ECE276A: Sensing & Estimation in Robotics
Lecture 5: Factor Graph SLAM

Nikolay Atanasov

natanasov@ucsd.edu

UC San Diego
JACOBS SCHOOL OF ENGINEERING
Electrical and Computer Engineering

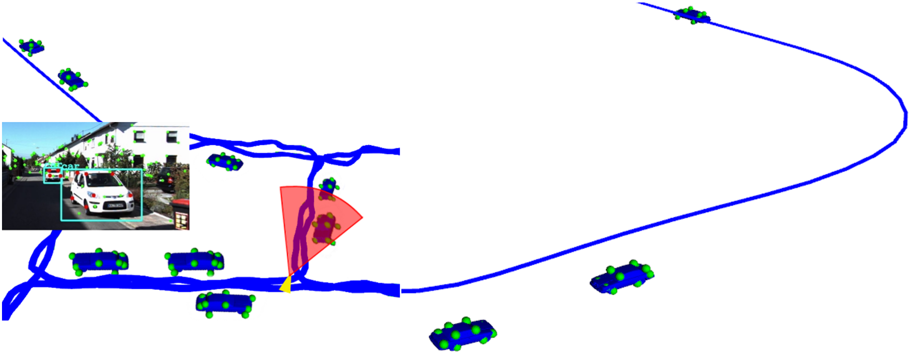
Outline

Introduction to SLAM

Factor Graph SLAM

Simultaneous Localization and Mapping (SLAM)

- SLAM is a fundamental problem for mobile robot autonomy
- Basic information necessary to perform any robot task:
  - Where am I? $\Rightarrow$ **Localization**
  - What is around me? $\Rightarrow$ **Mapping**
- SLAM problem: given sensor measurements $\mathbf{z}_{0:T}$ (e.g., images) and control inputs $\mathbf{u}_{0:T-1}$ (e.g., velocity), estimate the robot state trajectory $\mathbf{x}_{0:T}$ (e.g., pose) and build a map $\mathbf{m}$ of the environment



Mathematical Formulation of SLAM Problems

- **Mapping**: given robot state trajectory $\mathbf{x}_{0:T}$ and sensor measurements $\mathbf{z}_{0:T}$ with observation model h, build a map $\mathbf{m}$ of the environment

$$\min_{\mathbf{m}} \sum_{t=0}^{T} \|\mathbf{z}_t - h(\mathbf{x}_t, \mathbf{m})\|_2^2$$

- **Localization**: given a map $\mathbf{m}$ of the environment, sensor measurements $\mathbf{z}_{0:T}$ with observation model h, and control inputs $\mathbf{u}_{0:T-1}$ with motion model f, estimate the robot state trajectory $\mathbf{x}_{0:T}$

$$\min_{\mathbf{x}_{0:T}} \sum_{t=0}^{T} \|\mathbf{z}_t - h(\mathbf{x}_t, \mathbf{m})\|_2^2 + \sum_{t=0}^{T-1} \|\mathbf{x}_{t+1} - f(\mathbf{x}_t, \mathbf{u}_t)\|_2^2$$

- **SLAM**: given initial robot state $\mathbf{x}_0$, sensor measurements $\mathbf{z}_{1:T}$ with observation model h, and control inputs $\mathbf{u}_{0:T-1}$ with motion model f, estimate the robot state trajectory $\mathbf{x}_{1:T}$ and build a map $\mathbf{m}$

$$\min_{\mathbf{x}_{1:T}, \mathbf{m}} \sum_{t=1}^{T} \|\mathbf{z}_t - h(\mathbf{x}_t, \mathbf{m})\|_2^2 + \sum_{t=0}^{T-1} \|\mathbf{x}_{t+1} - f(\mathbf{x}_t, \mathbf{u}_t)\|_2^2$$

Example: Localization with Linear Models

- State: $\mathbf{x}_t \in \mathbb{R}^n$
- Motion model: $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) = F\mathbf{x}_t + G\mathbf{u}_t$
- Observation model: $\mathbf{z}_t = h(\mathbf{x}_t) = H\mathbf{x}_t$
- Localization: given $\mathbf{x}_0 = \mathbf{0}$, sensor measurements $\mathbf{z}_{1:T}$, and control inputs $\mathbf{u}_{0:T-1}$, estimate the state trajectory $\mathbf{x}_{1:T}$

$$\min_{\mathbf{x}_{1:T}} c(\mathbf{x}_{1:T}) := \sum_{t=1}^{T} \|\mathbf{z}_t - H\mathbf{x}_t\|_2^2 + \sum_{t=0}^{T-1} \|\mathbf{x}_{t+1} - F\mathbf{x}_t - G\mathbf{u}_t\|_2^2$$

- Gradient descent: initialize $\mathbf{x}_{1:T}^{(0)}$ and iterate:

$$\mathbf{x}_{1:T}^{(k+1)} = \mathbf{x}_{1:T}^{(k)} - \alpha^{(k)} \nabla c(\mathbf{x}_{1:T}^{(k)})$$

Example: Localization with Linear Models

- $\left\| \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \right\|_2^2 = \|x_1 - y_1\|_2^2 + \|x_2 - y_2\|_2^2$ for $x_1, y_1 \in \mathbb{R}^{d_1}$, $x_2, y_2 \in \mathbb{R}^{d_2}$
- Express the least-squares localization problem in matrix notation:

$$
\begin{aligned}
c(\mathbf{x}_{1:T}) &= \sum_{t=1}^{T} \|\mathbf{z}_t - H\mathbf{x}_t\|_2^2 + \sum_{t=0}^{T-1} \|\mathbf{x}_{t+1} - F\mathbf{x}_t - G\mathbf{u}_t\|_2^2 \\
&= \left\| \begin{bmatrix} \mathbf{z}_1 - H\mathbf{x}_1 \\ \vdots \\ \mathbf{z}_T - H\mathbf{x}_T \end{bmatrix} \right\|_2^2 + \left\| \begin{bmatrix} \mathbf{x}_1 - F\mathbf{x}_0 - G\mathbf{u}_0 \\ \vdots \\ \mathbf{x}_T - F\mathbf{x}_{T-1} - G\mathbf{u}_{T-1} \end{bmatrix} \right\|_2^2 \\
&= \left\| \begin{bmatrix} H & & \\ & \ddots & \\ & & H \end{bmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_T \end{pmatrix} - \begin{bmatrix} \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_T \end{bmatrix} \right\|_2^2 + \left\| \begin{bmatrix} -I & & & \\ F & \ddots & & \\ & \ddots & \ddots & \\ & & F & -I \end{bmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_T \end{pmatrix} + \begin{bmatrix} F\mathbf{x}_0 + G\mathbf{u}_0 \\ G\mathbf{u}_1 \\ \vdots \\ G\mathbf{u}_{T-1} \end{bmatrix} \right\|_2^2
\end{aligned}
$$

Example: Localization with Linear Models

- Objective:

$$c(\mathbf{x}_{1:T}) = \left\| \begin{bmatrix} H & & & \\ & \ddots & & \\ & & \ddots & \\ & & & H \\ -I & & & \\ F & \ddots & & \\ & \ddots & \ddots & \\ & & F & -I \end{bmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_T \end{pmatrix} + \begin{bmatrix} -\mathbf{z}_1 \\ \vdots \\ -\mathbf{z}_T \\ F\mathbf{x}_0 + G\mathbf{u}_0 \\ G\mathbf{u}_1 \\ \vdots \\ G\mathbf{u}_{T-1} \end{bmatrix} \right\|_2^2$$

$$= \|A\mathbf{x}_{1:T} + \mathbf{b}\|_2^2$$

- Gradient:

$$\nabla c(\mathbf{x}_{1:T}) = 2A^\top(A\mathbf{x}_{1:T} + \mathbf{b})$$

- Gradient descent: initialize $\mathbf{x}_{1:T}^{(0)}$ and iterate:

$$\mathbf{x}_{1:T}^{(k+1)} = \mathbf{x}_{1:T}^{(k)} - 2\alpha^{(k)} A^\top(A\mathbf{x}_{1:T}^{(k)} + \mathbf{b})$$

Project 1: Orientation Tracking

- Consider a rigid body undergoing pure rotation
- **State**: orientation $\mathbf{q}_t \in \mathbb{H}_*$ of the body frame relative to the world frame
- **Control**: body-frame angular velocity $\mathbf{u}_t \in \mathbb{R}^3$ obtained from gyroscope measurements in rad/sec during time interval τ_t
- **Motion model**: $\mathbf{q}_{t+1} = f(\mathbf{q}_t, \tau_t \mathbf{u}_t) := \mathbf{q}_t \circ \exp([0,\ \tau_t \mathbf{u}_t/2])$
- **Observation model**: body-frame acceleration $\mathbf{z}_t \in \mathbf{R}^3$ obtained from accelerometer measurements in m/sec^2 should approximately match the world-frame gravity acceleration $-g\mathbf{e}_3$:

$$\mathbf{z}_t = h(\mathbf{q}_t) := \mathbf{q}_t^{-1} \circ [0,\ -g\mathbf{e}_3] \circ \mathbf{q}_t$$

Project 1: Orientation Tracking

- Starting with $\mathbf{q}_0 = [1, \mathbf{0}] \in \mathbb{H}_*$, formulate an optimization problem to estimate $\mathbf{q}_{1:T}$ using the gyroscope inputs $\mathbf{u}_{0:T-1}$ and accelerometer measurements $\mathbf{z}_{1:T}$

- **Distance on $\mathbb{H}_*$**: the distance between two quaternions $\mathbf{q}_1, \mathbf{q}_2 \in \mathbb{H}_*$ can be measured by the rotation angle $\|\boldsymbol{\theta}_{12}\|_2$ of the axis-angle representation $\boldsymbol{\theta}_{12}$ of the relative rotation $\mathbf{q}_{12} = \mathbf{q}_1^{-1}\mathbf{q}_2$:

$$d(\mathbf{q}_1, \mathbf{q}_2) = \|\boldsymbol{\theta}_{12}\|_2 = \|2\log(\mathbf{q}_1^{-1}\mathbf{q}_2)\|_2$$

- We formulate a **constrained** optimization problem because we require that $\mathbf{q}_t$ is a valid orientation, i.e., $\mathbf{q}_t \in \mathbb{H}_*$:

$$\begin{aligned} \min_{\mathbf{q}_{1:T}} \quad & c(\mathbf{q}_{1:T}) := \sum_{t=1}^{T} \|\mathbf{z}_t - h(\mathbf{q}_t)\|_2^2 + \sum_{t=0}^{T-1} \|2\log\left(\mathbf{q}_{t+1}^{-1} \circ f(\mathbf{q}, \tau_t \mathbf{u}_t)\right)\|_2^2 \\ \text{s.t.} \quad & \|\mathbf{q}_t\|_2 = 1, \;\; \forall t \end{aligned}$$

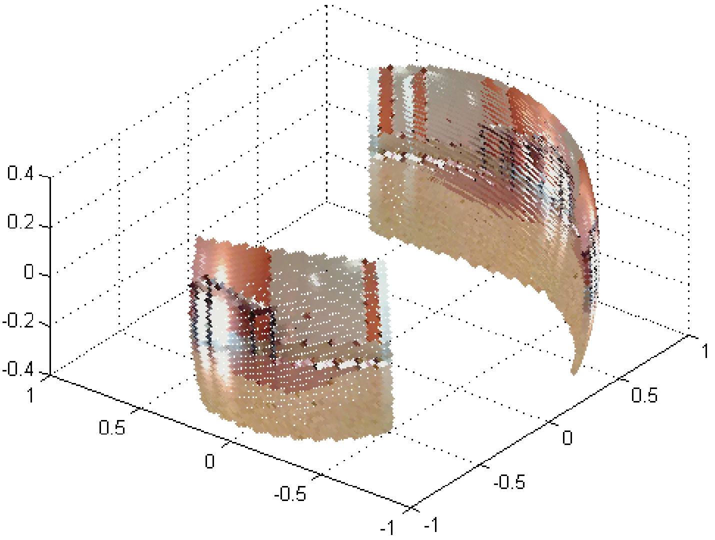
- **Possible approach**: projected gradient descent

$$\mathbf{q}_{1:T}^{(k+1)} = \Pi_{\mathbb{H}_*}\left(\mathbf{q}_{1:T}^{(k)} - \alpha^{(k)} \nabla c(\mathbf{q}_{1:T}^{(k)})\right)$$

Project 1: Panorama

- **Input**: image I and camera-to-world orientation R
- Suppose the image lies on a sphere and compute the world coordinates of each pixel:
  1. Find longitude (λ) and latitude (ϕ) of each pixel using the number of rows and columns and the horizontal (60°) and vertical (45°) fields of view
  2. Convert spherical $(\lambda, \phi, 1)$ to Cartesian coordinates assuming depth 1
  3. Rotate the Cartesian coordinates to the world frame using R
- Project world pixel coordinates to a cylinder and unwrap:
  1. Convert Cartesian to spherical coordinates
  2. Inscribe the sphere in a cylinder so that a point $(\lambda, \phi, 1)$ on the sphere has height ϕ on the cylinder and longitude λ along the cylinder circumference
  3. Unwrap the cylinder surface to a rectangular image with width 2π radians and height π radians
  4. Different options for sphere to plane projection: equidistant, equal area, Miller, etc. (see `https://en.wikipedia.org/wiki/List_of_map_projections`)

Project 1: Panorama

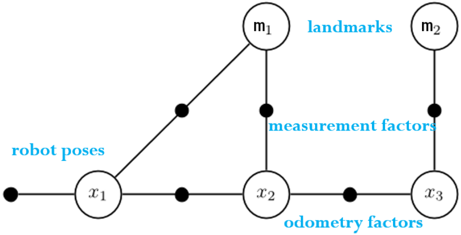


Outline

Introduction to SLAM

Factor Graph SLAM

Factor Graph



- **Factor graph**: bipartite graph describing data (observations $\mathbf{z}_t$, inputs $\mathbf{u}_t$) and variables (states $\mathbf{x}_t$, landmarks $\mathbf{m}_j$) in a SLAM problem

- **Nodes**: variables to be estimated: robot states $\mathbf{x}_t$ and landmark states $\mathbf{m}_j$

- **Factors**: relate two variables by input $\mathbf{u}_t$ or observation $\mathbf{z}_t$ data and associated motion or observation model:
  - Motion factor: error between state $\mathbf{x}_{t+1}$ and its motion prediction $f(\mathbf{x}_t, \mathbf{u}_t)$:

$$\mathbf{e}_f(\mathbf{x}_{t+1}, \mathbf{x}_t) = \mathbf{x}_{t+1} \ominus f(\mathbf{x}_t, \mathbf{u}_t)$$

  - Observation factor: error between observation $\mathbf{z}_{t,j}$ and its prediction $h(\mathbf{x}_t, \mathbf{m}_j)$

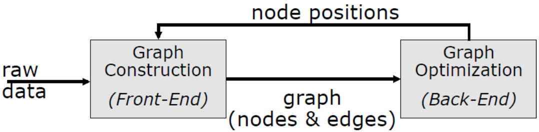
$$\mathbf{e}_h(\mathbf{x}_t, \mathbf{m}_j) = \mathbf{z}_{t,j} \ominus h(\mathbf{x}_t, \mathbf{m}_j)$$

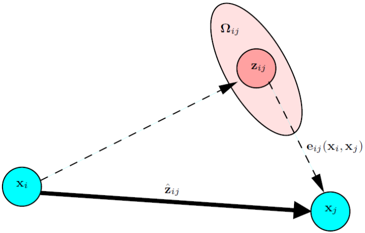
  - We use the symbol $\ominus$ to indicate that the difference between two variable should respect the geometry of their space, e.g., $\mathbf{y} \ominus \mathbf{x} = \mathbf{y} - \mathbf{x}$ for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ but $\mathbf{y} \ominus \mathbf{x} = 2\log(\mathbf{x}^{-1}\mathbf{y})$ for $\mathbf{x}, \mathbf{y} \in \mathbb{H}_*$

Factor Graph SLAM

- **Front-end**: construction of factor graph using odometry, laser-scan matching, feature matching, etc.

- **Back-end**: graph optimization to estimate the variables $(\mathbf{x}_{0:T}, \{\mathbf{m}_j\})$



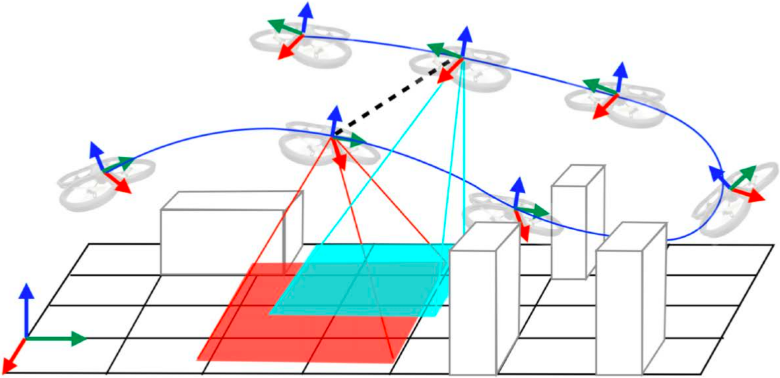


- **Back-end optimization problem** with variables $\mathbf{x}_i$ associated with the graph vertices $i \in \mathcal{V}$ and factors $\mathbf{e}(\mathbf{x}_i, \mathbf{x}_j)$ associated with the graph edges $(i,j) \in \mathcal{E}$:

$$\min_{\{\mathbf{x}_i\}} \sum_{(i,j)\in\mathcal{E}} \phi_{ij}(\mathbf{e}(\mathbf{x}_i, \mathbf{x}_j))$$

where $\phi_{ij} : \mathbb{R}^d \mapsto \mathbb{R}$ is a distance function, e.g., $\phi_{ij}(\mathbf{e}) = \mathbf{e}^\top \Omega_{ij} \mathbf{e}$ with positive-definite Ω_{ij}

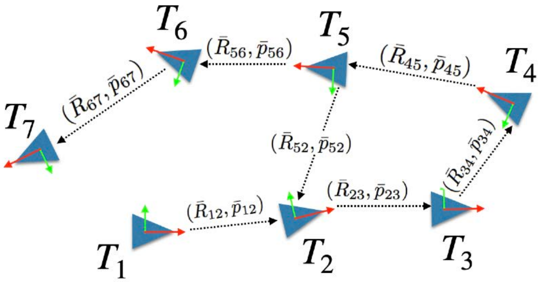
Pose Graph



- **Variables**: robot poses T_i
- **Measurements**: relative poses from odometry and loop closures: $\bar{T}_{ij}$
- **Factors**: relative pose vectors $\mathbf{e}(T_i, T_j) = \log(\bar{T}_{ij}^{-1} T_i^{-1} T_j)^\vee$

Pose Graph Optimization

- **Pose graph**



- **Loop closure**: observing previously seen areas generates factors between non-successive robot poses

- **Pose graph optimization**: with $\phi_{ij}(\mathbf{e}) = \mathbf{e}^\top W_{ij}^\top W_{ij}\mathbf{e} = \|W_{ij}\mathbf{e}\|_2^2$:

$$\min_{\{T_i\}} \sum_{(i,j)\in\mathcal{E}} \|W_{ij} \log(\bar{T}_{ij}^{-1} T_i^{-1} T_j)^\vee\|_2^2$$

Factor Graph Optimization

- **Factor graph optimization** with variables $\mathbf{x} = \begin{bmatrix} \mathbf{x}_1^\top & \cdots & \mathbf{x}_n^\top \end{bmatrix}^\top$:

$$\min_{\mathbf{x}} \sum_{(i,j)\in\mathcal{E}} \phi_{ij}(\mathbf{e}(\mathbf{x}_i, \mathbf{x}_j))$$

- **Initial guess** $\mathbf{x}^{(0)}$ is obtained from odometry (e.g., encoders, point cloud registration) and landmark initialization (e.g., triangulation of image features)

- A **descent method** is used for optimization:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)}\delta\mathbf{x}^{(k)}$$

- E.g., the **Levenberg-Marquardt** algorithm is used for $\phi_{ij}(\mathbf{e}) = \mathbf{e}^\top W_{ij}^\top W_{ij}\mathbf{e}$:

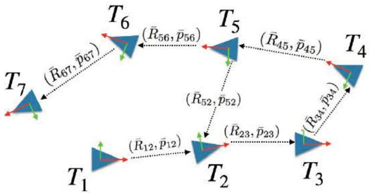
$$\left(\sum_{ij} J_{ij}^\top W_{ij}^\top W_{ij} J_{ij} + \lambda D\right)\delta\mathbf{x}^{(k)} = -\sum_{ij} J_{ij}^\top W_{ij}^\top \mathbf{e}(\mathbf{x}_i^{(k)}, \mathbf{x}_j^{(k)})$$

where $J_{ij} = \frac{\partial \mathbf{e}(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\mathbf{x}^{(k)}}$ is the Jacobian of $e(\mathbf{x}_i, \mathbf{x}_j)$ with respect to all variables $\mathbf{x}$ evaluated at $\mathbf{x} = \mathbf{x}^{(k)}$

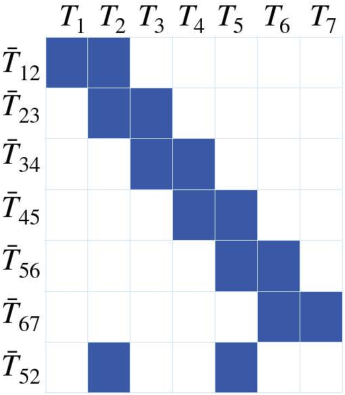
Factor Graph Optimization Libraries

- Georgia Tech Smoothing and Mapping (GTSAM) Library:
  `https://github.com/borglab/gtsam`
- General Graph Optimization (g2o) Library:
  `https://github.com/RainerKuemmerle/g2o`
- Ceres Solver: `https://github.com/ceres-solver/ceres-solver`
- SymForce: `https://github.com/symforce-org/symforce`
- miniSAM: `https://github.com/dongjing3309/minisam`

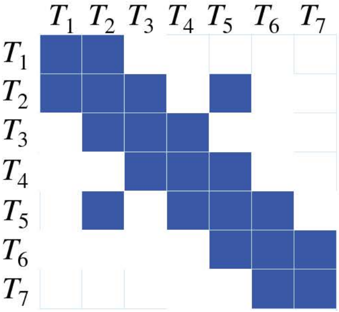
Factor Graph Optimization: Sparsity



Jacobian **J**

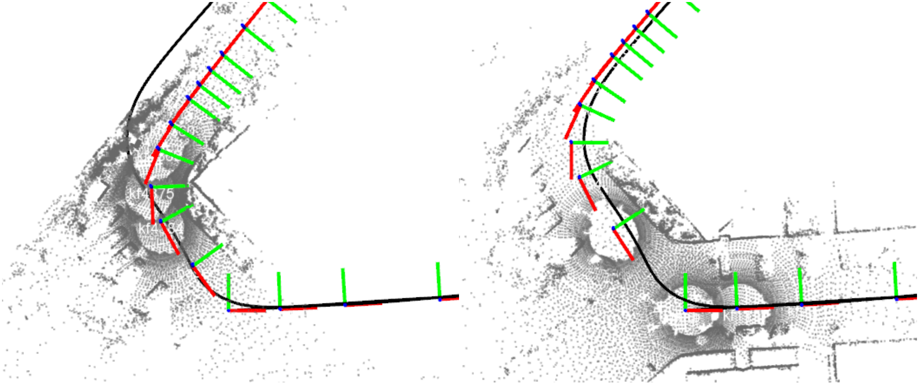


Hessian $\mathbf{J}^\mathsf{T}\mathbf{J}$



a.k.a.
Information
Matrix of
the estimate

Factor Graph Optimization: Example

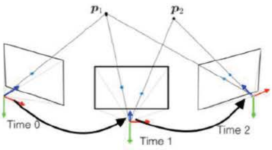


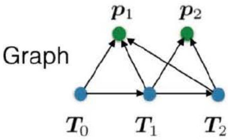
(a) Before optimization (b) After optimization

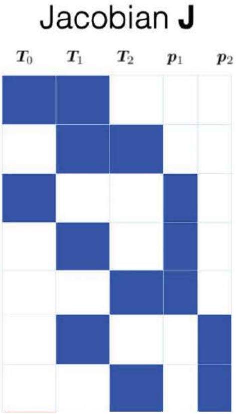
https://www.youtube.com/watch?v=KYvOqUB_odg

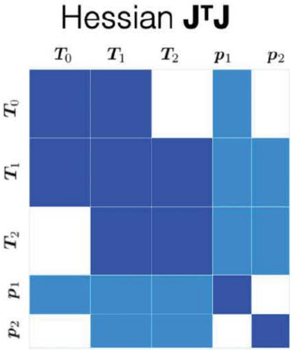
Landmark-Based SLAM

$$\min_{\{T_t\},\{\mathbf{m}_j\}} \sum_{t} \| W_{ij} \log(\bar{T}_{t,t+1}^{-1} T_t^{-1} T_{t+1})^{\vee} \|_2^2 + \sum_{t,j} \| V_{ij}(\mathbf{z}_{t,j} - h(T_t, \mathbf{m}_j)) \|_2^2$$

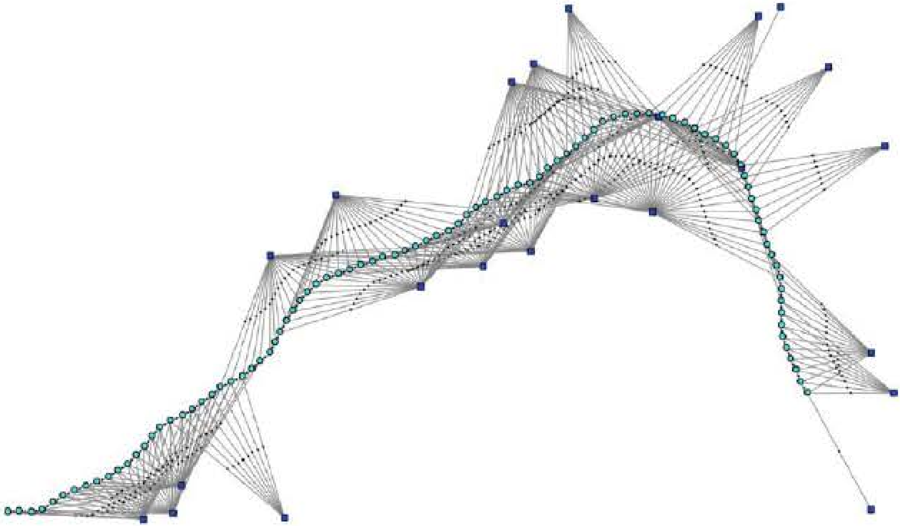




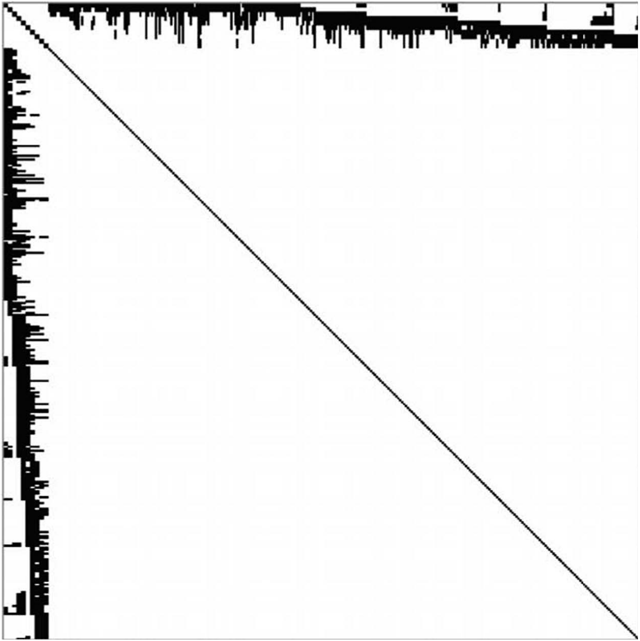




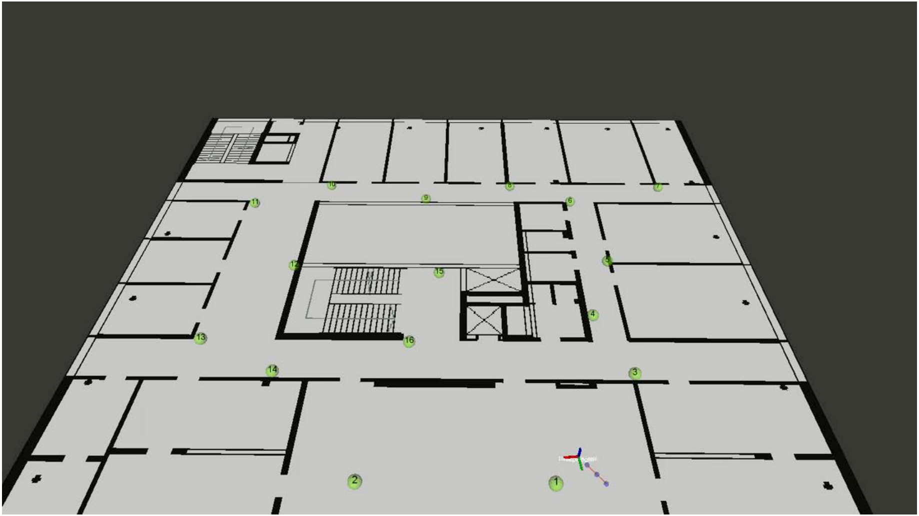
Landmark-Based SLAM



Landmark-Based SLAM: Sparsity

Hessian: $J^\top J =$ 

Landmark-Based SLAM: Example



https://www.youtube.com/watch?v=OdJ042prg_M

Landmark-Based SLAM: Variable Marginalization

- What if we only need a subset of the variables?
- Normal equations: $J^\top J \delta \mathbf{x} = -J^\top \mathbf{e}$
- Hessian matrix blocks:

$$J^\top J \delta \mathbf{x} = \begin{bmatrix} \Omega_{aa} & \Omega_{ab} \\ \Omega_{ab}^\top & \Omega_{bb} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{x}}_a \\ \tilde{\mathbf{x}}_b \end{bmatrix} = \begin{bmatrix} \mathbf{c}_a \\ \mathbf{c}_b \end{bmatrix} = -J^\top \mathbf{e}$$

- Pre-multiply by $\begin{bmatrix} I & -\Omega_{ab}\Omega_{bb}^{-1} \\ 0 & I \end{bmatrix}$ and subtract second from first equation:

$$\begin{bmatrix} \Omega_{aa} - \Omega_{ab}\Omega_{bb}^{-1}\Omega_{ab}^\top & 0 \\ \Omega_{ab}^\top & \Omega_{bb} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{x}}_a \\ \tilde{\mathbf{x}}_b \end{bmatrix} = \begin{bmatrix} \mathbf{c}_a - \Omega_{ab}\Omega_{bb}^{-1}\mathbf{c}_b \\ \mathbf{c}_b \end{bmatrix}$$

- We can obtain $\tilde{\mathbf{x}}_a$ by solving the smaller system determined by the Schur complement of Ω_{bb}:

$$(\Omega_{aa} - \Omega_{ab}\Omega_{bb}^{-1}\Omega_{ab}^\top)\tilde{\mathbf{x}}_a = \mathbf{c}_a - \Omega_{ab}\Omega_{bb}^{-1}\mathbf{c}_b$$

Landmark-Based SLAM: Variable Marginalization

- Probabilistic perspective of Schur complement:

$$\begin{bmatrix} \tilde{\mathbf{x}}_a \\ \tilde{\mathbf{x}}_b \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \Omega_{aa} & \Omega_{ab} \\ \Omega_{ab}^\top & \Omega_{bb} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{c}_a \\ \mathbf{c}_b \end{bmatrix}, \begin{bmatrix} \Omega_{aa} & \Omega_{ab} \\ \Omega_{ab}^\top & \Omega_{bb} \end{bmatrix}^{-1} \right)$$

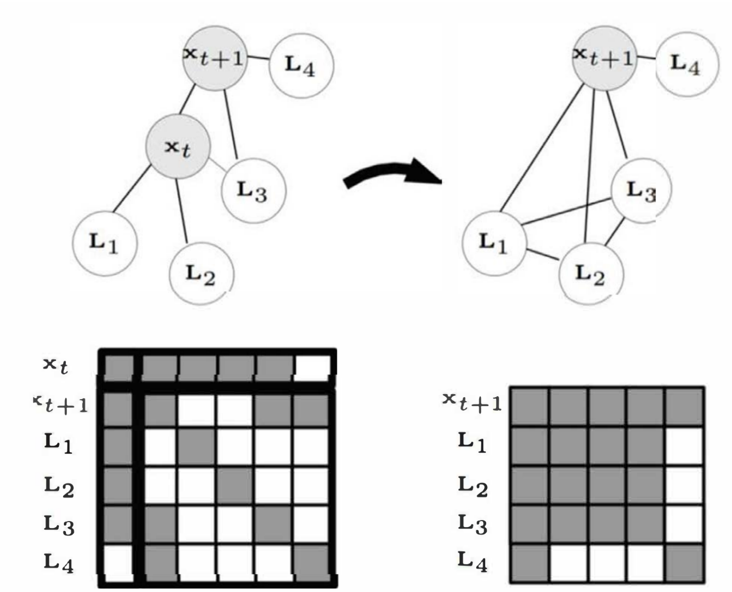
- Marginal of $\tilde{\mathbf{x}}_a$:

$$\begin{aligned} p(\tilde{\mathbf{x}}_a) &= \int p(\tilde{\mathbf{x}}_a, \tilde{\mathbf{x}}_b) d\tilde{\mathbf{x}}_b \\ &= \phi\left(\tilde{\mathbf{x}}_a; (\Omega_{aa} - \Omega_{ab}\Omega_{bb}^{-1}\Omega_{ab}^\top)^{-1}(\mathbf{c}_a - \Omega_{ab}\Omega_{bb}^{-1}\mathbf{c}_b), (\Omega_{aa} - \Omega_{ab}\Omega_{bb}^{-1}\Omega_{ab}^\top)^{-1}\right) \end{aligned}$$

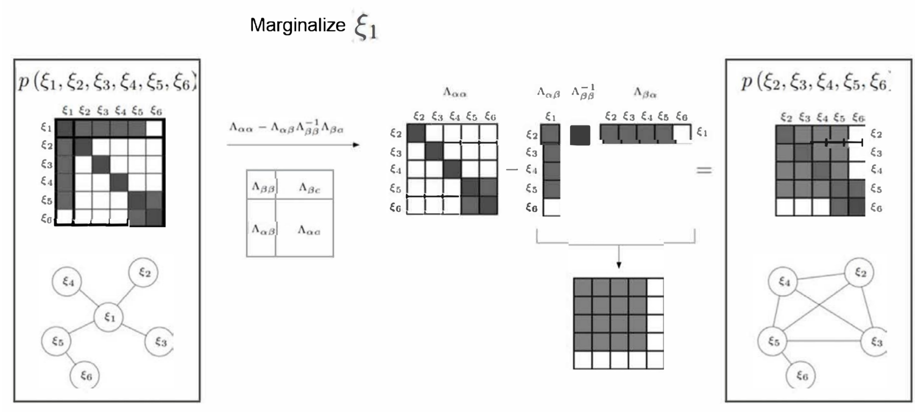
- Marginalizing a variable creates non-zero off-diagonals (called **fill-in**) in the information matrix for all variables that had a non-zero off-diagonal element with the marginalized variable $\Rightarrow$ **loss of sparsity**

- In graph terms, variable elimination creates a clique between the neighbors of the eliminated node

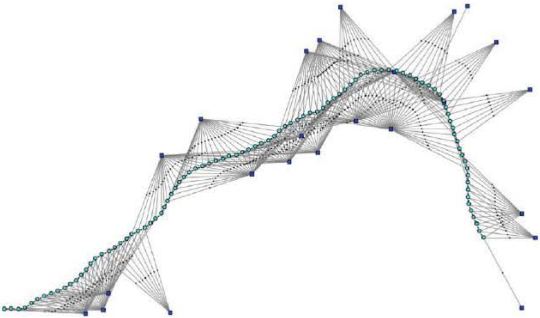
Landmark-Based SLAM: Variable Marginalization



Landmark-Based SLAM: Variable Marginalization



Smoothing vs Filtering



- **Smoothing**: equivalent to MAP optimization
  - **many variables**: estimates entire robot trajectory and map
  - **sparse** Hessian matrix $J^\top J$
- **Fixed-lag smoothing**:
  - **fewer variables**: estimate only variables in a time window
  - **denser** Hessian matrix after Schur complement to marginalize old variables
- **Filtering**:
  - **fewest variables**: estimate only current pose and landmarks
  - **densest** Hessian matrix after Schur complement to marginalize all old variables