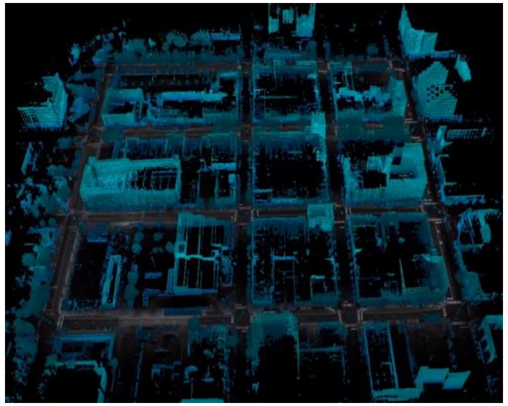
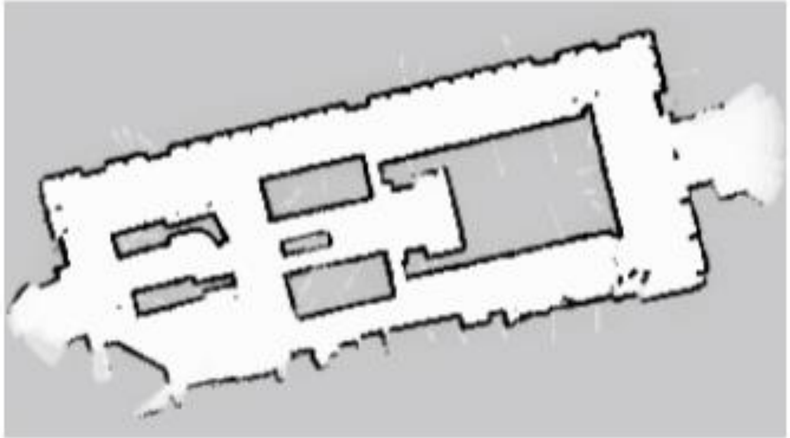
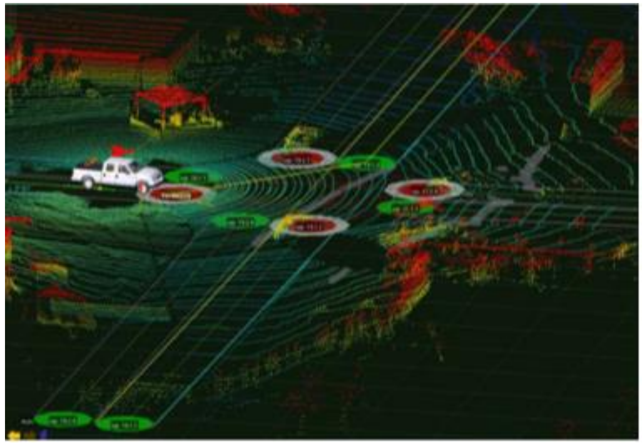
BYU Electrical & Computer Engineering





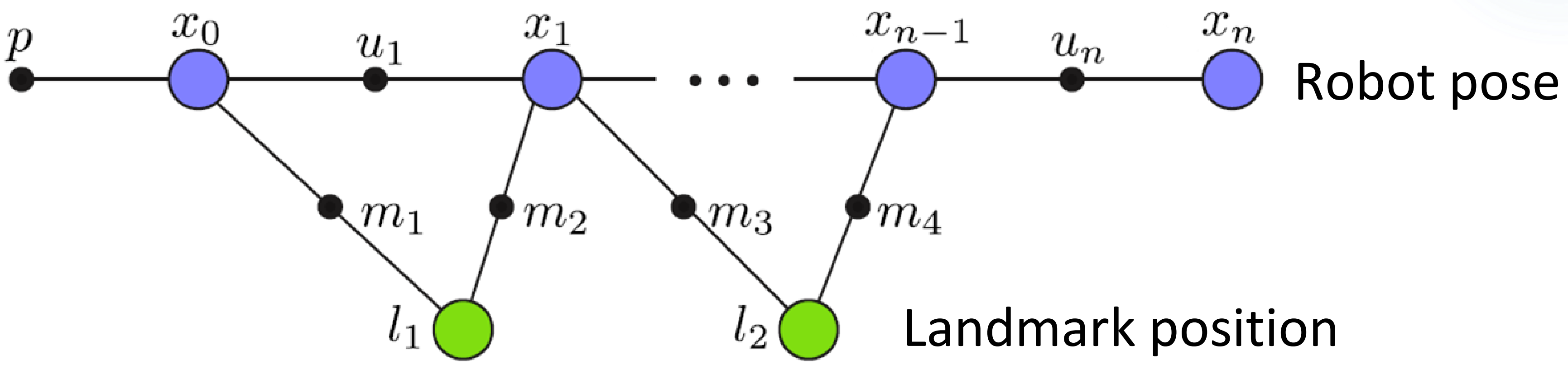


FACTOR GRAPH SLAM

ECEN 633: Robotic Localization and Mapping

Many slides courtesy of Michael Kaess

Factor Graph Representation of SLAM



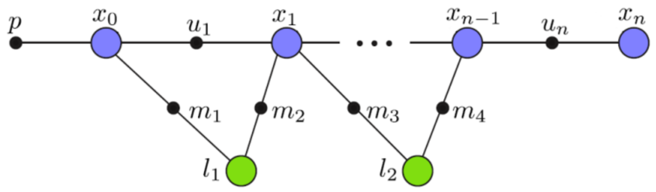
Variables: $\Theta = \{x_0, x_1 \cdots x_n, l_1, l_2\}$

Measurements: $\mathrm{Z} = \{p, u_1 \cdots u_n, m_1 \cdots m_4\}$

↕

Factorization: $\mathrm{p(Z|\Theta) = argmax_{\Theta}} \prod_{z \in Z} p(z|\Theta)$

SLAM as a Least-Squares Problem



$$\operatorname{argmax}_\Theta \prod_{z \in Z} p\left(z|\Theta\right)$$

Gaussian noise

$$\operatorname{argmin}_\Theta \sum_i \|h_i(\Theta) - z_i\|^2$$

$$\operatorname{argmin}_\theta \|A\theta - b\|^2$$

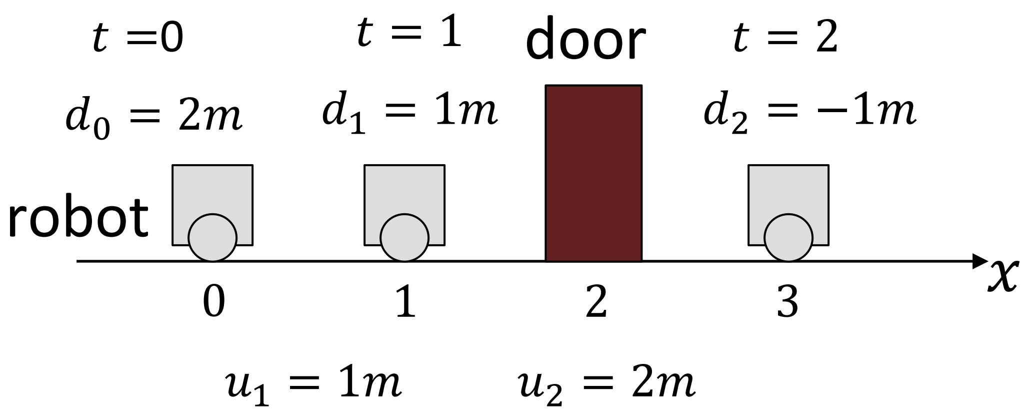
Normal equations:

$$A^T A\theta = A^T b$$

Solving for θ by matrix inversion is too expensive!

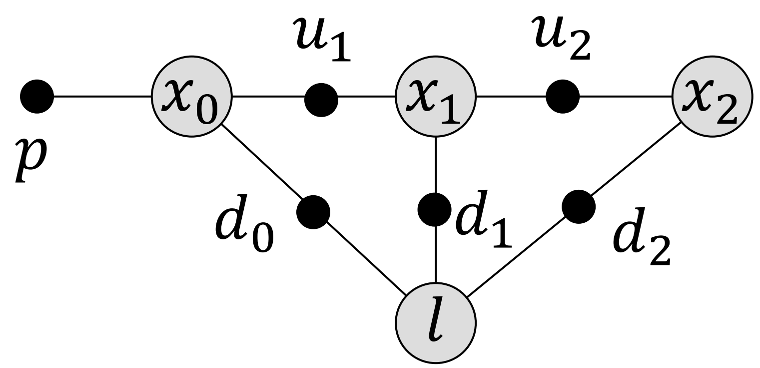
SLAM as a Least-Squares Problem: Example

Localize robot and door based on 1D range measurements



Measurements: distance to the door, signed

Factor graph:



SLAM as a Least-Squares Problem: Example

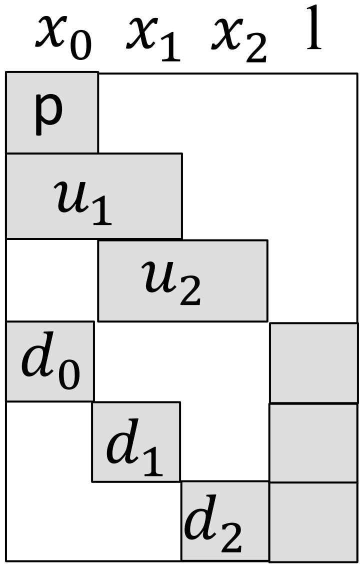
Localize robot and door based on 1D range measurements

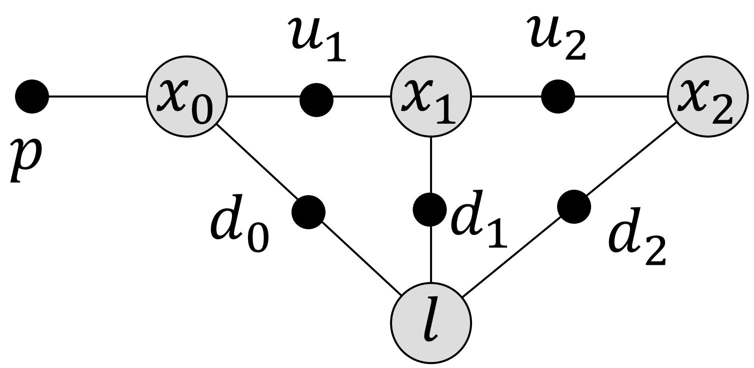
Matrix A:

Each row corresponds to a factor

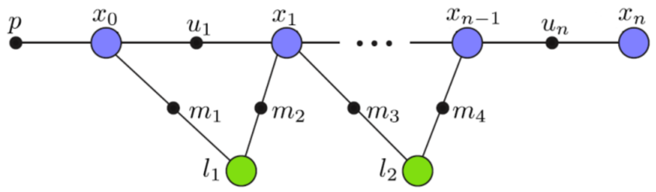
Each column to a variable

A is sparse!





SLAM as a Least-Squares Problem



$$\text{argmax}_{\Theta} \prod_{z \in Z} p\left(z|\Theta\right)$$

Gaussian noise

$$\text{argmin}_{\Theta} \sum_i \|h_i(\Theta) - z_i\|^2$$

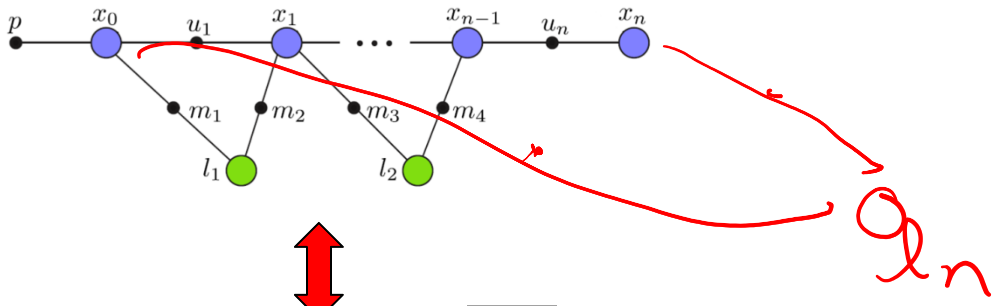
$$\text{argmin}_{\theta} \|A\theta - b\|^2$$

Normal equations:

$$A^T A\theta = A^T b$$

Solving for θ by matrix inversion is too expensive!

SLAM as a **Sparse** Least-Squares Problem



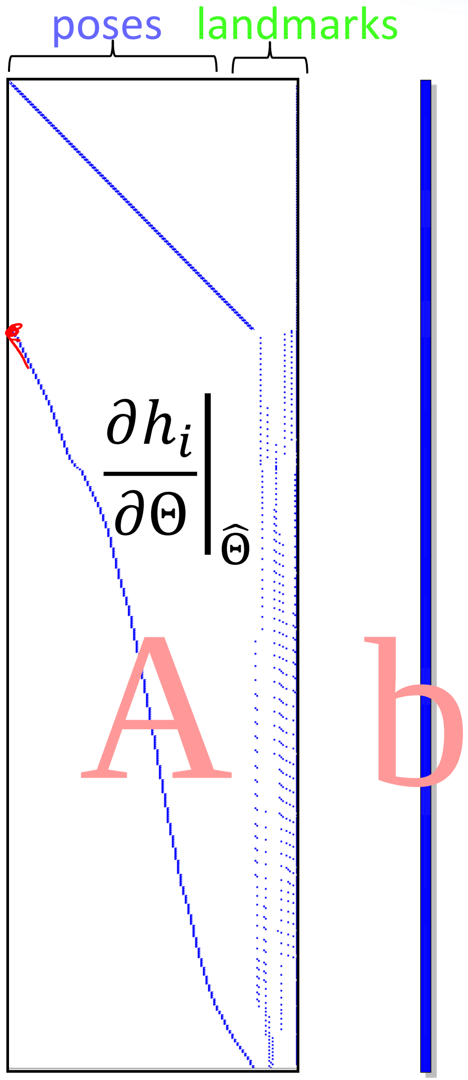
$$\operatorname{argmax}_{\Theta} \prod_{z \in Z} p\left(z \mid \Theta\right)$$

Gaussian noise

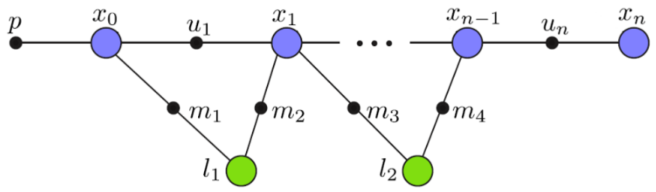
$$\operatorname{argmin}_{\Theta} \sum_{i} \|h_i(\Theta) - z_i\|^2$$

$$\operatorname{argmin}_{\theta} \|A\theta - b\|^2$$

Sparse measurement Jacobian



SLAM as a **Sparse** Least-Squares Problem



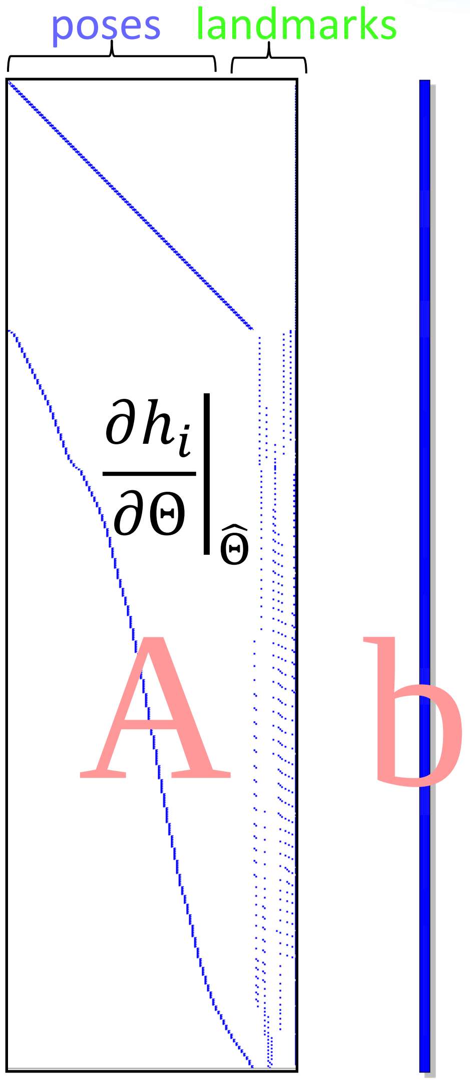
$$\operatorname{argmax}_{\Theta} \prod_{z \in Z} p\,(z|\Theta)$$

Gaussian noise

$$\operatorname{argmin}_{\Theta} \sum_{i} \|h_i(\Theta) - z_i\|^2$$

$$\operatorname{argmin}_{\theta} \|A\theta - b\|^2$$

Sparse measurement Jacobian



$$\left.\frac{\partial h_i}{\partial \Theta}\right|_{\hat{\Theta}}$$

Efficient Solution

- On the board:
  - Sparse matrix factorization
  - Solving by back substitution

Efficient Solution: Cholesky Factorization

Cholesky factor R is an upper triangular matrix so that

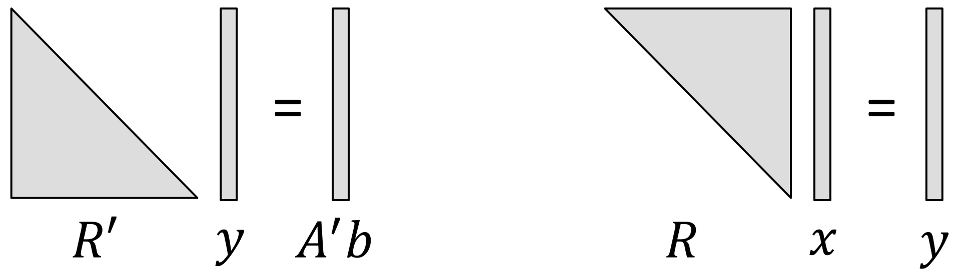
$$R'R = A'A$$

Yielding

$$R'Rx = A'b$$

Solve by forward-/backsubstitution

$$R'y = A'b$$
$$Rx = y$$



Similar: LDL' factorization, faster than Cholesky, avoids square roots

Efficient Solution: QR Factorization

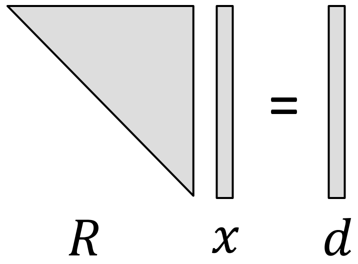
$$A = Q \begin{bmatrix} R \\ 0 \end{bmatrix}$$

Yielding

$$\|Ax - b\|^2 = \left\| Q \begin{bmatrix} R \\ 0 \end{bmatrix} x - b \right\|^2 = \left\| Q'Q \begin{bmatrix} R \\ 0 \end{bmatrix} x - \begin{bmatrix} d \\ e \end{bmatrix} \right\|^2 = \|Rx - d\|^2 + \|e\|^2$$

Solve by backsubstitution

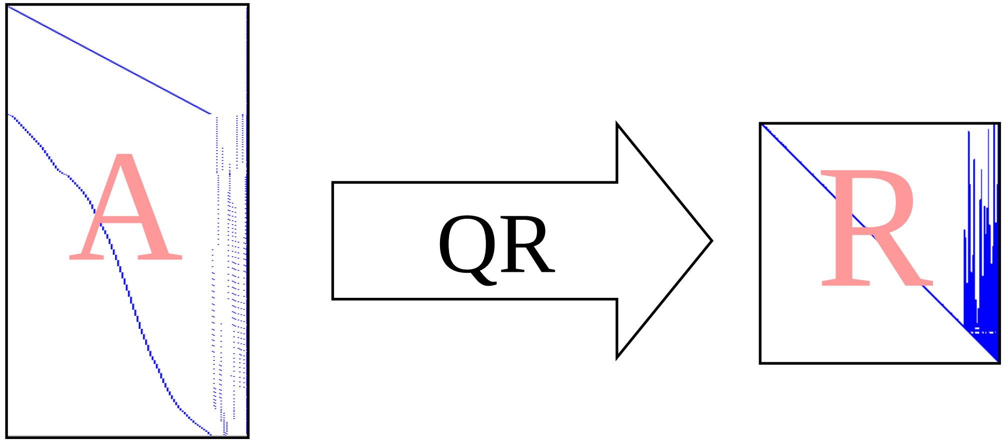
$$Rx = d$$



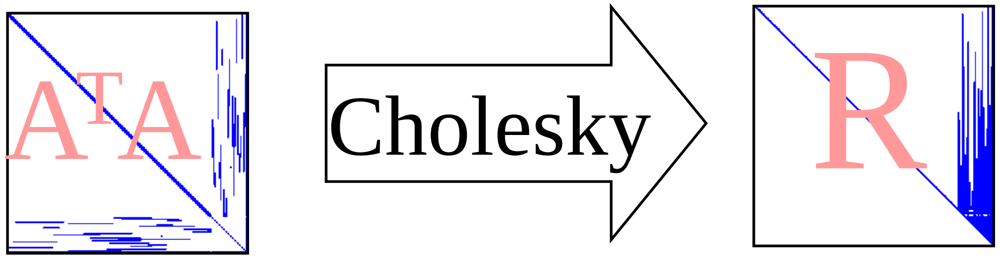
Note that in practice Q is never explicitly formed.

Matrix Factorization

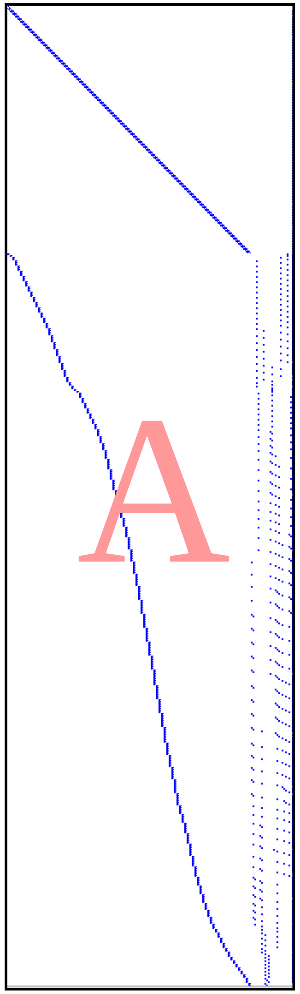
- QR on A: Numerically more stable



- Cholesky on A^TA: Faster



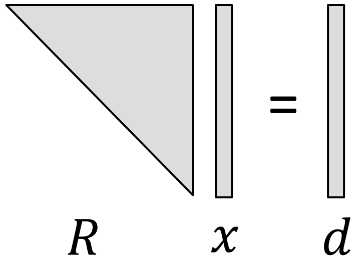
Solving the Sparse Linear Least-Squares Problem via QR

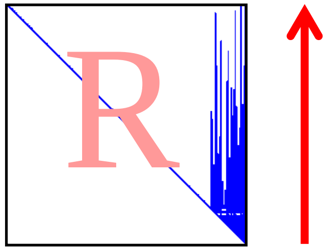


Measurement Jacobian

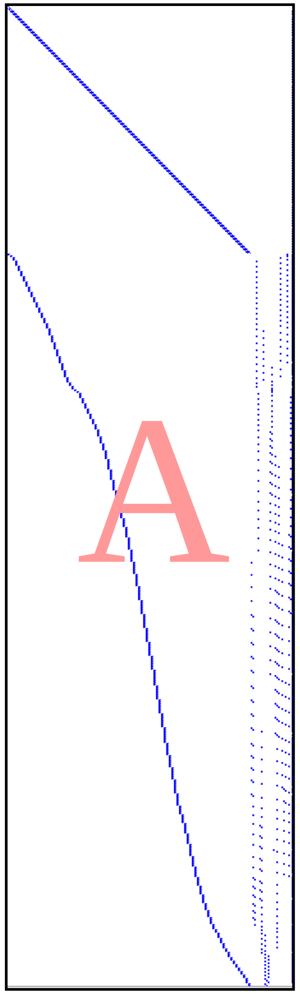
Solve: $\text{argmin}_\theta \|A\theta - b\|^2 = \text{argmin}_\theta \|Rx - d\|^2 + \|e\|^2$

$$Rx = d$$





Solving the Sparse Linear Least-Squares Problem via Cholesky

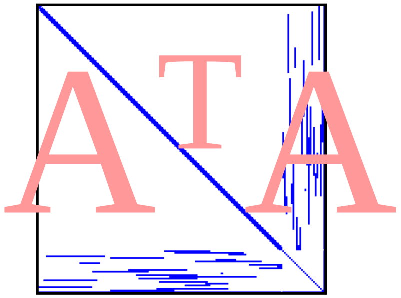


Measurement Jacobian

Solve: $\operatorname{argmin}_\theta \|A\theta - b\|^2$

Normal equations

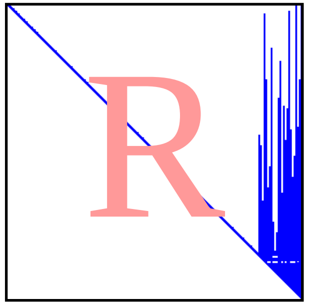
$$A^T A\theta = A^T b$$



Information matrix

Matrix factorization

$$A^T A = R^T R$$



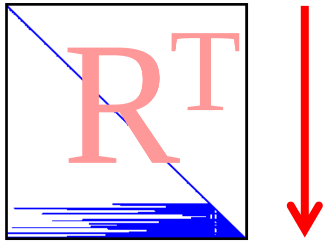
Square root information matrix

Solving by Forward and Back substitution (Cholesky)

After factorization: $R^T R\, x = A^T b$

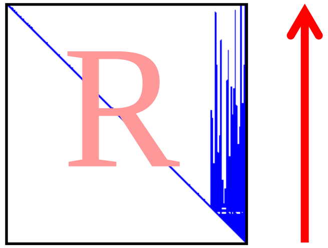
- Forward substitution

  $R^T y = A^T b$, solve for y



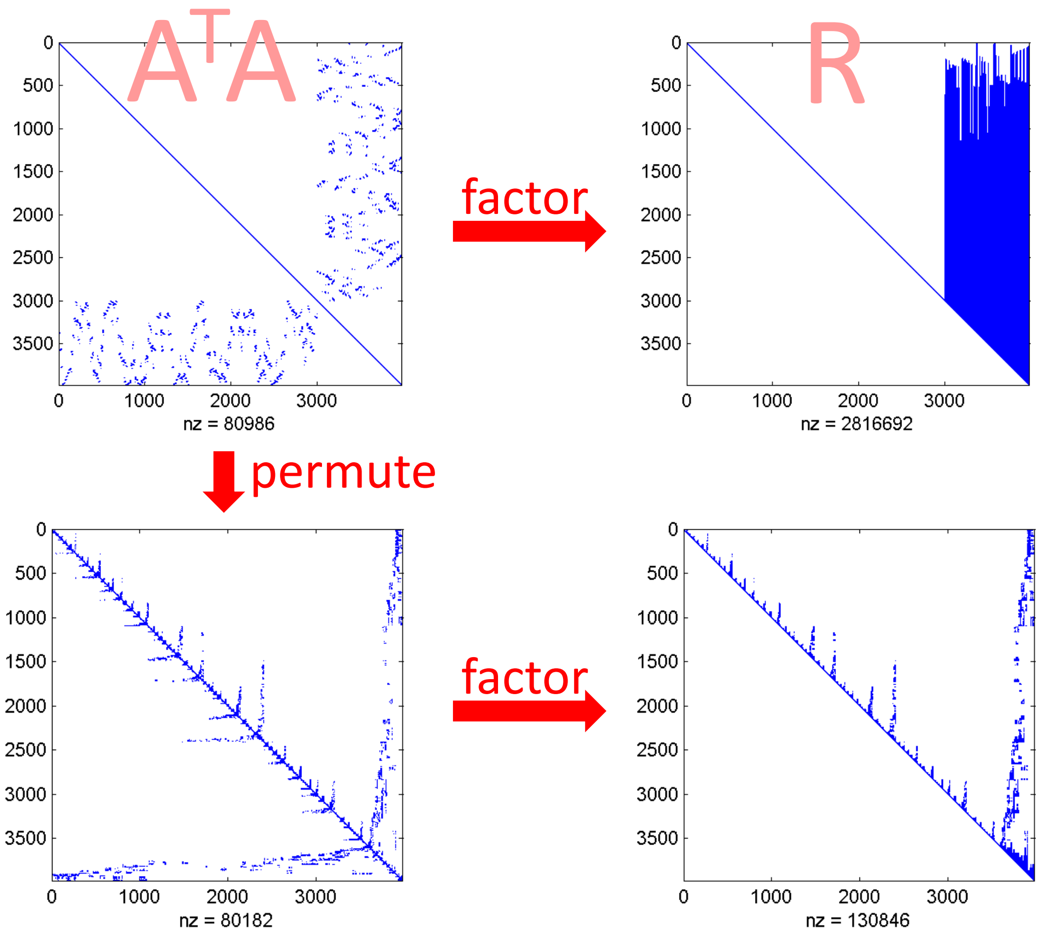
- Backsubstitution

  $R\, x = y$, solve for x



Retaining Sparsity: Variable Ordering

Fill-in depends on elimination order:



Default ordering (poses, landmarks)

Ordering based on COLAMD heuristic [Davis04]
(best order: NP hard)

Sparse Factorization Example

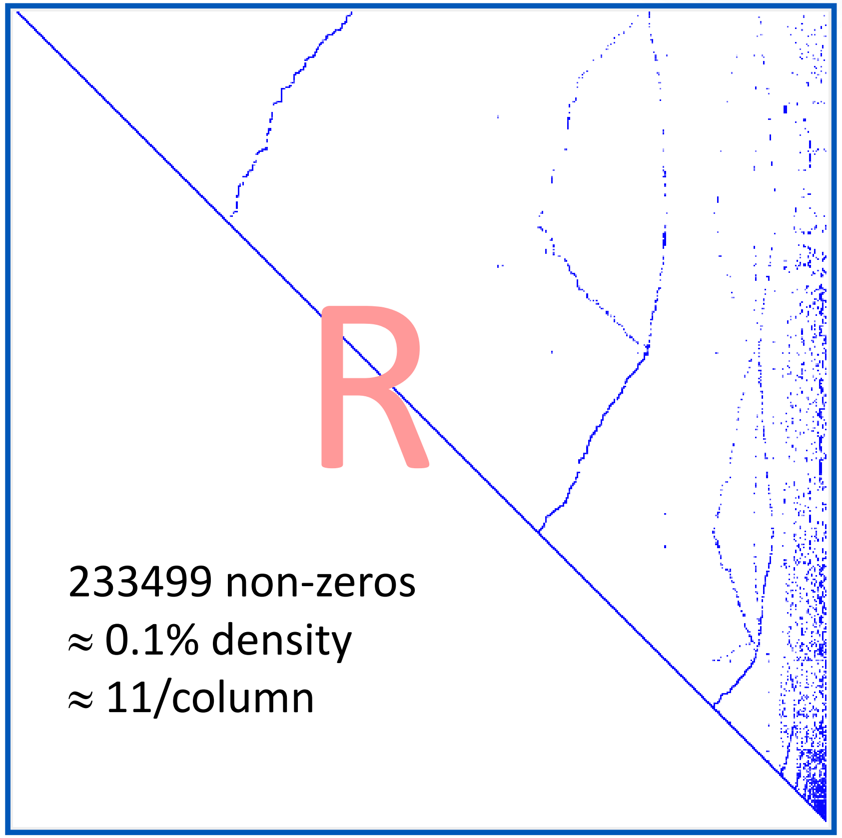
[Kaess et al., TRO 08]

Example from real sequence:

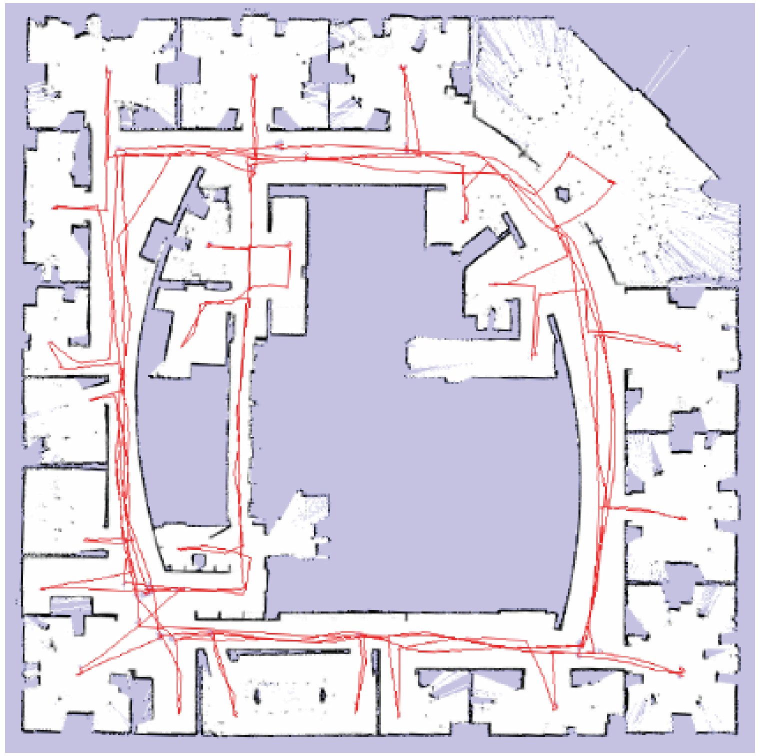
Square root inf. matrix

Side length: 21000 variables

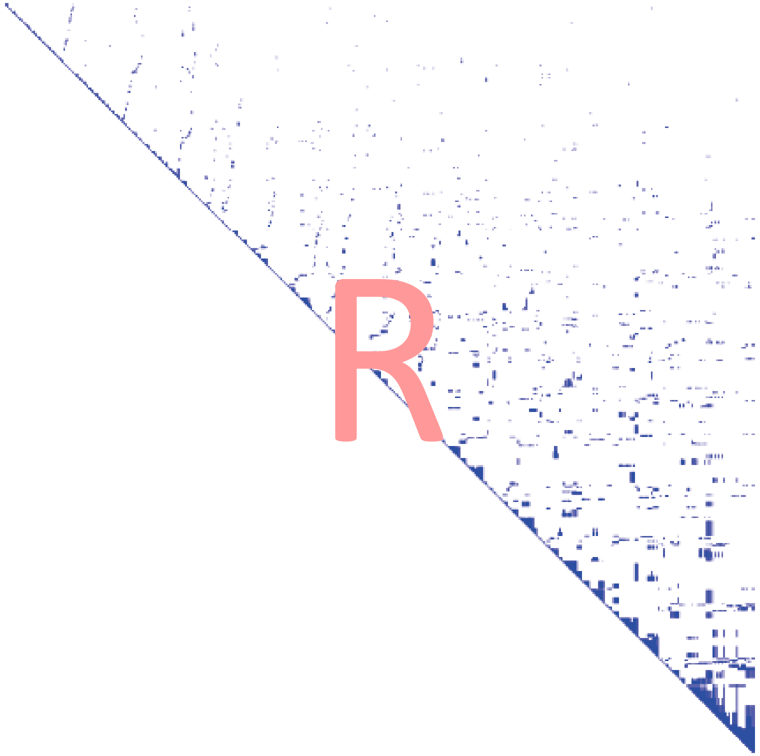
Dense: 1.7GB, sparse: 1MB



Example 2 - Standard Intel Dataset



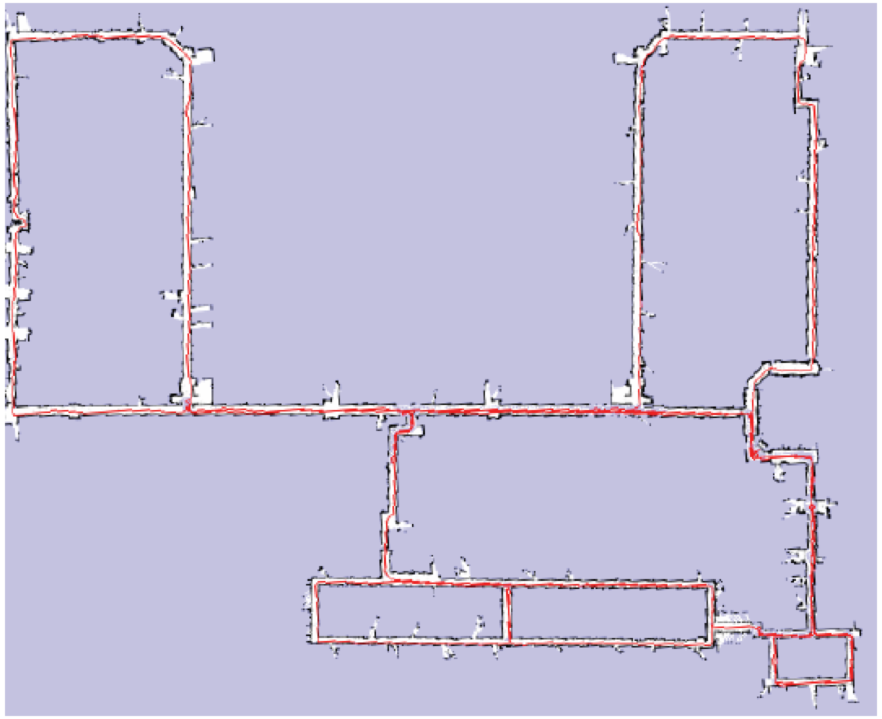
(b) Final trajectory and evidence grid map.



(c) Final R factor with side length 2730.

910 poses, 4453 constraints

Example 3 - MIT Killian Court Dataset



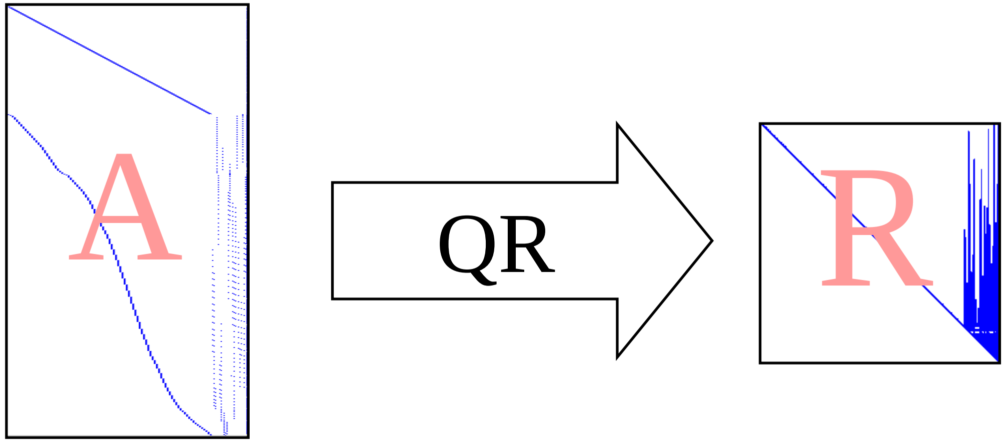
(b) Final trajectory and evidence grid map.

(c) Final R factor with side length 5823.

1941 poses, 2190 constraints

Questions

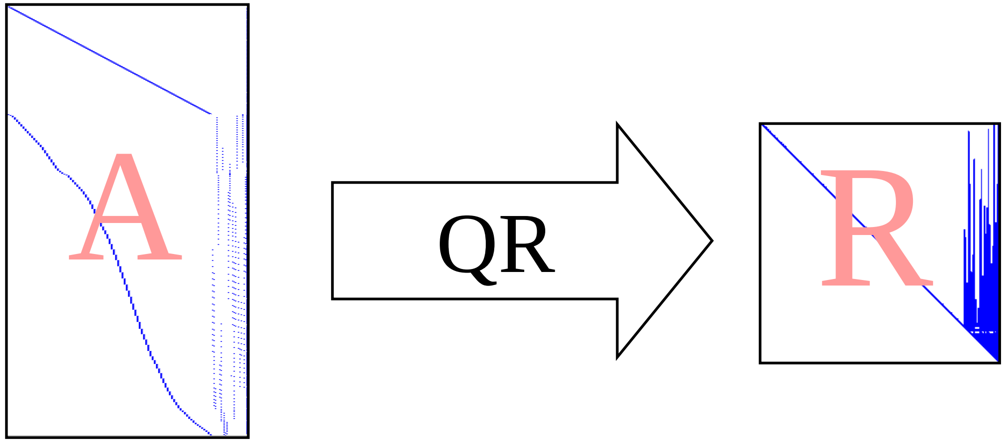
QR on A:



- Does the order of the rows of A impact fill-in?

Questions

QR on A:

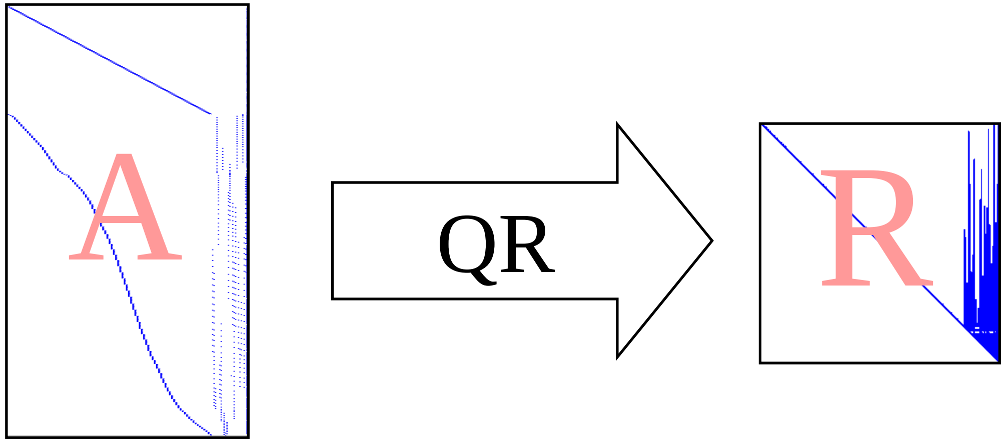


- Does the order of the rows of A impact fill-in?

    No

Questions

QR on A:



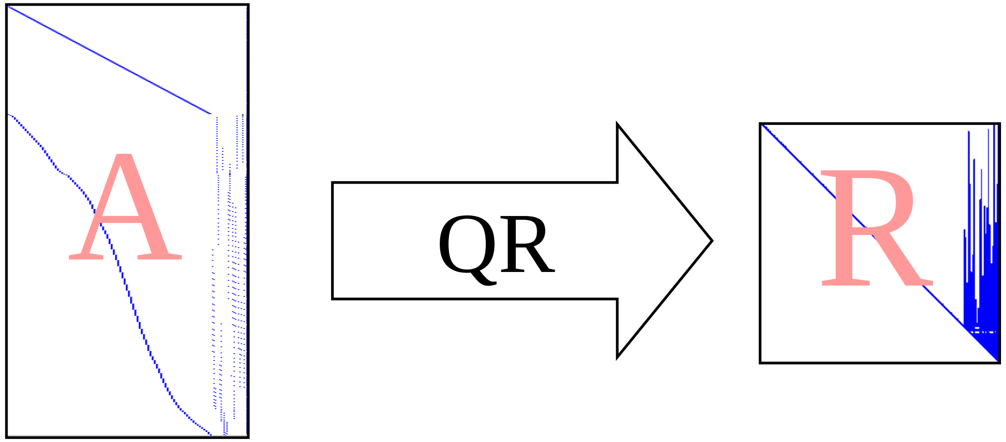
- Does the order of the rows of A impact fill-in?

  No

- Does the order of the columns of A impact fill-in?

Questions

QR on A:



- Does the order of the rows of A impact fill-in?

    No

- Does the order of the columns of A impact fill-in?

    Yes, the order will influence the fill-in in R and therefore efficiency!

Nonlinear -> Linear Least Squares

Taylor series expansion:

$$h_i(X_i) = h_i(X_i^0 + \Delta_i) \approx h_i(X_i^0) + H_i\Delta_i$$

Measurement Jacobian: $H_i \triangleq \left.\frac{\partial h_i(X_i)}{\partial X_i}\right|_{X_i^0}$

State update vector: $\Delta_i \triangleq X_i - X_i^0$

Linear least-squares problem:

$$\begin{aligned}\Delta^* &= \underset{\Delta}{\operatorname{argmin}} \sum_i \left\| h_i(X_i^0) + H_i\Delta_i - z_i \right\|_{\Sigma_i}^2 \\ &= \underset{\Delta}{\operatorname{argmin}} \sum_i \left\| H_i\Delta_i - \underbrace{\left\{ z_i - h_i(X_i^0) \right\}}_{\text{Prediction error}} \right\|_{\Sigma_i}^2\end{aligned}$$

Simplifying to Quadratic Form

Original term with Mahalanobis Distance:

$$\begin{aligned}\Delta^* &= \underset{\Delta}{\operatorname{argmin}} \sum_i \left\| h_i(X_i^0) + H_i\Delta_i - z_i \right\|_{\Sigma_i}^2 \\ &= \underset{\Delta}{\operatorname{argmin}} \sum_i \left\| H_i\Delta_i - \left\{ z_i - h_i(X_i^0) \right\} \right\|_{\Sigma_i}^2\end{aligned}$$

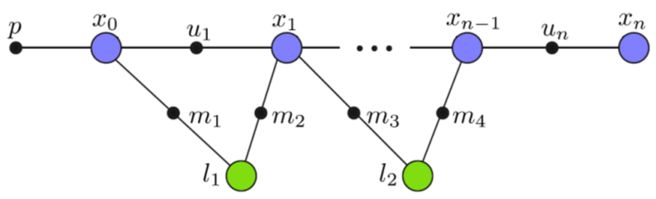
Conversion to l2 Norm:

$$\begin{aligned}A_i &= \Sigma_i^{-1/2} H_i \\ b_i &= \Sigma_i^{-1/2} \left(z_i - h_i(X_i^0) \right)\end{aligned}$$

Quadratic form:

$$\begin{aligned}\Delta^* &= \underset{\Delta}{\operatorname{argmin}} \sum_i \| A_i\Delta_i - b_i \|_2^2 \\ &= \underset{\Delta}{\operatorname{argmin}} \| A\Delta - b \|_2^2\end{aligned}$$

SLAM as a **Sparse** Least-Squares Problem



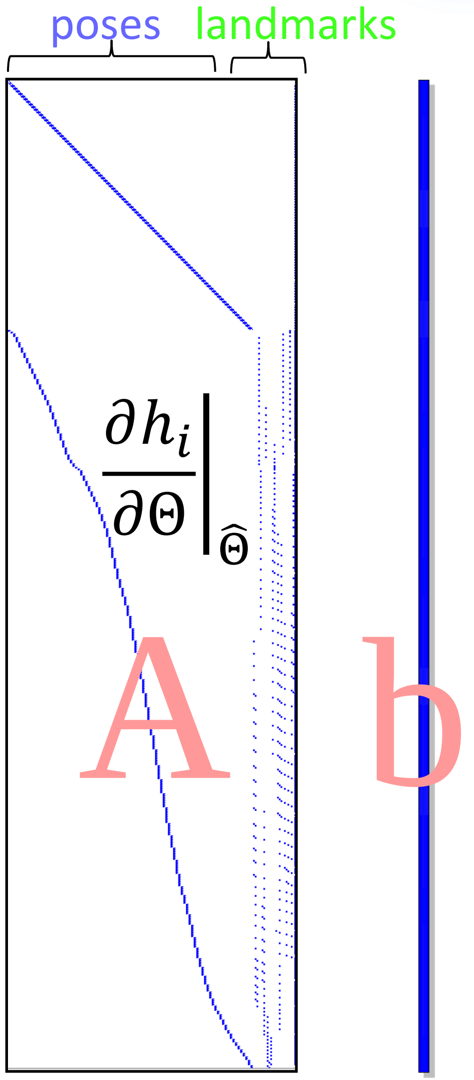
$$\text{argmax}_\Theta \prod_{z \in Z} p\left(z|\Theta\right)$$

Gaussian noise

$$\text{argmin}_\Theta \sum_i \|h_i(\Theta) - z_i\|^2$$

$$\text{argmin}_\theta \|A\theta - b\|^2$$

Sparse measurement Jacobian



Steepest Descent

Cost function:

$$g(X) \triangleq \sum_i \|h_i(X_i) - z_i\|^2_{\Sigma_i}$$

$$g(X) \approx \|A(X - X^t) - b\|_2^2$$

Steepest descent step:

$$\Delta_{sd} = -\alpha \, \nabla g\left(X\right)\big|_{X=X^t}$$

gradient: $\nabla g\left(X\right)\big|_{X=X^t} = -2A^T b$

Gauss-Newton

Cost function:

$$g(X) \approx \|A(X - X^t) - b\|_2^2$$

Gauss-Newton step:

$$A^T A \Delta_{gn} = A^T b$$

Levenberg-Marquardt

Levenberg:

$$(A^T A + \lambda I)\Delta_{lb} = A^T b$$

Levenberg-Marquardt:

$$\left(A^T A + \lambda \operatorname{diag}(A^T A)\right) \Delta_{lm} = A^T b$$

Levenberg-Marquardt

Algorithm 2.1 The Levenberg-Marquardt algorithm

function LM($g(), X^0$) ▷ quadratic cost function $g()$, ▷ initial estimate X^0

  $\lambda = 10^{-4}$

  $t = 0$

  repeat

    $A, b \leftarrow$ linearize $g(X)$ at X^t

    $\Delta \leftarrow$ solve $\left(A^T A + \lambda \operatorname{diag}(A^T A)\right) \Delta = A^T b$

    if $\mathrm{g}(X^t + \Delta) < \mathrm{g}(X^t)$ **then**

      $X^{t+1} = X^t + \Delta$ ▷ accept update

      $\lambda \leftarrow \lambda / 10$

    else

      $X^{t+1} = X^t$ ▷ reject update

      $\lambda \leftarrow \lambda * 10$

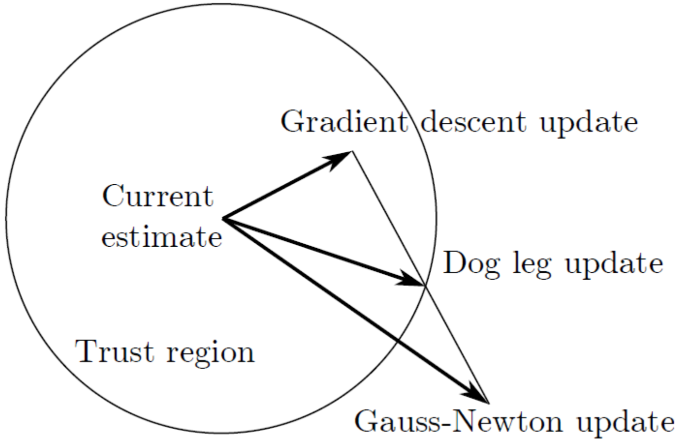
    $t \leftarrow t + 1$

  until convergence

  return X^t ▷ return latest estimate

Powell's Dog-Leg Algorithm

Key idea: Explicitly maintain a trust region



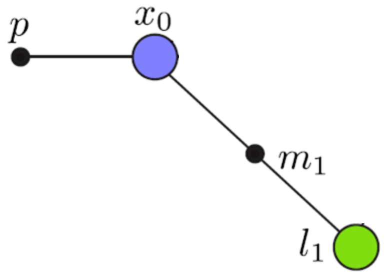
Given a trust region of radius Δ, Powell's dog leg method selects the update step $\boldsymbol{\delta_k}$ as equal to:

- $\boldsymbol{\delta_{gn}}$, if the Gauss–Newton step is within the trust region ($\|\boldsymbol{\delta_{gn}}\| \leq \Delta$);
- $\dfrac{\Delta}{\|\boldsymbol{\delta_{sd}}\|}\boldsymbol{\delta_{sd}}$ if both the Gauss–Newton and the steepest descent steps are outside the trust region ($t\|\boldsymbol{\delta_{sd}}\|$);
- $t\boldsymbol{\delta_{sd}} + s\left(\boldsymbol{\delta_{gn}} - t\boldsymbol{\delta_{sd}}\right)$ with s such that $\|\boldsymbol{\delta}\| = \Delta$, if the Gauss–Newton step is outside the trust region but the steepest descent step is inside (dog leg step).[1]

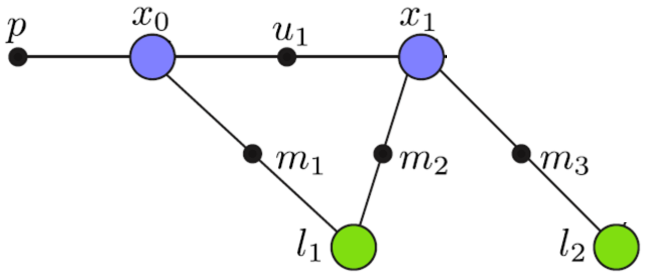
Wikipedia.com

Online SLAM is a Sequential Estimation Problem

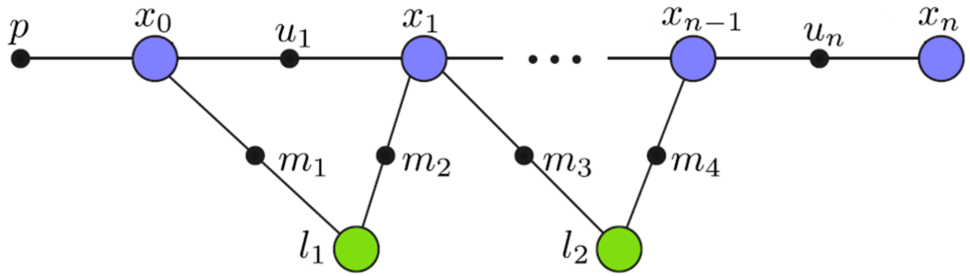
t=0



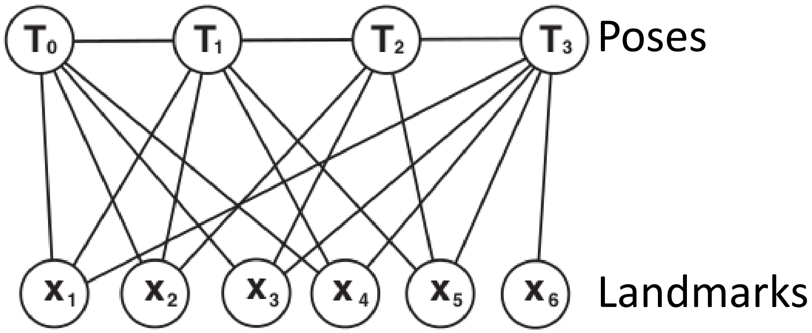
t=1



t=n-1



Full SLAM (Computer Vision: Bundle Adjustment)

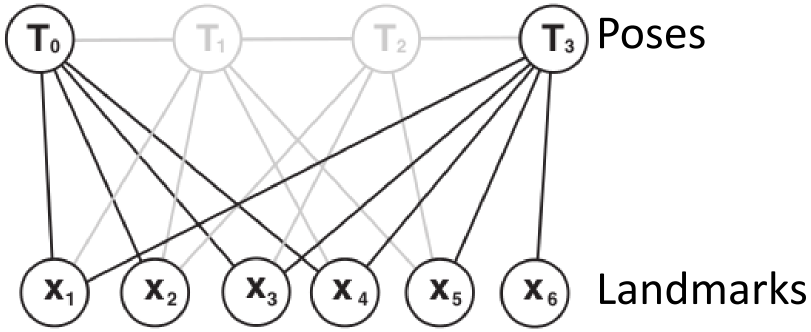


From Strasdat et al, 2011 IVC "Visual SLAM: Why filter?"

- Graph grows with time:
  - Have to solve a sequence of increasingly larger problems
  - Will become too expensive even for sparse Cholesky

F. Dellaert and M. Kaess, "Square Root SAM: Simultaneous localization and mapping via square root information smoothing," IJRR 2006

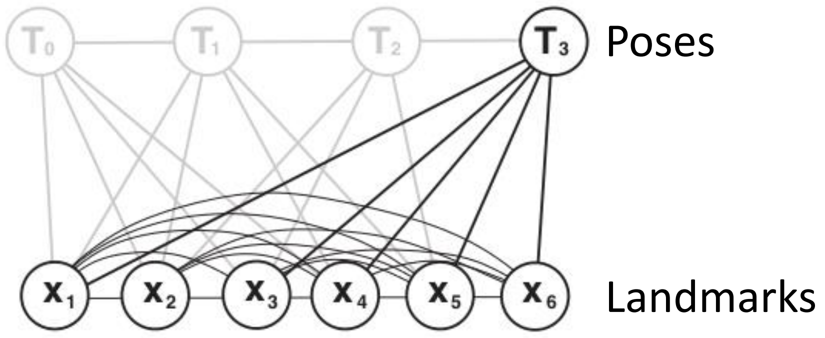
Keyframe SLAM



- Drop subset of poses to reduce density/complexity
- Only retain "keyframes" necessary for good map

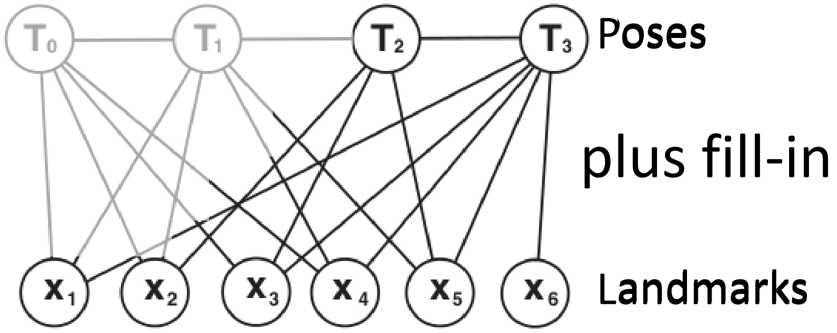
- Complexity still grows with time, just slower

Filter



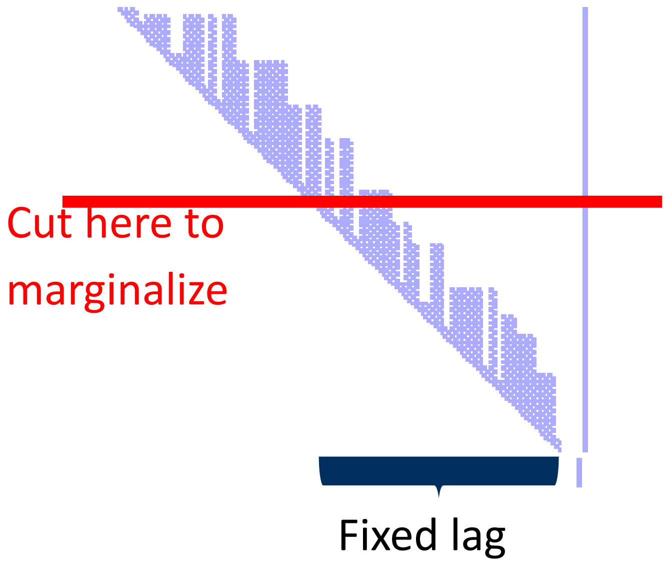
- Keyframe idea not applicable: map would fall apart
- Instead, marginalize out previous poses
  - Extended Kalman Filter (EKF)
- Problems when used for SLAM:
  - All landmarks become fully connected -> **expensive**
  - Relinearization not possible -> **inconsistent**

Fixed-lag Smoothing



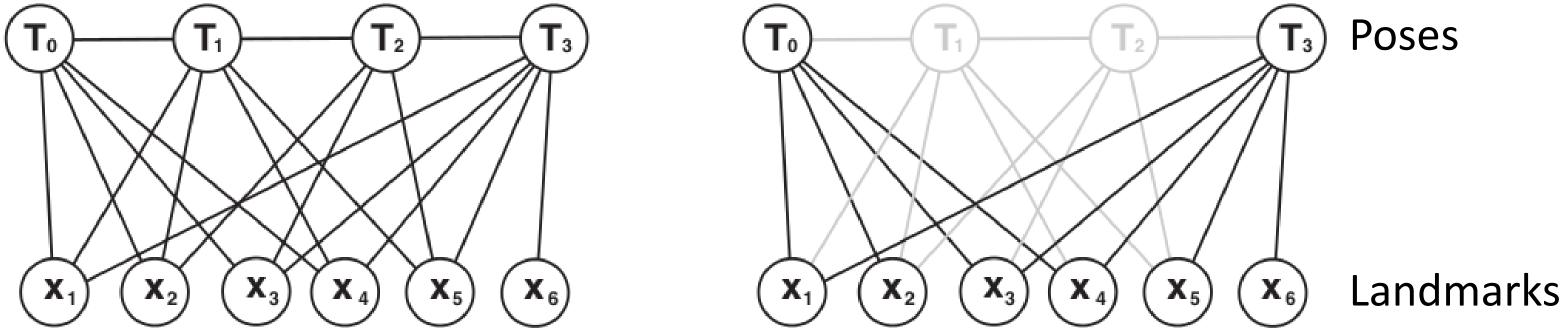
plus fill-in

- Marginalize out all but last n poses and connected landmarks
  - Relinearization possible
- Linear case
- Nonlinear (with some restrictions)



Is Cheap and Exact Achievable?

- Back to full BA and keyframes:



- New information is added to the graph
- Older information does not change
- Can be exploited to obtain an efficient solution!

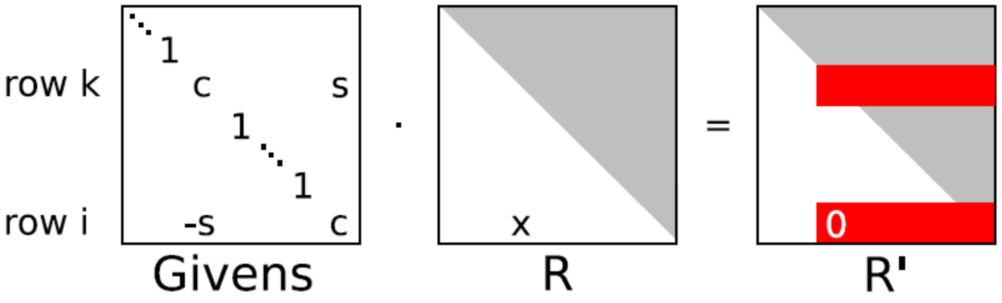
Incremental Smoothing and Mapping (iSAM)

Solving a growing system:

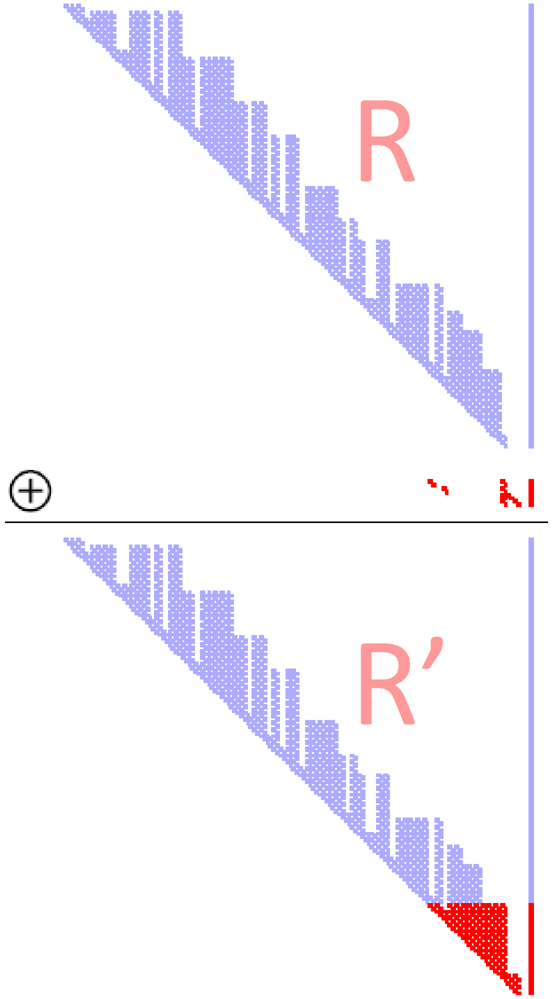
- R factor from previous step
- How do we add new measurements?

Key idea:

- Append to existing matrix factorization
- “Repair” using Givens rotations



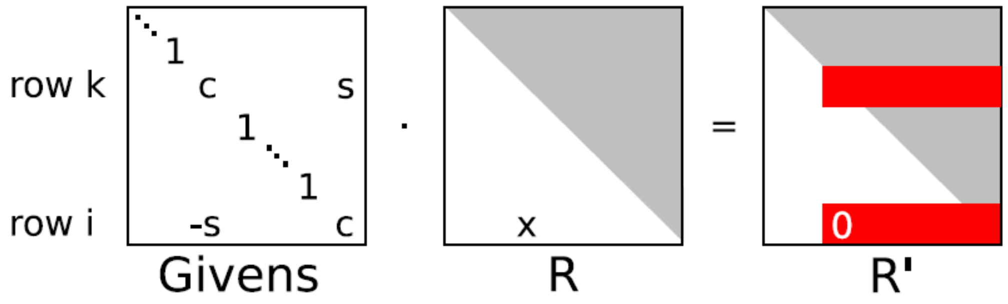
New measurements ->



QR Factorization: Householder Reflections

- On the board

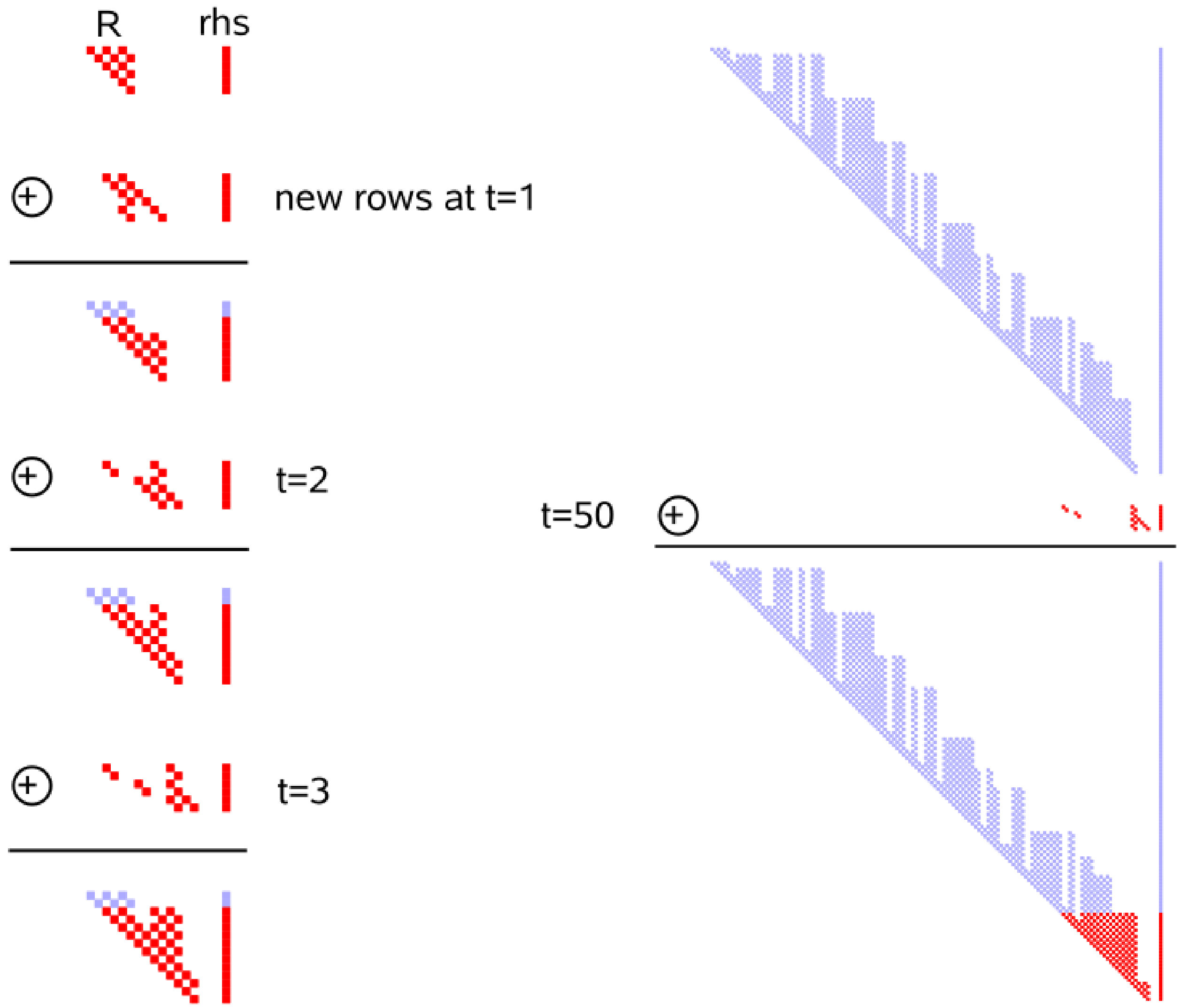
Givens Rotations



$$(\cos\phi, \sin\phi) = \begin{cases} (1,0) & \text{if } \beta = 0 \\ \left(\frac{-\alpha}{\beta\sqrt{1+\left(\frac{\alpha}{\beta}\right)^2}}, \frac{1}{\sqrt{1+\left(\frac{\alpha}{\beta}\right)^2}} \right) & \text{if } |\beta| > |\alpha| \\ \left(\frac{1}{\sqrt{1+\left(\frac{\beta}{\alpha}\right)^2}}, \frac{-\beta}{\alpha\sqrt{1+\left(\frac{\beta}{\alpha}\right)^2}} \right) & \text{otherwise} \end{cases}$$

where $\alpha := a_{kk}$ and $\beta := a_{ik}$.

iSAM Updates



[Kaess et

Incremental Smoothing and Mapping (iSAM)

Update and solution are O(1)

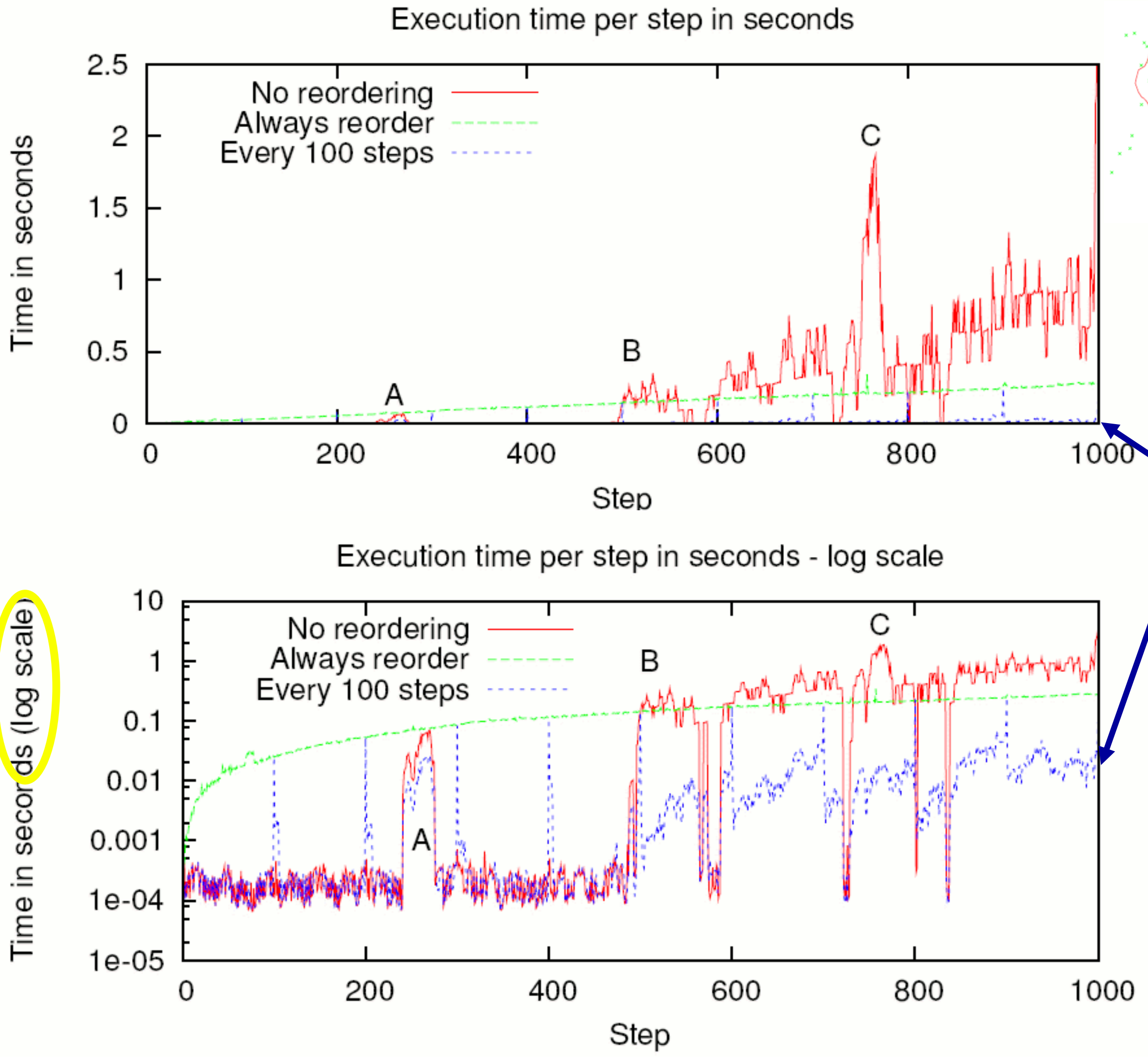
Are we done?

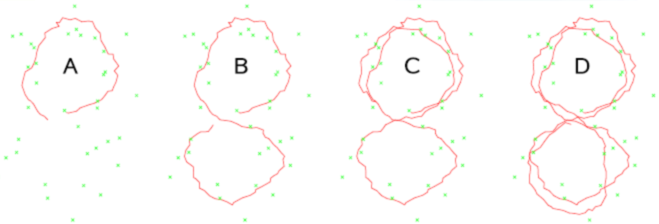
SLAM is nonlinear…

iSAM requires periodic batch factorization to relinearize

Also: loop closures cause fill-in! -> Reordering

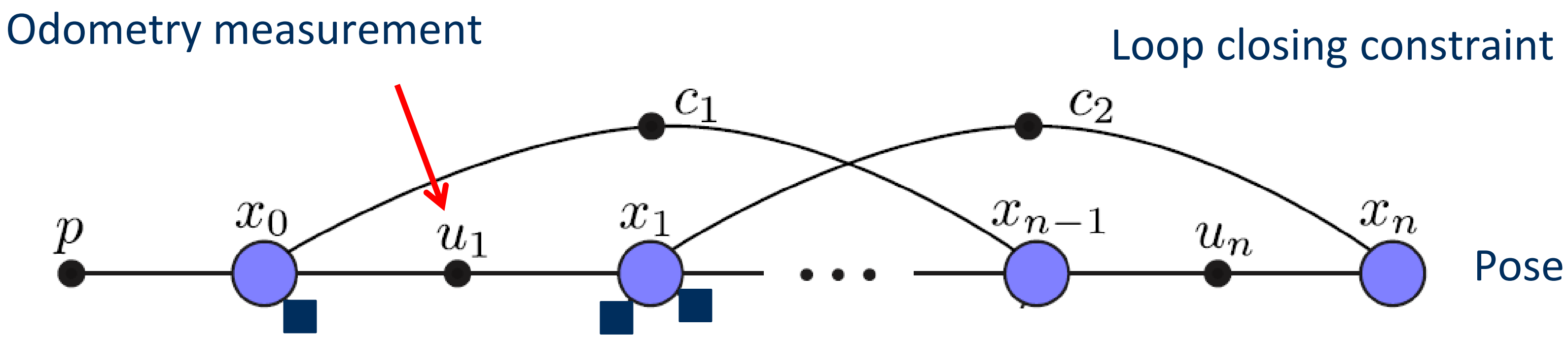
Periodic Variable Reordering – Timing





Combines advantages of incremental SAM and batch reordering

Pose Graph SLAM - Scalability



Smoothing: Grows unboundedly in time

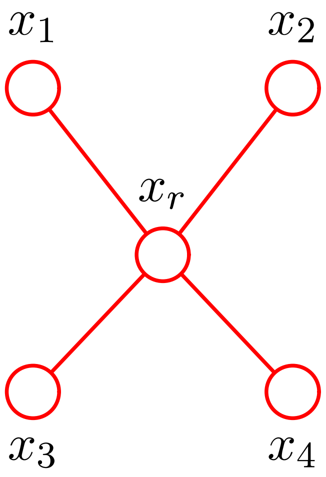
Should only depend on explored space

Solution: Reduced Pose Graph

Johannsson, Kaess, Fallon, Leonard (ICRA 13)

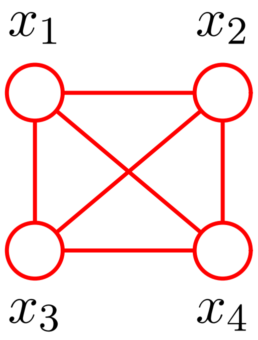
Pose Graph Reduction

Reduction by marginalization



Remove x_r

$$p(x_1, x_2, x_3, x_4) = \int p(x_1, x_2, x_3, x_4, x_r)\,dx_r$$

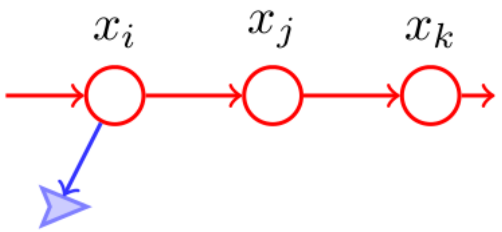


Avoiding dense graphs:

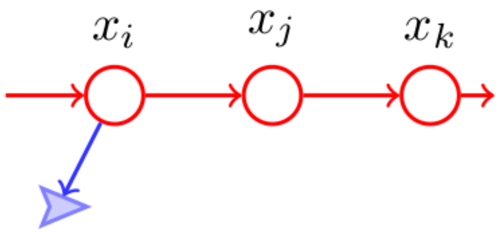
- Kretzschmar et al. (IROS 11): approximate marginal using Chow-Liu tree
- Eade et al. (IROS 10): limit degree of nodes and remove edges
- Carlevaris-Bianco, Kaess, Eustice (TRO 14): consistent sparsification
- **Our approach:** keeping the graph simple during construction

Reduced Pose Graph (step n)

In general, not revisiting exactly same poses

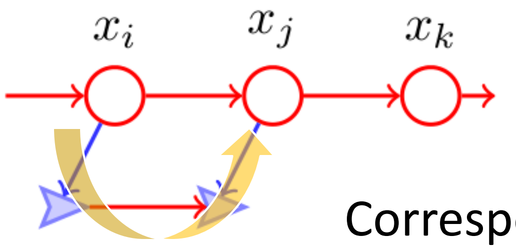


Standard pose graph:



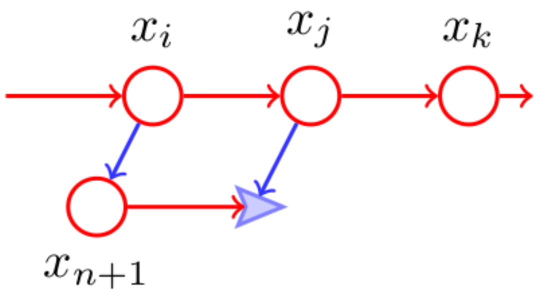
Reduced Pose Graph (step n+1)

In general, not revisiting exactly same poses



Corresponds to a constraint between x_i and x_j

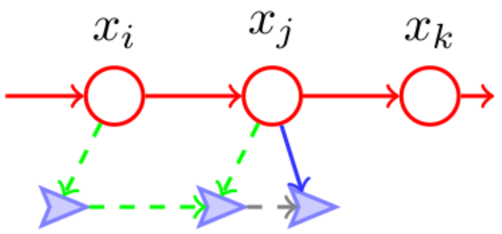
Standard pose graph:



New pose is added

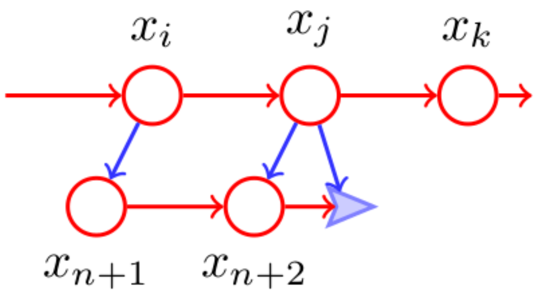
Reduced Pose Graph (step n+2)

Avoiding inconsistency



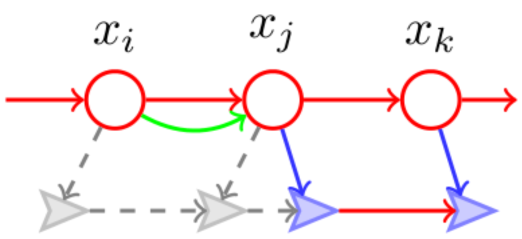
Second loop closure to x_j to avoid double use of constraint

Standard pose graph:



Reduced Pose Graph (step n+3)

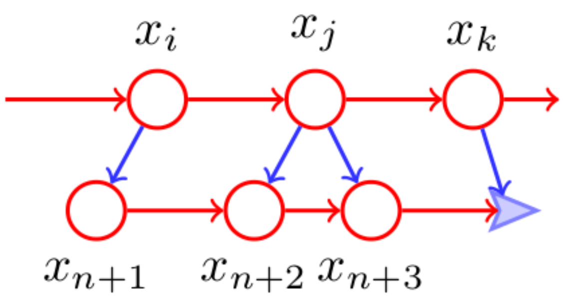
Avoiding inconsistency



Constraint between x_i and x_j added

Omitting short odometry links
similar to ESEIF by Walter 07

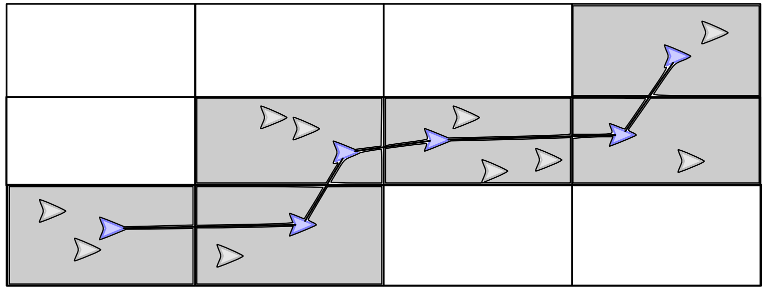
Standard pose graph:



Marginalization instead would lead to fully correlated pose graph !!

Partitioning

- How to know when to add a new pose?
- Partitioning schemes
  - Regular grid (x, y, heading)
  - Based on visibility (view frustum)
  - Based on feature overlap (typically done for keyframes)
- Choice of scheme depends on the sensors and motion



MIT Stata Center Data Set

Publically available: http://projects.csail.mit.edu/stata/

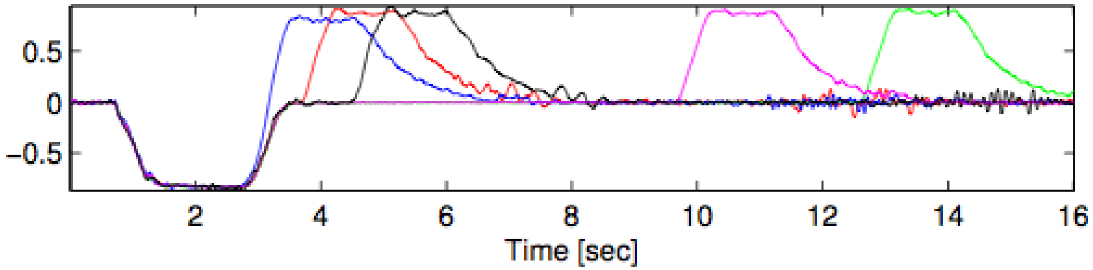
- IJRR data paper (Fallon, Johannsson, Kaess, Leonard)
- Duration: 18 months
- Operation time: 38 hours
- Distance travelled: 42 km (26 miles)
- Size: 2.3TB
- **Ground truth** by aligning laser scans with floor plans

Reduced Pose Graph – Second Floor

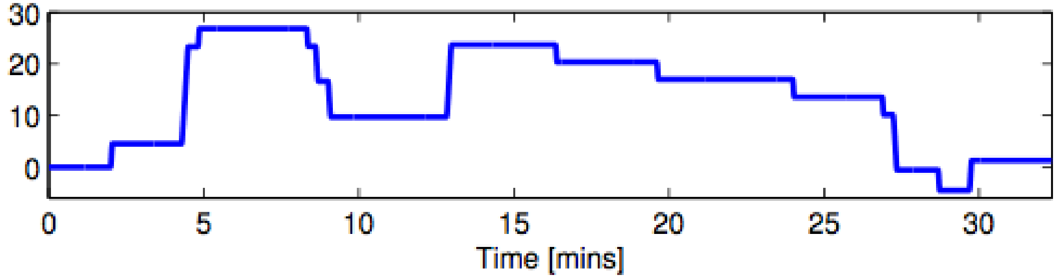
Multiple Floors – Elevator Transitions

- Accelerometer sufficient to determine floor

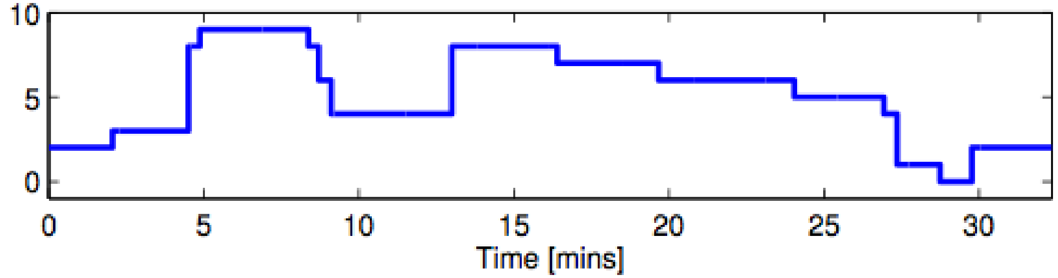
Filtered vertical acceleration during elevator ride



Height (m)



Floor Assignment



Reduced Pose Graph

Map of 10 floors

- Accelerometer used to detect elevator transitions
- iSAM optimizes RPG to achieve real-time