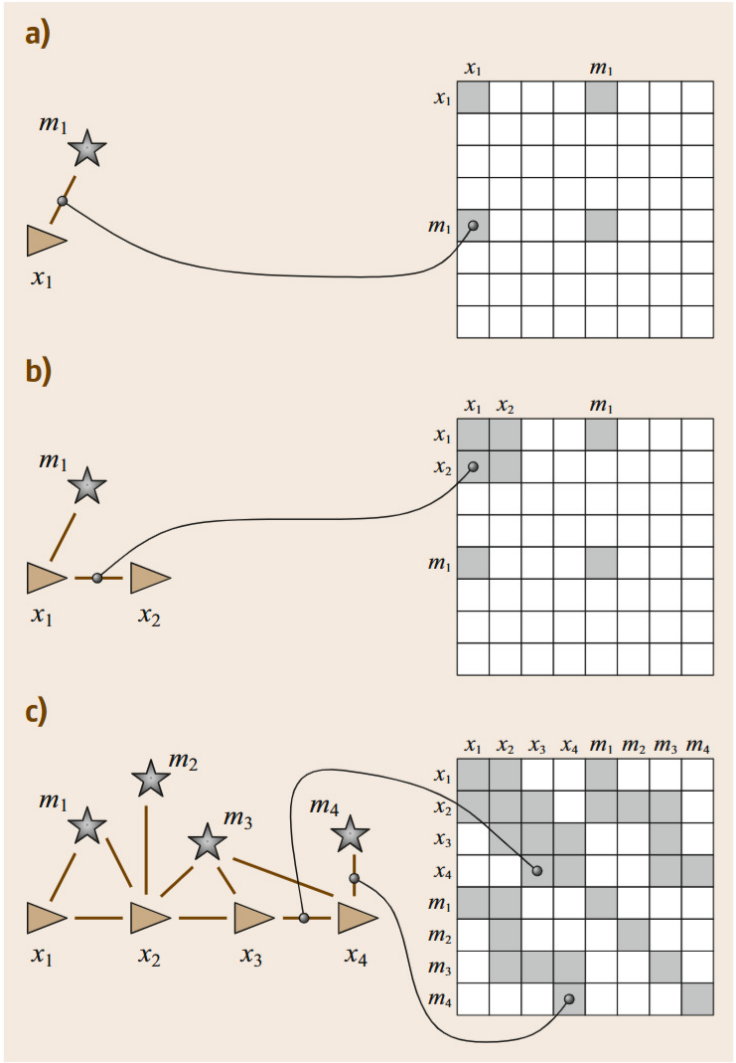
- In a large-scale env., drift of odometry becomes more severe
- It results in an imprecise map
- Add *strong springs* by checking similarity of acquired data
  - high similarity → high prob. of revisit
- Then, minimize the cost function!
- After optimization, rebuild a map using optimized poses



14

Graph SLAM Can Also Contain *Landmarks*





Cyrill Stachniss, John J. Leonard, Sebastian Thrun, Simultaneous Localizationand Mapping, 2016: https://www.cs.columbia.edu/~allen/F19/NOTES/slam_paper.pdf

Conclusion

Filtering (e.g., Kalman Filter)

- ▶ Processes data **sequentially**.
- ▶ Maintains only current state $(\hat{x}_k, P_k)$.
- ▶ **Marginalizes** out all past states.
- ⊕ Constant memory and time per step.
- ⊖ Past states cannot be revised.
- ⊖ Linearization errors are permanent.

Smoothing (e.g., Graph SLAM)

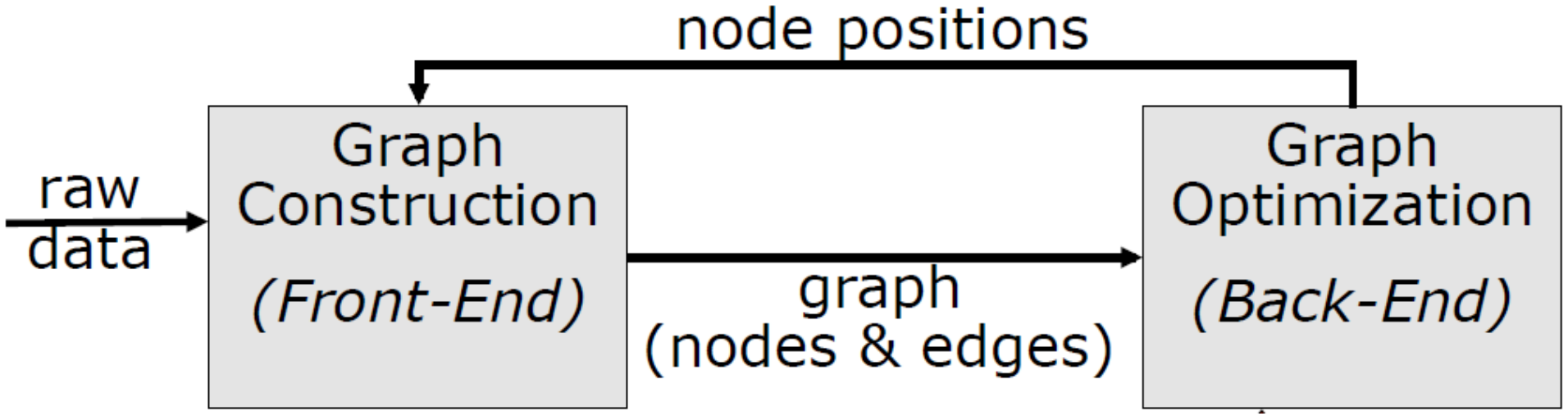
- ▶ Optimizes **all states jointly**.
- ▶ Retains the full trajectory.
- ▶ **Relinearizes** at each solver iteration.
- ⊕ Loop closures improve all poses.
- ⊕ Better accuracy for nonlinear models.
- ⊖ Cost grows with trajectory (mitigated by sparsity).

Marginalization is irreversible.

Once a past state is marginalized out, new information (e.g., a loop closure connecting the current pose back to that state) **cannot improve it**. Smoothing avoids this fundamental limitation.

Conclusion (Cont'd)

- Graph-based SLAM algorithm consists of two parts
  - Front-end (Graph construction)
    - a) Odometry: pose estimation between *consecutive* frames
    - b) loop closing: pose estimation between *non-consecutive* frames
  - Back-end (Graph optimization)



Q. Which One is Better?

The filtering camp cruise

Stergios I. Roumeliotis
Prof. @ Univ. Minnesota
→ Apple vision pro team
: Author of MSCKF

Guoquan (Paul) Huang
Prof. @ Univ. Delaware
: Author of OpenVINS

vs.

The smoothing camp nuro

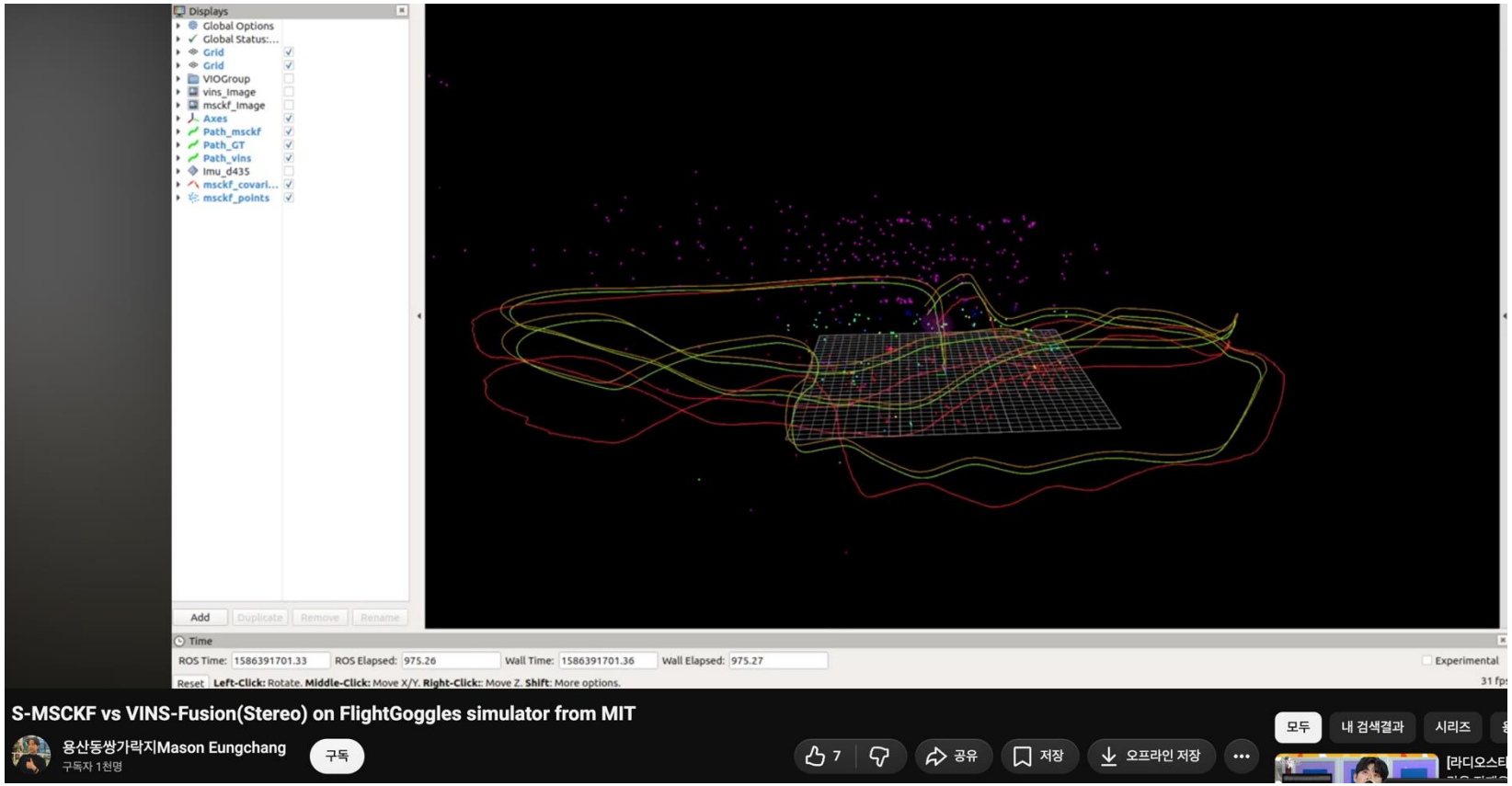
Frank Dellaert
prof. @ GeorgiaTech
: GTSAM

Luca Carlone
prof. @ MIT

Cyrill Stachniss
prof. @ Univ. Bonn

Q. Which One is Better? (Cont'd)

- In terms of *performance*: graph optimization-based approaches are better
- In terms of *computational cost*: filtering approaches are more lightweight



https://www.youtube.com/watch?v=s_OI-k8rhwY