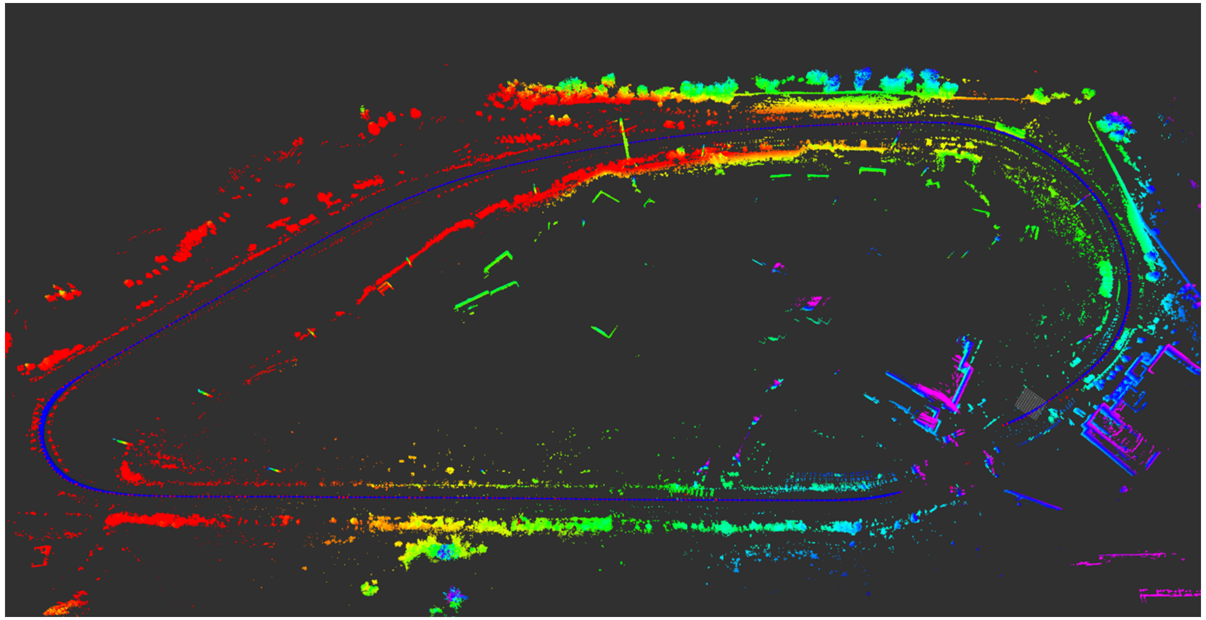
Factor Graphs and Nonlinear Least-Squares Problems on Manifolds

June 6 2017

Pranav Ganti

Outline

- Overview
- Factor Graphs
- Least Squares
- Nonlinear Least Squares
- NLLS – extended to Manifolds!



OVERVIEW

- SLAM is an optimization problem
- Given **many** measurements, what is the value of the parameters we are trying to estimate?
  - Overdetermined system
- Want to estimate the state, using the incoming measurements
  - Nonlinear
  - SE(3) or SE(2)

$$\begin{aligned} p(\mathbf{x} \mid \mathbf{z}) &= p(\mathbf{x}_1, \ldots, \mathbf{x}_N \mid \mathbf{z}_1, \ldots, \mathbf{z}_K) \\ &= p(\mathbf{x}_{1:N} | \mathbf{z}_{1:K}). \end{aligned}$$

- Independent
  - Previous sensory information does not affect the next reading

- Identically Distributed
  - Sensor noise distribution is unchanged between samples
  - **Gaussian**

UNIVERSITY OF WATERLOO

- Likelihood

$$\mathcal{L}(\theta; x_1, \ldots, x_n) = f(x_1, x_2, \ldots, x_n \mid \theta) = \prod_{i=1}^{n} f(x_i \mid \theta)$$

- Posterior

$$p(\mathbf{x}_{1:N} | \mathbf{z}_{1:K}) = \frac{\overbrace{p(\mathbf{z}_{1:K} | \mathbf{x}_{1:N})}^{\text{likelihood}} \cdot \overbrace{p(\mathbf{x}_{1:N})}^{\text{prior}}}{\underbrace{p(\mathbf{z}_{1:K})}_{\text{normalizer}}}$$

- ML – Maximum Likelihood

$$\hat{\theta}_{\mathrm{ML}}(x) = \underset{\theta}{\arg\max}\; f(x \mid \theta)$$

$$\{\hat{\theta}_{\mathrm{mle}}\} \subseteq \{\underset{\theta \in \Theta}{\arg\max}\; \hat{\ell}\,(\theta\,;\, x_1, \ldots, x_n)\}$$

- MAP – Maximum A Posteriori

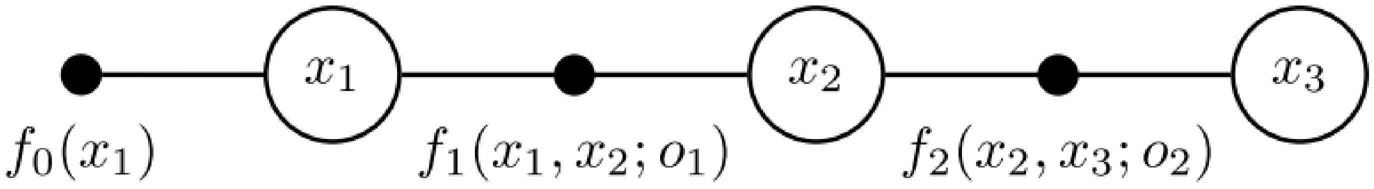
$$\hat{\theta}_{\mathrm{MAP}}(x) = \underset{\theta}{\arg\max}\; f(\theta \mid x) = \underset{\theta}{\arg\max}\; \frac{f(x \mid \theta)\, g(\theta)}{\displaystyle\int_{\vartheta} f(x \mid \vartheta)\, g(\vartheta)\, d\vartheta} = \underset{\theta}{\arg\max}\; f(x \mid \theta)\, g(\theta).$$

FACTOR GRAPHS

- Probabilistic model which illustrates the factorization of a function
- Highlights **conditional dependence** between random variables
- Bipartite – graph has two distinct nodes
    - Classified into **variables** and **factors**
    - Connected together by **edges**
- Excels in problems such as SLAM or SFM

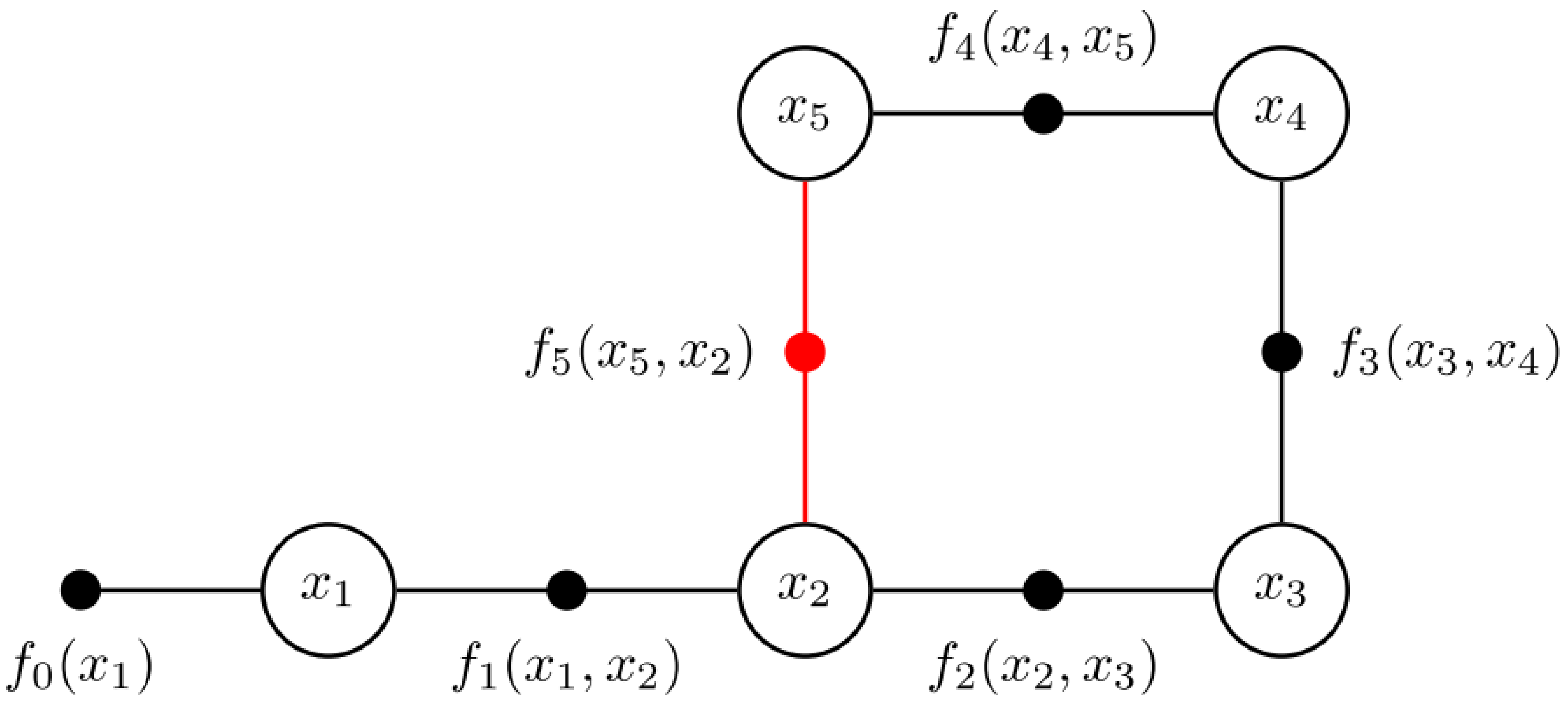
- Variables are the parameters that we are looking to optimize.
  - **For SLAM:** the robot (and landmark) poses.

- Factors are **probability statements**
  - Highlight the constraints between variables (conditional dependence)
  - Derived from measurement or mathematical fundamentals
  - **For SLAM:** odometry, reprojection error, GPS measurements, etc.

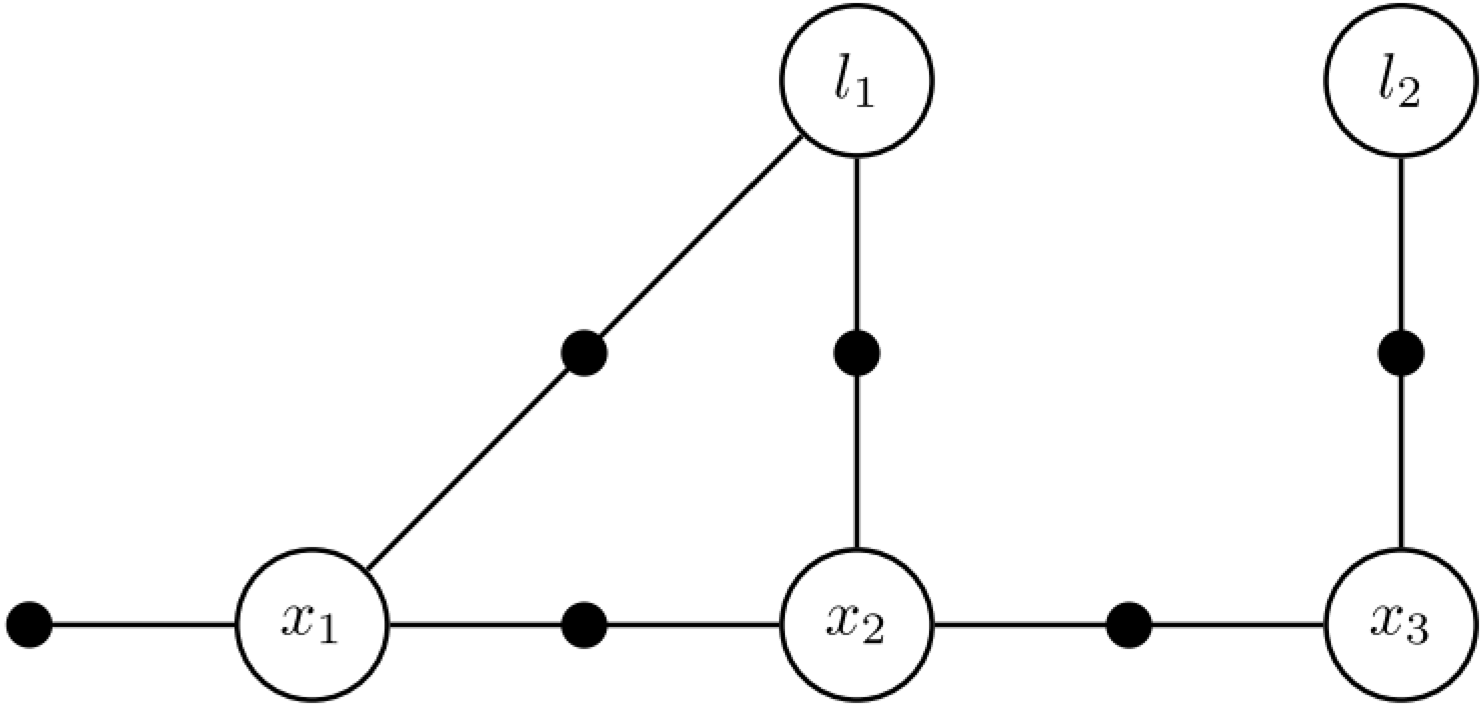


- x_1, x_2, x_3 are robot poses over 3 time steps
- $f_0(x_1)$ is the **prior**
  - **Unary** factor
- f_1, f_2 are **odometry** measurements
  - **Binary** factor

FACTOR GRAPHS | EXAMPLE – LOOP CLOSURE



- Let’s draw a factor graph with:
  - 3 timesteps
  - 2 landmarks
  - Odometry
  - LIDAR measurements to landmarks


l_1
l_2
x_1
x_2
x_3

- The **value** of the factor graph is the product of all factors.

$$f(X_1, X_2, X_3) = \prod f_i(\mathcal{X}_i)$$

- Maximizing the value is equivalent to the MAP estimation.
  - The **prior** is already included as a factor.
  - Recall:

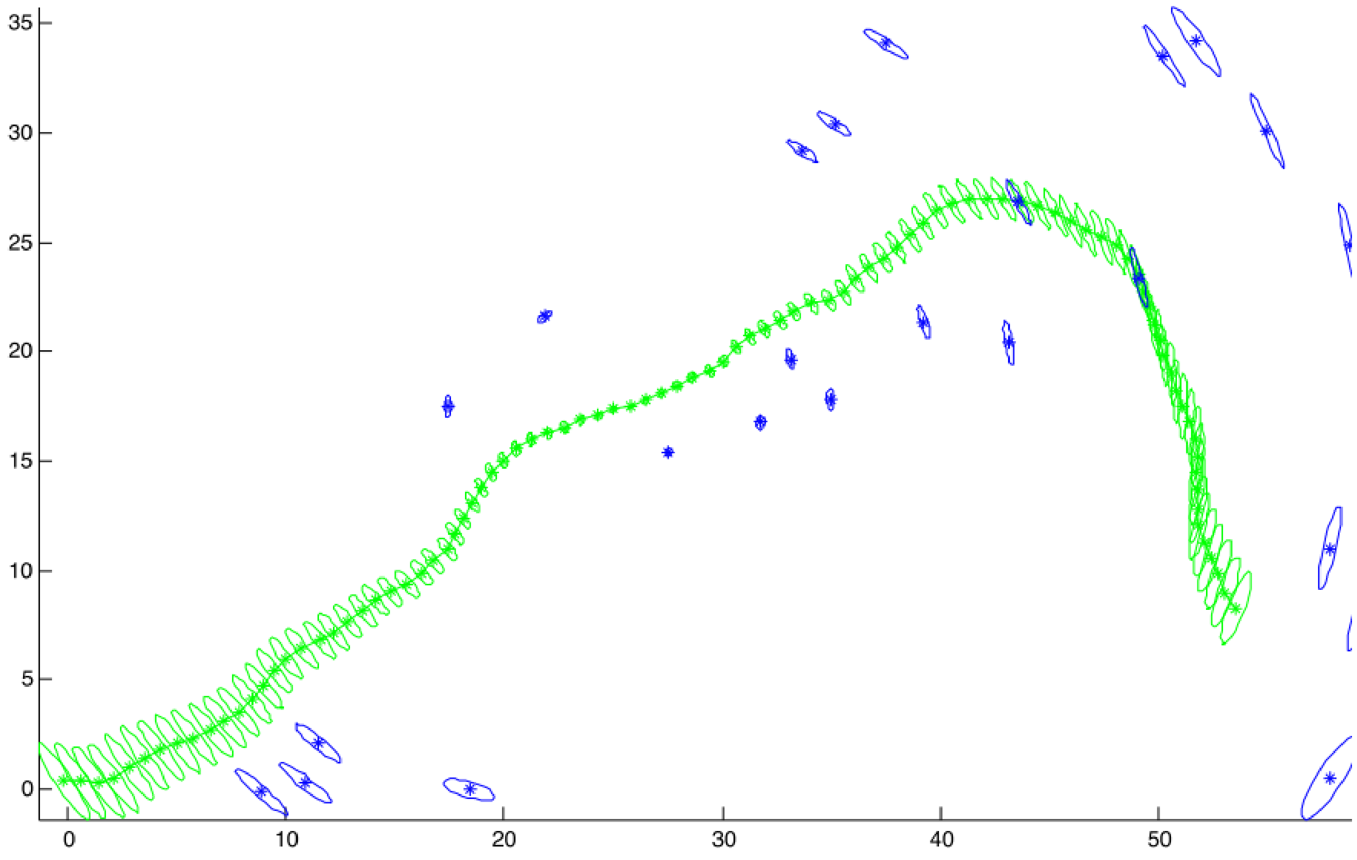
$$\hat{\theta}_{\mathrm{MAP}}(x) = \arg\max_{\theta} f(x \mid \theta)\, g(\theta)$$

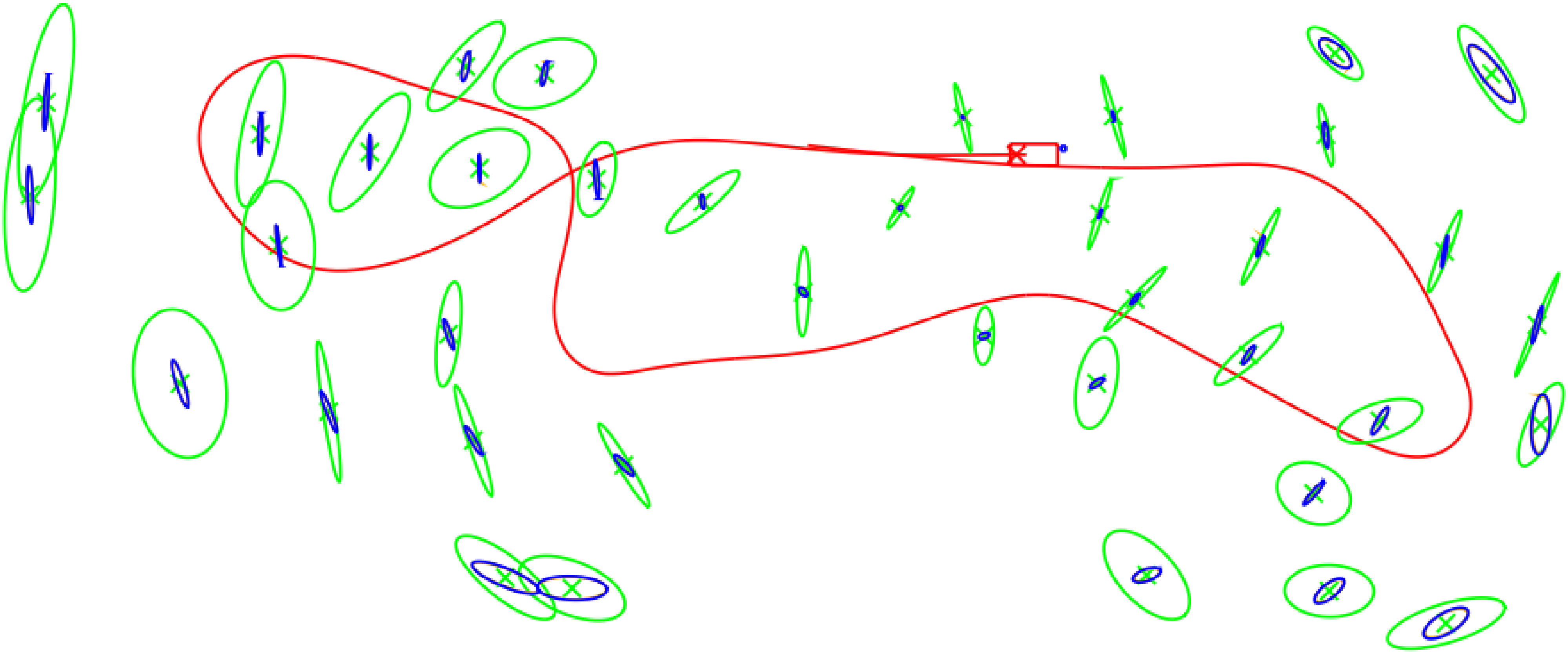
- The graph describes the posterior density over the full trajectory of the robot

- The graph **does not** contain a solution
  - The graph is a **function**, applied to the parameters

- An initial guess + nonlinear least squares can be used to find the MAP estimate for the trajectory

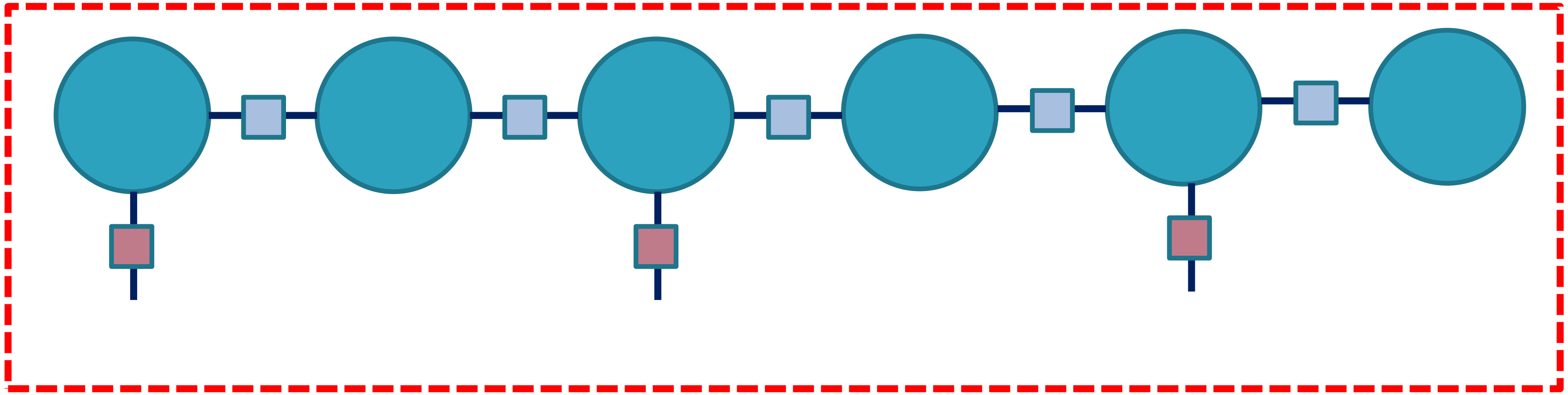
- The graph consists of:
  - 100 poses
  - 30 landmarks
- Using GTSAM with an initial guess solves for the full pose estimate of the robot and landmarks
  - Also includes covariances



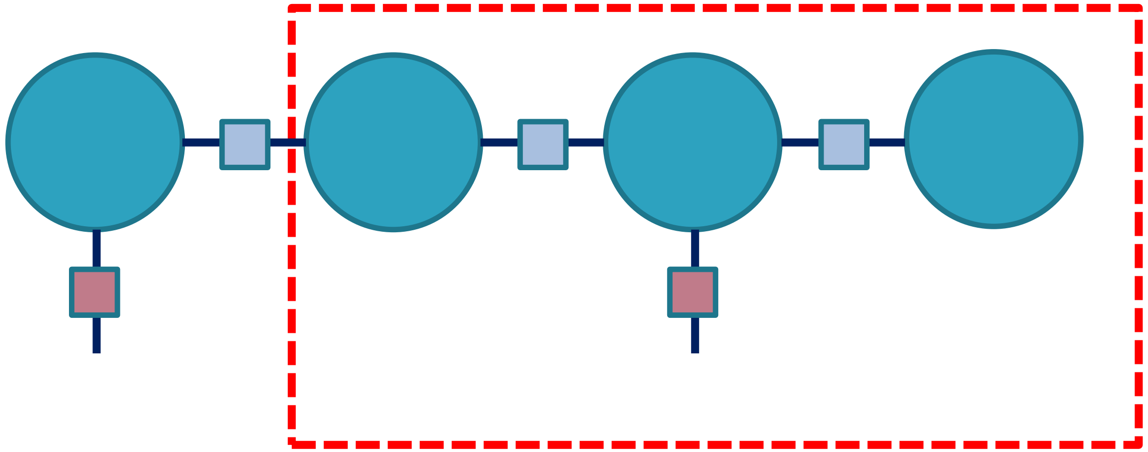


UNIVERSITY OF WATERLOO

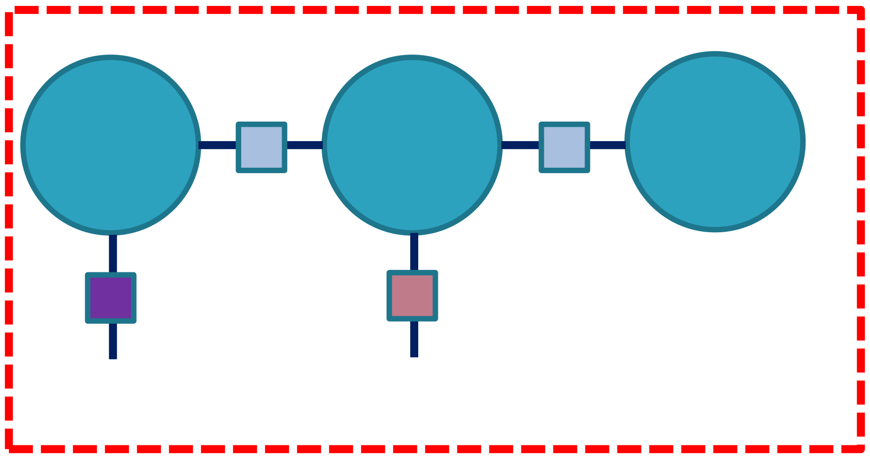
- Batch Estimation
  - Optimize over all poses in the trajectory



- Sliding window
  - Optimize only over poses in the window



- Sliding window
  - Optimize only over poses in the window



  - All previous information is encoded in as a prior

- Visual Odometry
  - Pose constraints provided by tracking features
- Visual SLAM
  - Extension of VO, to observing 3D points with mapping and loop closure
- Fixed-lag Smoothing and Filtering
  - Recursive estimation – only require a subset of the poses
  - Can marginalize for online estimation
- Discrete Variables and Hidden Markov Models

Least Squares

- Purpose: to solve an overdetermined system of equations.
- Review:

$$\hat{\mathbf{\Theta}} = \arg\min_{\mathbf{\Theta}} S(\mathbf{\Theta})$$

$$S(\mathbf{\Theta}) = \sum_{i=1}^{m} \left| y_i - \sum_{j=1}^{n} X_{ij}\theta_j \right|^2 = \left\| \mathbf{y} - \mathbf{X}\mathbf{\Theta} \right\|^2$$

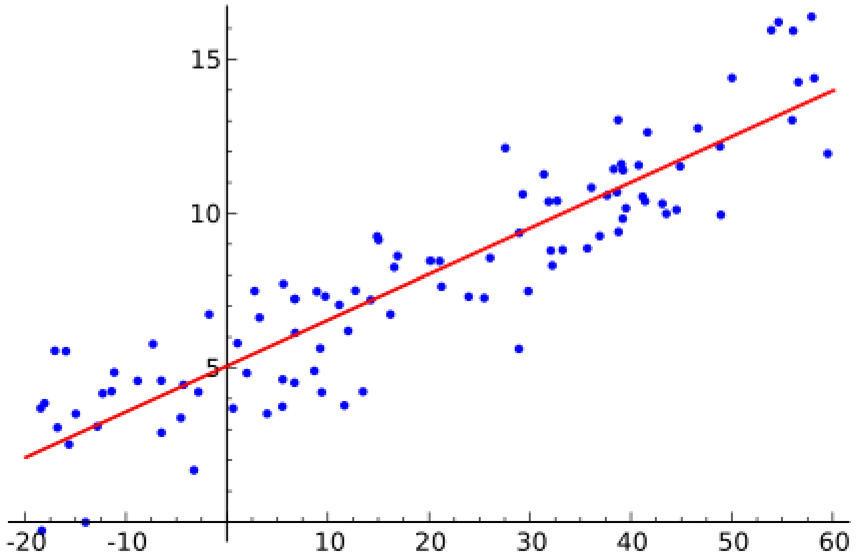
$$\mathcal{L}(\Theta\,;\, x_1, \ldots, x_n) = \frac{\exp\left(-\frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{\Theta})^{\mathrm{T}}\mathbf{\Sigma}^{-1}(\mathbf{y} - \mathbf{X}\mathbf{\Theta})\right)}{\sqrt{|2\pi\mathbf{\Sigma}|}}$$

$$S(\mathbf{\Theta}) = \mathbf{y}^{\mathrm{T}}\mathbf{y} - 2\mathbf{\Theta}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{y} + \mathbf{\Theta}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{\Theta}$$

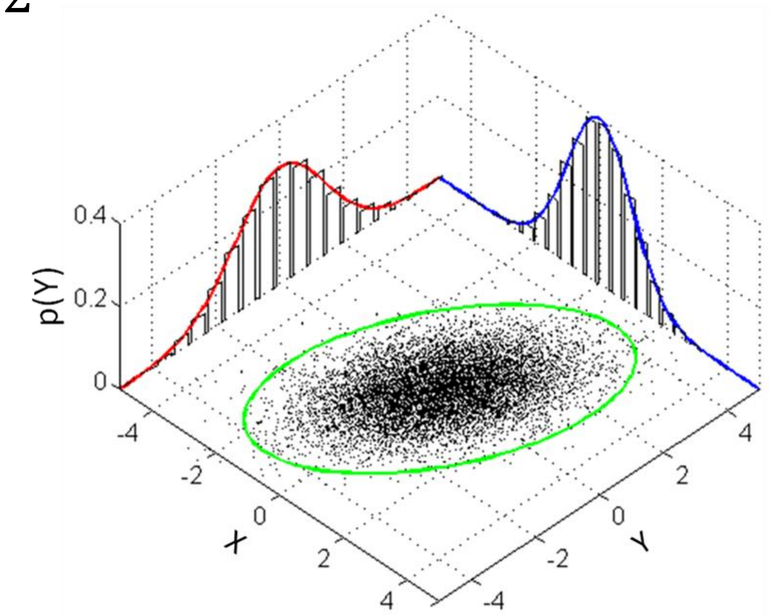
$$(\mathbf{X}^{\mathrm{T}}\mathbf{X})\mathbf{\Theta} = \mathbf{X}^{\mathrm{T}}\mathbf{y}$$

UNIVERSITY OF WATERLOO

- Linear:
  - $y = \beta_1 x + \beta_2$
  - $y = \beta_1 x^2$



- Nonlinear:
  - $y = e^{\beta_1 x}$
  - $y = \frac{1}{\sqrt{2\pi\Sigma}} e^{\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right)}$



- Not all factors are linear wrt the parameters.
  - **Odometry, reprojection error, etc.**

- **Main idea:** linearize model about current parameters and refine

- Let us work through the derivation:

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmax}} \prod_{k=1}^{K} p(\mathbf{z}_k | \mathbf{x})$$

UNIVERSITY OF WATERLOO

- The measurement (and therefore error) can be expressed as (Gaussian Assumption)
  - Error has zero mean, with Information Matrix Ω_k

$$= \quad \underset{\mathbf{x}}{\operatorname{argmax}} \prod_{k=1}^{K} \exp[-(\mathbf{h}_k(\mathbf{x}) - \mathbf{z}_k)^T \mathbf{\Omega}_k (\mathbf{h}_k(\mathbf{x}) - \mathbf{z}_k)]$$

- Remove the minus: (does this equation look familiar)?

$$= \quad \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{k=1}^{K} (\mathbf{h}_k(\mathbf{x}) - \mathbf{z}_k)^T \mathbf{\Omega}_k (\mathbf{h}_k(\mathbf{x}) - \mathbf{z}_k)$$

UNIVERSITY OF WATERLOO

- Let $\mathbf{e}_k(\mathbf{x}) = \mathbf{h}_k(\mathbf{x}) - \mathbf{z}_k$

- Therefore, the argument of minimization is:

$$F(\mathbf{x}) = \sum_{k=1}^{K} \underbrace{\mathbf{e}_k(\mathbf{x})^T \boldsymbol{\Omega}_k \mathbf{e}_k(\mathbf{x})}_{e_k(\mathbf{x})}$$

- We can take the **first-order Taylor Approximation** of the error function
  - Taken about the **initial guess**
  - J_k is the Jacobian at the guess.

$$e_k(\breve{x} + \Delta x) \approxeq e_k + J_k \Delta x$$

- Inserting this into the error function yields:

$$F(\breve{\mathbf{x}} + \mathbf{\Delta x}) = (\mathbf{h}_k(\breve{\mathbf{x}} + \mathbf{\Delta x}) - \mathbf{z}_k)^T \mathbf{\Omega}_k (\mathbf{h}_k(\breve{\mathbf{x}} + \mathbf{\Delta x}) - \mathbf{z}_k)$$

$$\simeq (\mathbf{J}_k \mathbf{\Delta x} + \mathbf{h}_k(\breve{\mathbf{x}}) - \mathbf{z}_k)^T \mathbf{\Omega}_k (\mathbf{J}_k \mathbf{\Delta x} + \mathbf{h}_k(\breve{\mathbf{x}}) - \mathbf{z}_k)$$

- Substitute error definition $\mathbf{h}_k(\breve{\mathbf{x}}) - \mathbf{z}_k = \mathbf{e}_k$

$$= (\mathbf{J}_k \mathbf{\Delta x} + \mathbf{e}_k)^T \mathbf{\Omega}_k (\mathbf{J}_k \mathbf{\Delta x} + \mathbf{e}_k)$$

- Multiplying the terms and expanding yields:

$$= \quad \Delta\mathbf{x}^T \underbrace{\mathbf{J}_k^T \boldsymbol{\Omega}_k \mathbf{J}_k}_{\mathbf{H}_k} \Delta\mathbf{x} + 2 \underbrace{\mathbf{e}_k^T \boldsymbol{\Omega}_k \mathbf{J}_k}_{\mathbf{b}_k^T} \Delta\mathbf{x} + \mathbf{e}_k^T \boldsymbol{\Omega}_k \mathbf{e}_k$$

- Which looks quite familiar!

$$\boxed{= \quad \Delta\mathbf{x}^T \mathbf{H}_k \Delta\mathbf{x} + 2\mathbf{b}_k^T \Delta\mathbf{x} + \mathbf{e}_k^T \boldsymbol{\Omega}_k \mathbf{e}_k}$$

- We can solve this!

UNIVERSITY OF WATERLOO

- The error function is the sum over all timesteps
  - Expressed as:

$$F(\breve{\mathbf{x}} + \mathbf{\Delta x}) \simeq \sum_{k=1}^{K} \mathbf{\Delta x}^T \mathbf{H}_k \mathbf{\Delta x} - 2\mathbf{b}_k^T \mathbf{\Delta x} + \mathbf{e}_k^T \mathbf{\Omega}_k \mathbf{e}_k$$

$$= \mathbf{\Delta x}^T \underbrace{\left[\sum_{k=1}^{K} \mathbf{H}_k\right]}_{\mathbf{H}} \mathbf{\Delta x} + 2 \underbrace{\left[\sum_{k=1}^{K} \mathbf{b}_k^T\right]}_{\mathbf{b}^T} \mathbf{\Delta x} + \underbrace{\left[\sum_{k=1}^{K} \mathbf{e}_k^T \mathbf{\Omega}_k \mathbf{e}_k\right]}_{c}$$

- The final step is to take the derivative, and set to zero.

$$\frac{\partial(\mathbf{\Delta x}^T \mathbf{H} \mathbf{\Delta x} - 2\mathbf{b}\mathbf{\Delta x} + c)}{\partial \mathbf{\Delta x}} = 2\mathbf{H}\mathbf{\Delta x} - 2\mathbf{b}$$

- This yields:

$$\boxed{\mathbf{H}\mathbf{\Delta x}* = \mathbf{b}}$$

- And the next iteration occurs at: $\mathbf{x}* = \check{\mathbf{x}} + \mathbf{\Delta x}^*$
  - Until the convergence condition is met

- This is the **Gauss-Newton** algorithm.
  - However, sometimes Gauss-Newton can lead to **worse** estimates

- Levenberg-Marquardt is a **damped** (λ) version of the Gauss-Newton algorithm
  - Introduces contingency to recover from a worse estimate
  - As λ moves to ∞, Δx^* moves to 0
    - Controls size of increments

$$(\mathbf{H} + \lambda \mathbf{I}) \boldsymbol{\Delta} \mathbf{x}^* = \mathbf{b}$$

- Both SLAM and Bundle Adjustment have a characteristic structure
  - **Sparsity**
  - Only non-zero between poses connected by a constraint
    - = 2x # of constraints + # of nodes

- We can exploit this structure to solve $\mathbf{H}\Delta\mathbf{x}* = \mathbf{b}$

- From: $e_k(\breve{x} + \Delta x) \approxeq e_k + J_k \Delta x$

- The Jacobian J_k can be expressed as:

$$\mathbf{J}_k = \left(\mathbf{0} \cdots \mathbf{0}\ \mathbf{J}_{k_1}\ \cdots\ \mathbf{J}_{k_i}\ \cdots \mathbf{0}\ \cdots\ \mathbf{J}_{k_q} \mathbf{0} \cdots \mathbf{0}\right)$$

- Each $\mathbf{J}_{k_i} = \frac{\partial \mathbf{e}(\mathbf{x}_k)}{\partial \mathbf{x}_{k_i}}$ corresponds to the derivative wrt the nodes connected by the k^{th} edge

UNIVERSITY OF WATERLOO

- By setting
  - $\mathbf{H}_k = \mathbf{J}_k^T \boldsymbol{\Omega}_k \mathbf{J}_k$
  - $\mathbf{b}_k = \mathbf{J}_k^T \boldsymbol{\Omega}_k \mathbf{e}_k$

$$\mathbf{b}_k = \begin{pmatrix} \vdots \\ \mathbf{J}_{k_1} \boldsymbol{\Omega}_k \mathbf{e}_k \\ \vdots \\ \mathbf{J}_{k_i}^T \boldsymbol{\Omega}_k \mathbf{e}_k \\ \vdots \\ \mathbf{J}_{k_q}^T \boldsymbol{\Omega}_k \mathbf{e}_k \\ \vdots \end{pmatrix}$$

$$\mathbf{H}_k = \begin{pmatrix} \ddots & & & & & & \\ & \mathbf{J}_{k_1}^T \boldsymbol{\Omega}_k \mathbf{J}_{k_1} & \cdots & \mathbf{J}_{k_1}^T \boldsymbol{\Omega}_k \mathbf{J}_{k_i} & \cdots & \mathbf{J}_{k_1}^T \boldsymbol{\Omega}_k \mathbf{J}_{k_q} & \\ & \vdots & & \vdots & & \vdots & \\ & \mathbf{J}_{k_i}^T \boldsymbol{\Omega}_k \mathbf{J}_{k_1} & \cdots & \mathbf{J}_{k_i}^T \boldsymbol{\Omega}_k \mathbf{B}_{k_i} & \cdots & \mathbf{J}_{k_i}^T \boldsymbol{\Omega}_k \mathbf{J}_{k_q} & \\ & \vdots & & \vdots & & \vdots & \\ & \mathbf{J}_{k_q}^T \boldsymbol{\Omega}_k \mathbf{J}_{k_1} & \cdots & \mathbf{J}_{k_q}^T \boldsymbol{\Omega}_k \mathbf{B}_{k_i} & \cdots & \mathbf{J}_{k_q}^T \boldsymbol{\Omega}_k \mathbf{J}_{k_q} & \\ & & & & & & \ddots \end{pmatrix}$$

- This can now be solved by Sparse Cholesky Factorization
  - Cholesky decomposition: $\mathbf{A} = \mathbf{L}\mathbf{L}^{\mathrm{T}}$

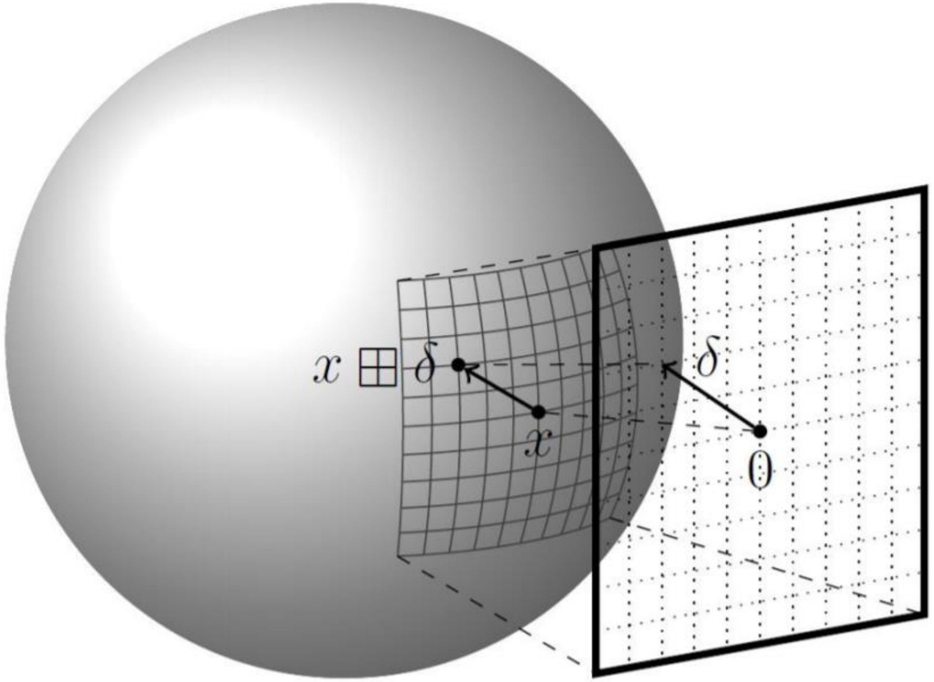
$$\mathbf{A}\mathbf{x} = \mathbf{b} \longrightarrow \mathrm{L}\mathrm{L}^{\mathrm{T}}\mathrm{x} = \mathrm{b} \longrightarrow \mathbf{L}\mathbf{y} = \mathbf{b} \longrightarrow \mathbf{L}^{\mathbf{T}}\mathbf{x} = \mathbf{y}$$

- Solvers:
  - CSparse
  - CHOLMOD
  - Preconditioned Gradient (PCG)
    - **Only if system is too large**

- The previous derivation assumes the parameters are Euclidean
  - **Not true** for SLAM
  - **Pose** estimates are SE(3)

- Box-plus and box-minus

$$\boxplus : \mathcal{S} \times \mathbb{R}^n \rightarrow \mathcal{S},$$
$$\boxminus : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}^n$$



- Recall: taking Taylor expansion of error near initial guess
  - Now, we must look at a perturbation

$$\begin{aligned} \mathbf{h}_k(\breve{\mathbf{X}} \boxplus \boldsymbol{\Delta}\mathbf{x}) &\simeq \mathbf{h}_k(\mathbf{X}) + \underbrace{\left.\frac{\partial \mathbf{h}_k(\breve{\mathbf{X}} \boxplus \boldsymbol{\Delta}\mathbf{x})}{\partial \boldsymbol{\Delta}\mathbf{x}}\right|_{\boldsymbol{\Delta}\mathbf{x}=\mathbf{0}}}_{\tilde{\mathbf{J}}_k} \cdot \boldsymbol{\Delta}\mathbf{x} \\ &= \mathbf{h}_k(\breve{\mathbf{X}}) + \tilde{\mathbf{J}}_k \boldsymbol{\Delta}\mathbf{x}. \end{aligned}$$

- Replace addition with box plus, and subtraction with box-minus

$$\begin{aligned} F(\breve{\mathbf{x}} + \boldsymbol{\Delta}\mathbf{x}) &= (\mathbf{h}_k(\breve{\mathbf{x}} + \boldsymbol{\Delta}\mathbf{x}) - \mathbf{z}_k)^T \boldsymbol{\Omega}_k (\mathbf{h}_k(\breve{\mathbf{x}} + \boldsymbol{\Delta}\mathbf{x}) - \mathbf{z}_k) \\ &\simeq (\mathbf{J}_k \boldsymbol{\Delta}\mathbf{x} + \mathbf{h}_k(\breve{\mathbf{x}}) - \mathbf{z}_k)^T \boldsymbol{\Omega}_k (\mathbf{J}_k \boldsymbol{\Delta}\mathbf{x} + \mathbf{h}_k(\breve{\mathbf{x}}) - \mathbf{z}_k) \end{aligned}$$

- Substitute error definition with box-plus and box-minus

$$\boxed{\tilde{\mathbf{e}}_k(\mathbf{x}) = \tilde{\mathbf{e}}_k(\hat{\mathbf{z}}_k, \mathbf{z}_k) = \hat{\mathbf{z}}_k \boxminus \mathbf{z}_k = \mathbf{h}_k(\mathbf{x}) \boxminus \mathbf{z}_k}$$

- The Jacobian takes on the form:

$$\tilde{\mathbf{J}}_{ij} = \left(\cdots \underbrace{\left. \frac{\partial \mathbf{e}_{ij}(\breve{\mathbf{x}} \boxplus \mathbf{\Delta}\tilde{\mathbf{x}})}{\partial \mathbf{\Delta}\tilde{\mathbf{x}}_i} \right|_{\mathbf{\Delta}\tilde{\mathbf{x}}=\mathbf{0}}}_{\tilde{\mathbf{A}}_{ij}} \cdots \underbrace{\left. \frac{\partial \mathbf{e}_{ij}(\breve{\mathbf{x}} \boxplus \mathbf{\Delta}\tilde{\mathbf{x}})}{\partial \mathbf{\Delta}\tilde{\mathbf{x}}_j} \right|_{\mathbf{\Delta}\tilde{\mathbf{x}}=\mathbf{0}}}_{\tilde{\mathbf{B}}_{ij}} \cdots \right)$$

- The remaining steps are just an extension of the Euclidean derivation
  - Solution does depend on implementation of box-plus and box-minus

$$\tilde{\mathbf{H}}\,\mathbf{\Delta}\tilde{\mathbf{x}}^* = -\tilde{\mathbf{b}} \qquad \mathbf{x}^* = \breve{\mathbf{x}} \boxplus \mathbf{\Delta}\tilde{\mathbf{x}}^*$$

- Linearized manifold representation has the **same structure** as Euclidean case
- Steps:
  - Compute set of increments in local Euclidean approximation
- Libraries such as GTSAM and g^2o incorporate this functionality, as they are designed for SLAM

HOMEWORK

- Draw a factor graph with the following:
  - 5 timesteps
  - 6 landmarks
  - GPS measurements at each timestep
  - Odometry
  - IMU (at same frequency as odometry)
  - LIDAR measurements to each landmark
  - Reprojection error

Course References

- [1] R. Kümmerle, G. Grisetti. H. Strasdat, K. Konolige, and W. Burgard, “g 2 o: A general framework for graph optimization,” in *Robotics and Automation (ICRA), 2011 IEEE International Conference on,* pp. 3607-3613, IEEE, 2011.
- [2] F. Dellaert, “Factor graphs and gtsam: A hands-on introduction,” tech.rep., Georgia Institute of Technology, 2012.

Additional References

- [3] M. Kaess, A. Ranganathan, and F. Dellaert, “isam: Incremental smoothing and mapping,” *IEEE Transactions on Robotics,* vol. 24, no. 6, pp. 1365-1378, 2008.
- [4] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, “A tutorial on graph-based SLAM,” *IEEE Intelligent Transportation Systems Magazine,* vol. 2, no. 4, pp. 31-43, 2010.
- [5] G. Grisetti, “Notes on Least-Squares and SLAM DRAFT,” 2015. Retrieved from http://www.dis.uniroma1.it/~grisetti/teaching/lectures-ls-slam-master_2015_16/web/reading_material/grisetti12stest.pdf

REFERENCES…CONT

- [6] C. Stachniss, "Least squares approach to SLAM." *Lecture Slides, University of Freiburg.* Retrieved from http://ais.informatik.uni-freiburg.de/teaching/ws12/mapping/pdf/slam15-ls-slam.pdf
- [7] W. Burgard, C. Stachniss, K. Arras, M. Bennewitz, "Advanced Techniques for Mobile Robotics – Least Squares." *Lecture Slides, University of Freiburg.* Retrieved from http://ais.informatik.uni-freiburg.de/teaching/ws11/robotics2/pdfs/rob2-06-least-squares.pdf.

All images retrieved from wikipedia.org.