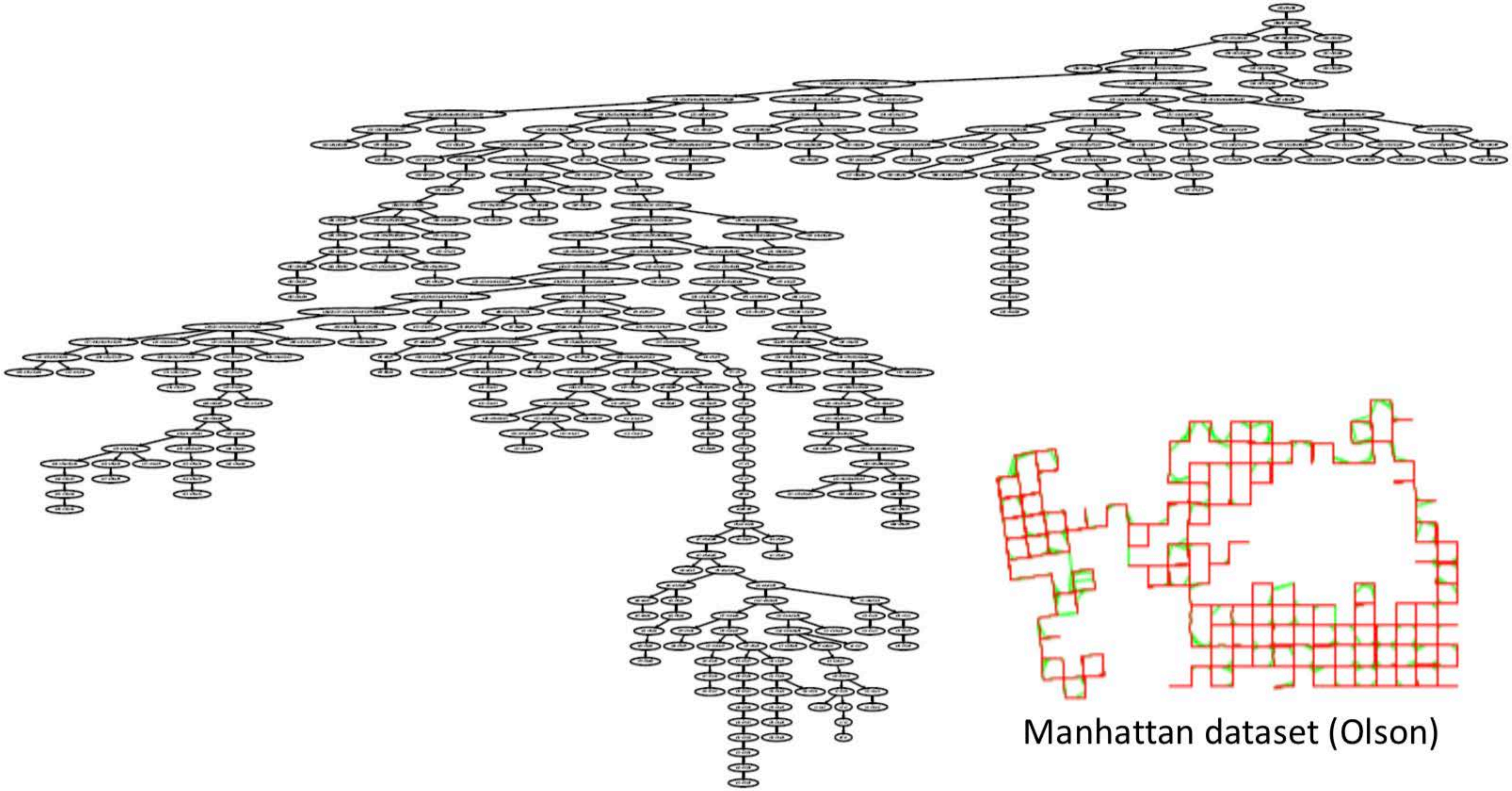


Manhattan dataset (Olson)

16.485: VNAV - Visual Navigation for Autonomous Vehicles

Luca Carlone

Lecture 28: Incremental Solvers for SLAM

(based on slides by Michael Kaess and Frank Dellaert)

Today: Incremental Solvers for SLAM

- iSAM (Incremental Smoothing And Mapping)

IEEE TRANSACTIONS ON ROBOTICS, MANUSCRIPT SEPTEMBER 7, 2008 1

T-RO 2008

iSAM: Incremental Smoothing and Mapping

Michael Kaess, *Student Member, IEEE,* Ananth Ranganathan, *Student Member, IEEE,* and Frank Dellaert, *Member, IEEE*

***Abstract*—We present incremental smoothing and mapping (iSAM), a novel approach to the simultaneous localization and mapping problem that is based on fast incremental matrix factorization. iSAM provides an efficient and exact solution by updating a QR factorization of the naturally sparse smoothing information matrix, therefore recalculating only the matrix entries that actually change. iSAM is efficient even for robot trajectories**

counterintuitive at first, because more variables are added to the estimation problem, the simplification arises from the fact that the smoothing information matrix is naturally sparse. In contrast, in filtering approaches the information matrix becomes dense when marginalizing out robot poses. As a consequence of applying smoothing, we are able to

- Bayes Tree and iSAM2

iSAM2: Incremental Smoothing and Mapping Using the Bayes Tree

Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John Leonard, and Frank Dellaert

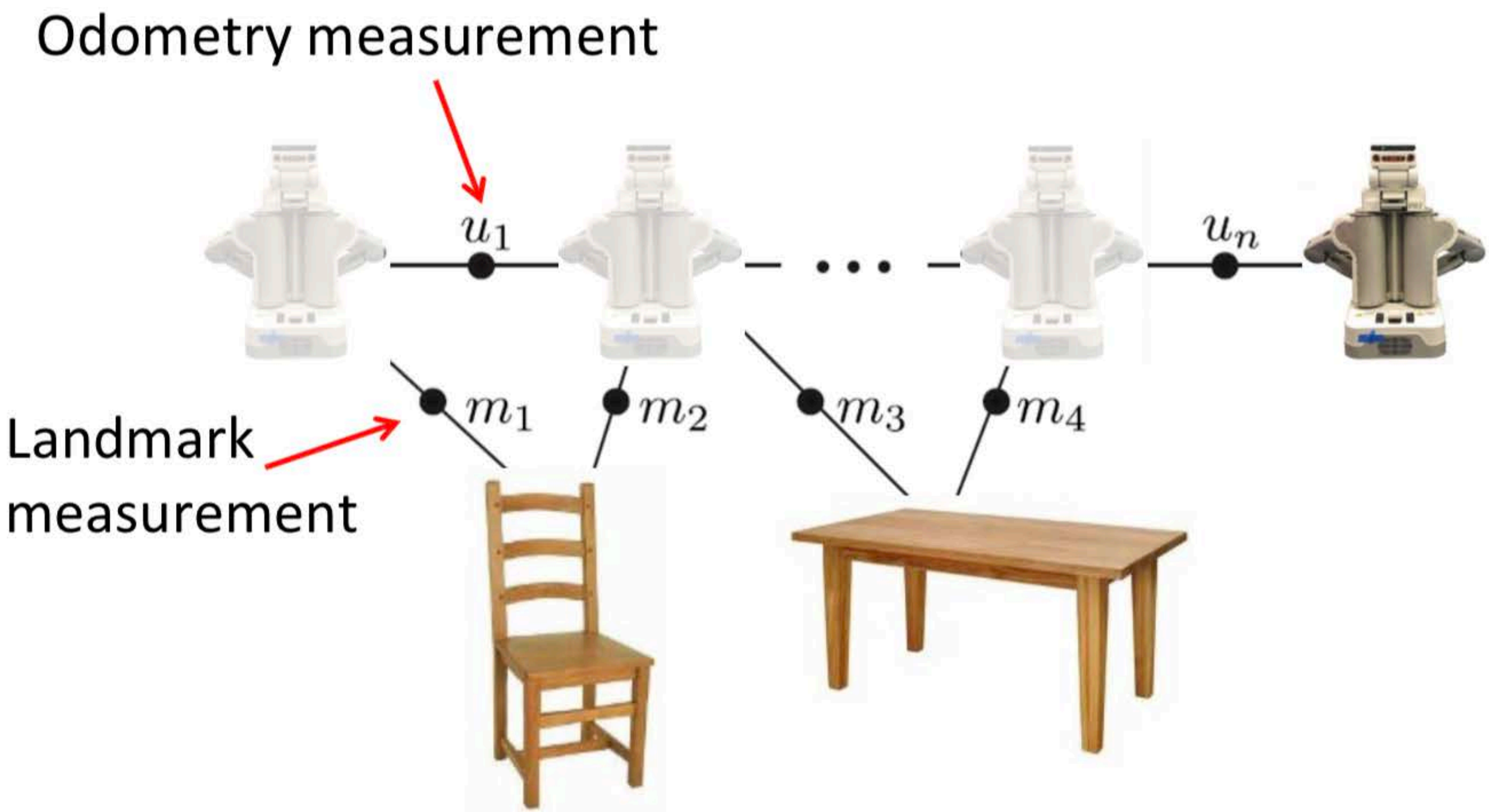
IJRR 2011

Abstract

We present a novel data structure, the Bayes tree, that provides an algorithmic foundation enabling a better understanding of existing graphical model inference algorithms and their connection to sparse matrix factorization methods. Similar to a clique tree, a Bayes tree encodes a factored probability density, but unlike the clique tree it is directed and maps more naturally to the square root information matrix of the simultaneous localization and mapping (SLAM) problem. In this paper, we highlight three insights provided by our new data structure. First, the Bayes tree provides a better understanding of the matrix factorization in terms of probability densities. Second, we show how the fairly abstract updates to a matrix factorization translate to a simple editing of the Bayes tree and its conditional densities. Third, we apply the Bayes tree to obtain a completely novel algorithm for sparse nonlinear incremental optimization, named iSAM2, which achieves improvements in efficiency through incremental variable re-ordering and fluid relinearization, eliminating the need for periodic batch steps. We analyze various properties of iSAM2 in detail, and show on a range of real and simulated datasets that our algorithm compares favorably with other recent mapping algorithms in both quality and efficiency.

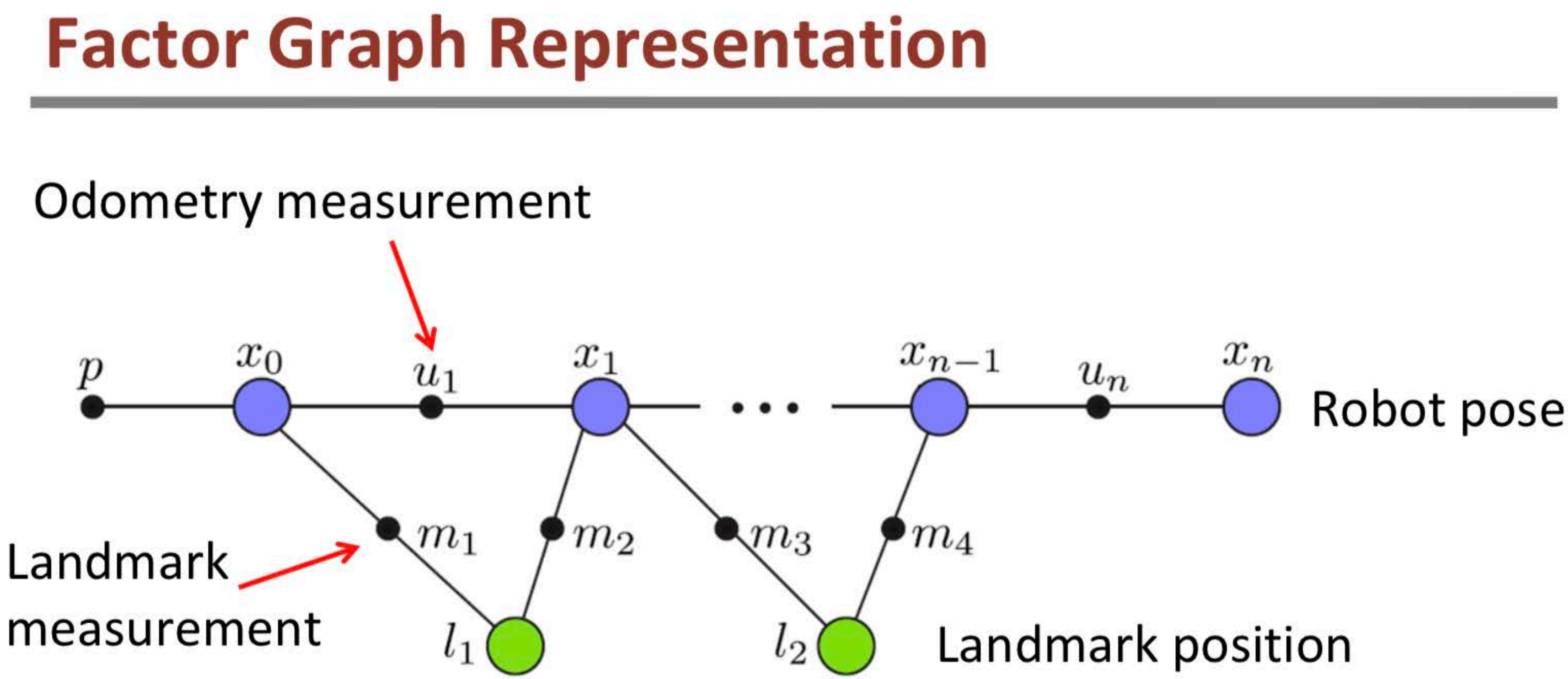
Keywords: graphical models, clique tree, junction tree, probabilistic inference, sparse linear algebra, nonlinear optimization, smoothing and mapping, SLAM

Solving SLAM Incrementally



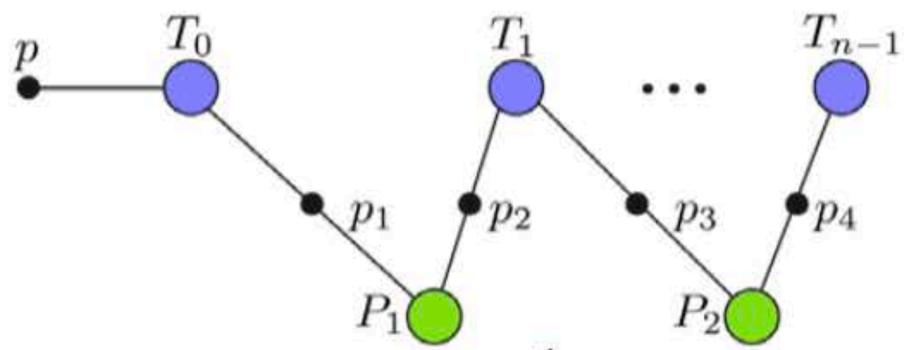
- At each time the robot collects new data and solves an optimization problem to compute the MAP estimate
- **Question**: can we avoid solving the optimization from scratch each time? How can we reuse computation?

Solving SLAM Incrementally



- At each time the robot collects new data and solves an optimization problem to compute the MAP estimate
- **Question**: can we avoid solving the optimization from scratch each time? How can we reuse computation?

Previously on VNAV: Batch Solvers

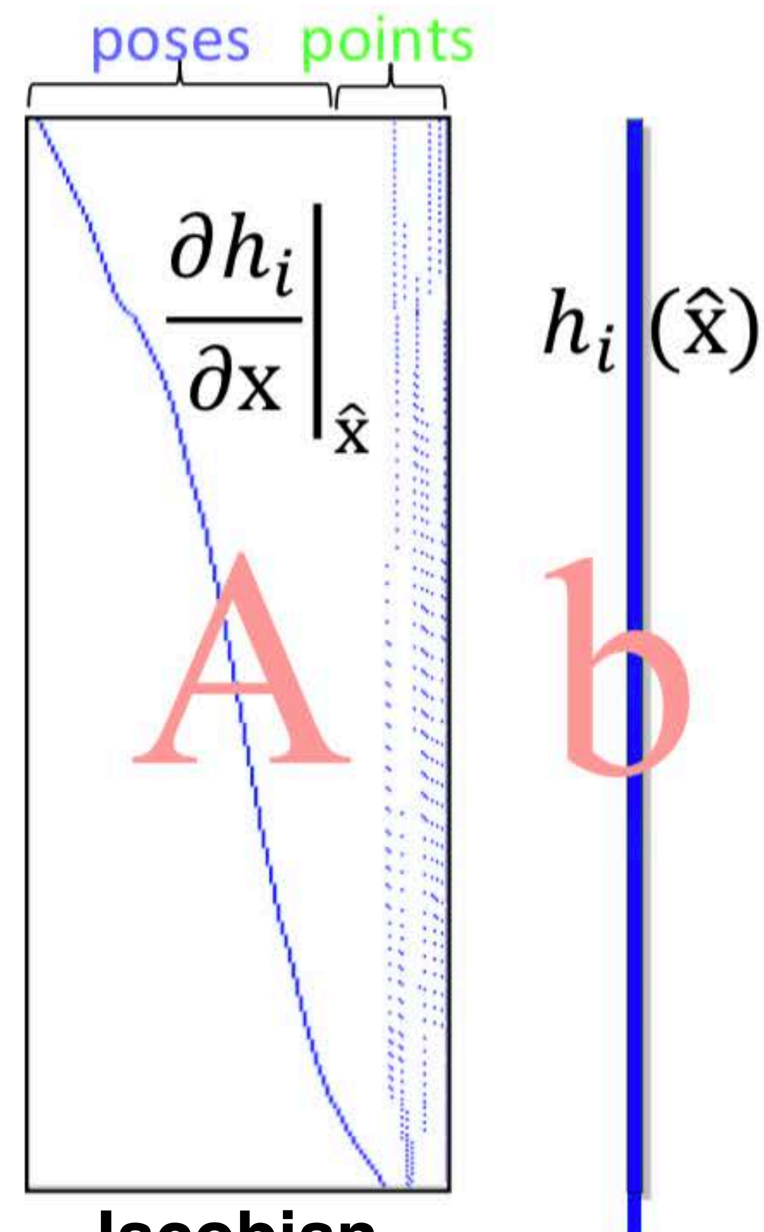


Gaussian noise

$$\mathrm{argmin}_{\mathrm{x}} \sum_i \|h_i(\mathrm{x})\|_{\Xi}{}^2$$

Repeatedly solve linearized system (GN)

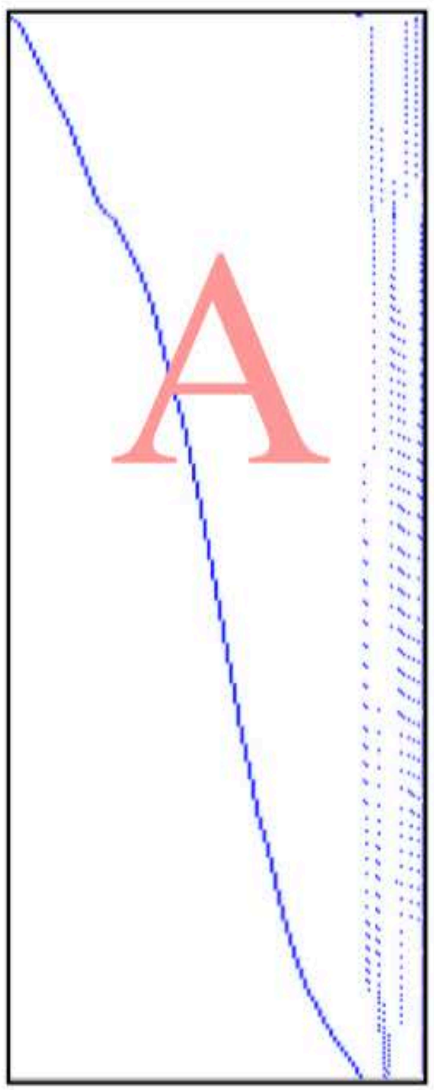
$$\mathrm{argmin}_x \|Ax - b\|^2$$



Jacobian

Previously on VNAV: Batch Solvers

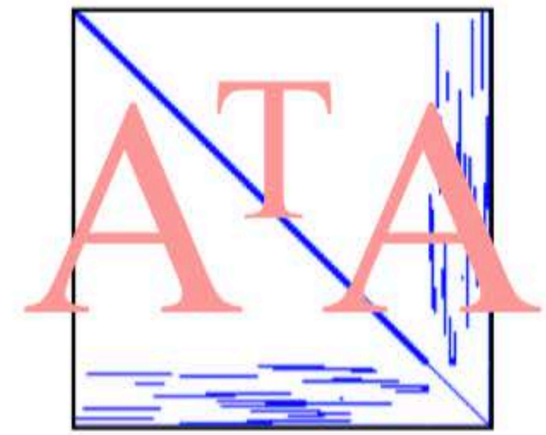
Solve: $\operatorname{argmin}_x \|Ax - b\|^2$



Measurement Jacobian

Normal equations

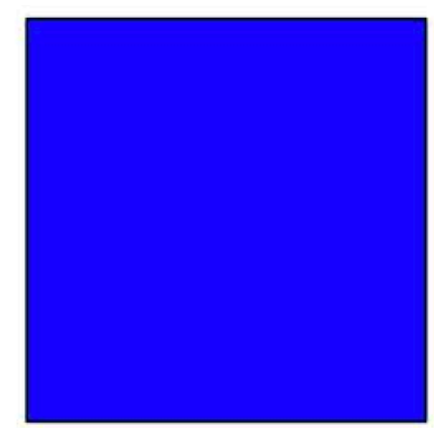
$$A^T A x = A^T b$$



Information matrix
(of the estimate)

From optimization lectures:

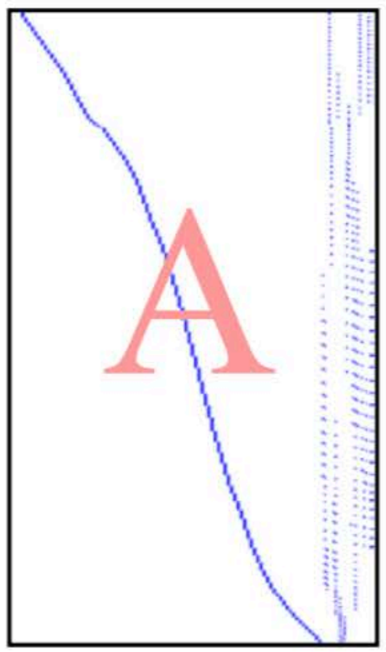
- we never compute the inverse (dense, O(n^3) computational complexity)
- We perform sparse QR or Cholesky factorizations to solve linear system



Covariance matrix
(of the estimate)

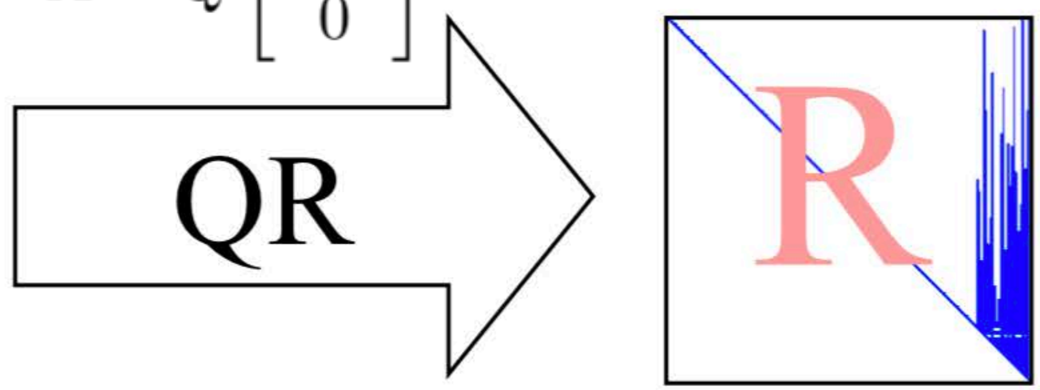
Previously on VNAV: Batch Solvers

- QR on A: Numerically More Stable



Orthogonal Upper-triangular square matrix

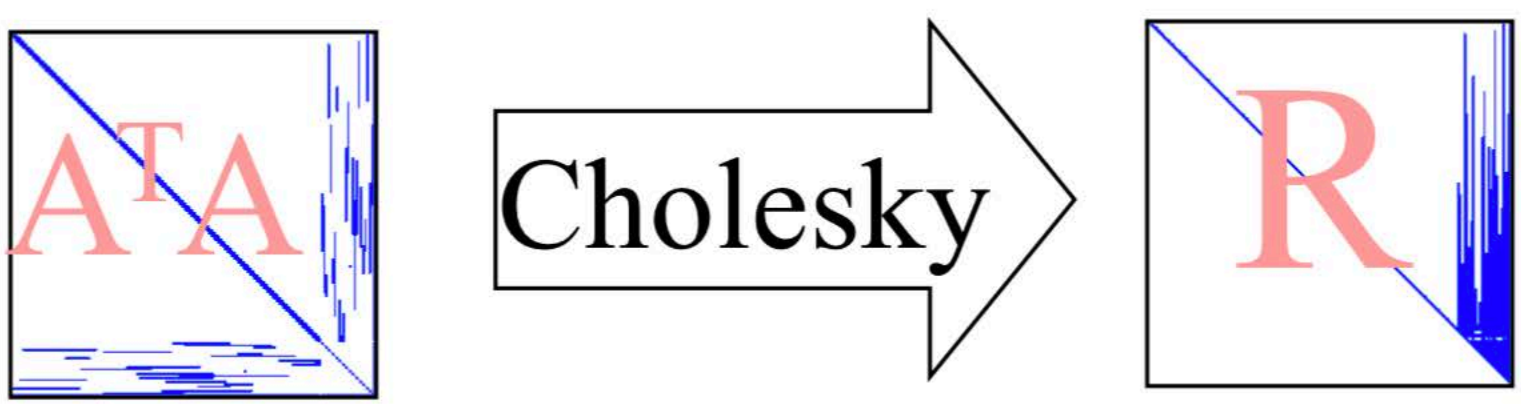
$$A = Q \begin{bmatrix} R \\ 0 \end{bmatrix}$$



Matrix factorization

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

- Cholesky on A^TA: Faster



Square Root Information Matrix (of the estimate)

Previously on VNAV: Batch Solvers

Using QR factorization:

$$\|A\boldsymbol{x}-\mathbf{b}\|^2 = \left\| Q \begin{bmatrix} R \\ 0 \end{bmatrix} \boldsymbol{x} - \mathbf{b} \right\|^2$$

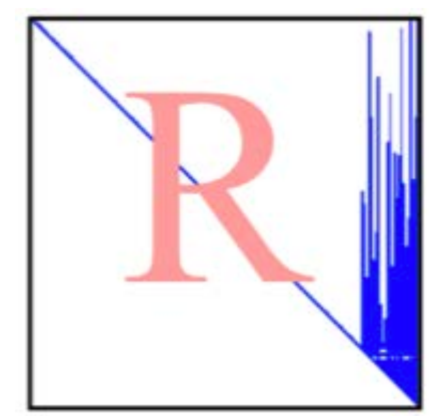
(Norm invariant to multiplication by Q)

$$= \left\| Q^T Q \begin{bmatrix} R \\ 0 \end{bmatrix} \boldsymbol{x} - Q^T \mathbf{b} \right\|^2$$

($Q^TQ = I$)

$$= \left\| \begin{bmatrix} R \\ 0 \end{bmatrix} \boldsymbol{x} - \begin{bmatrix} \mathbf{d} \\ \mathbf{e} \end{bmatrix} \right\|^2$$

$$= \|R\boldsymbol{x}-\mathbf{d}\|^2 + \|\mathbf{e}\|^2$$



QR transforms the problem into one where measurement Jacobian is a square upper-triangular matrix (note: ***Rx = d***)

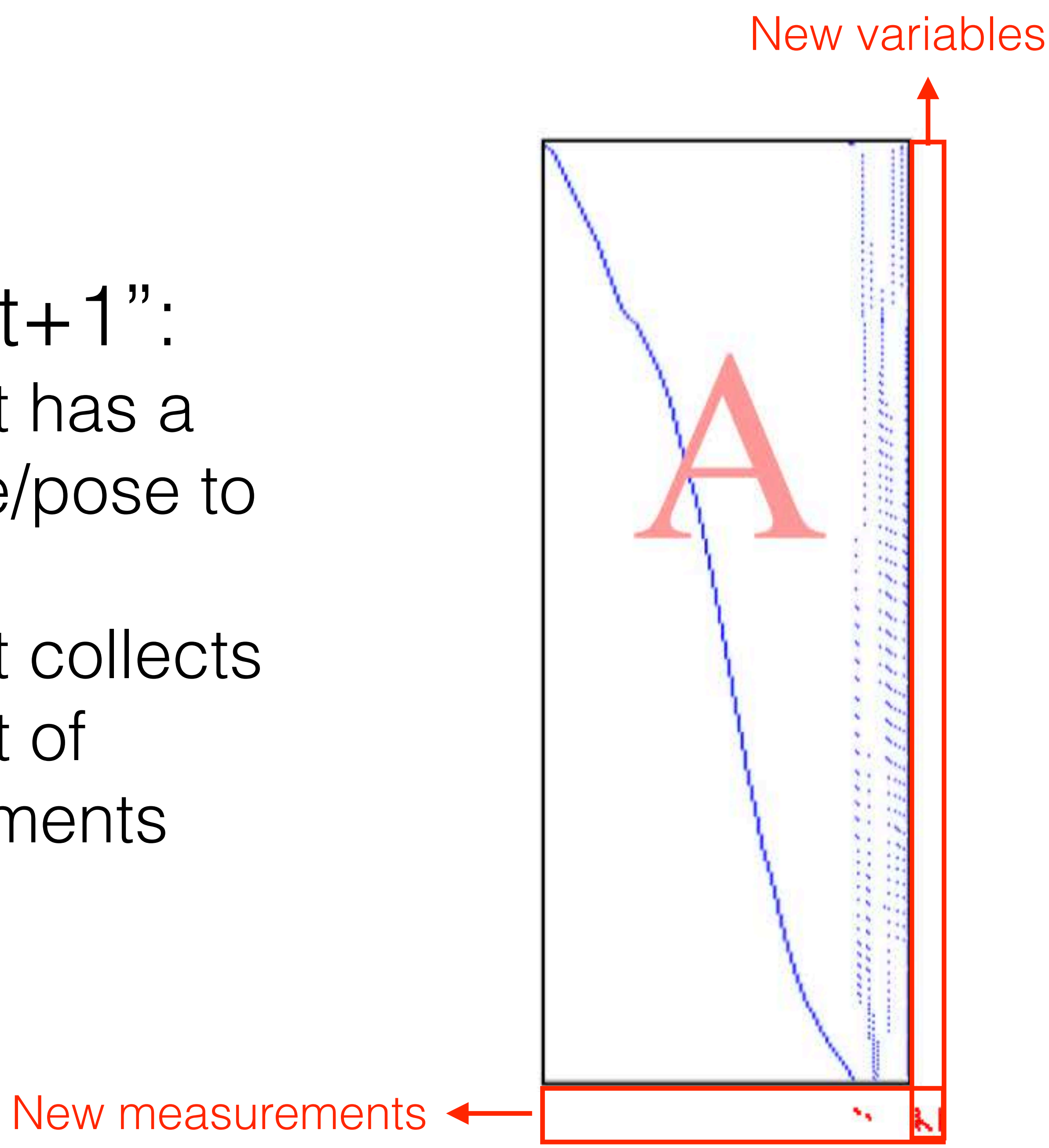
Incremental Smoothing and Mapping (iSAM)

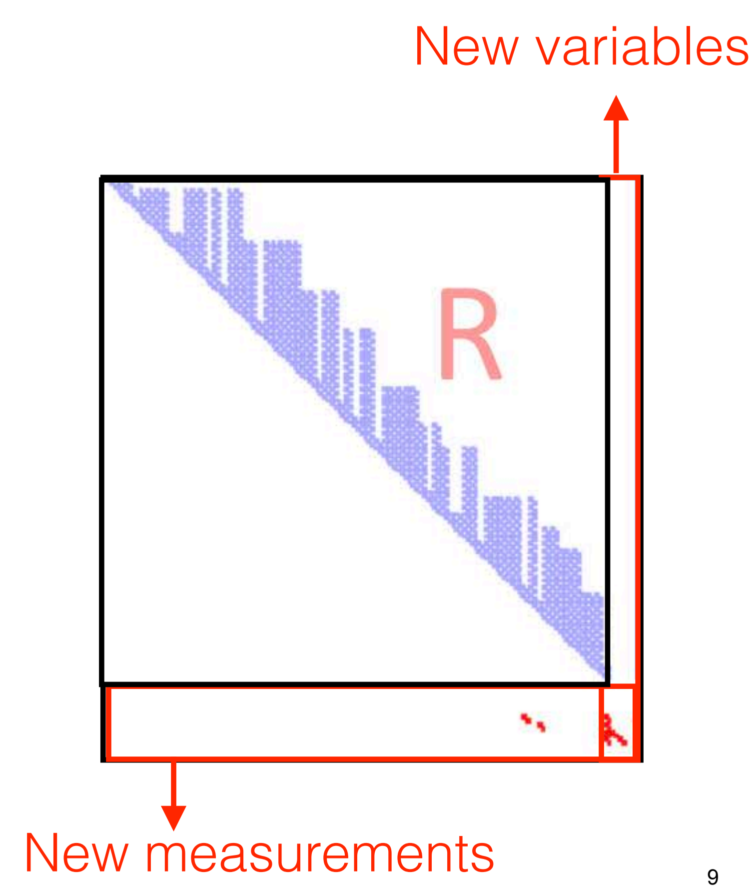
After applying QR at time "t":

$$\|Ax - \mathbf{b}\|^2 = \|Rx - \mathbf{d}\|^2 + \|\mathbf{e}\|^2$$

At time "t+1":

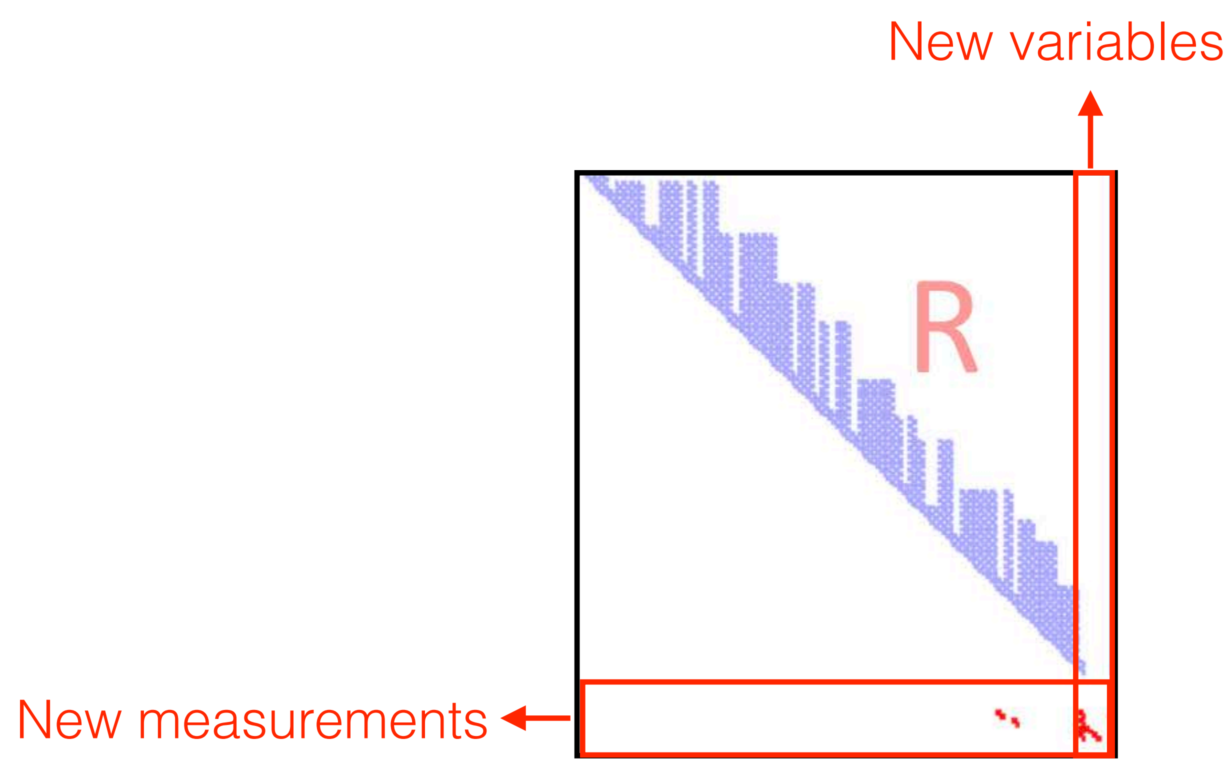
- The robot has a new state/pose to estimate
- The robot collects a new set of measurements





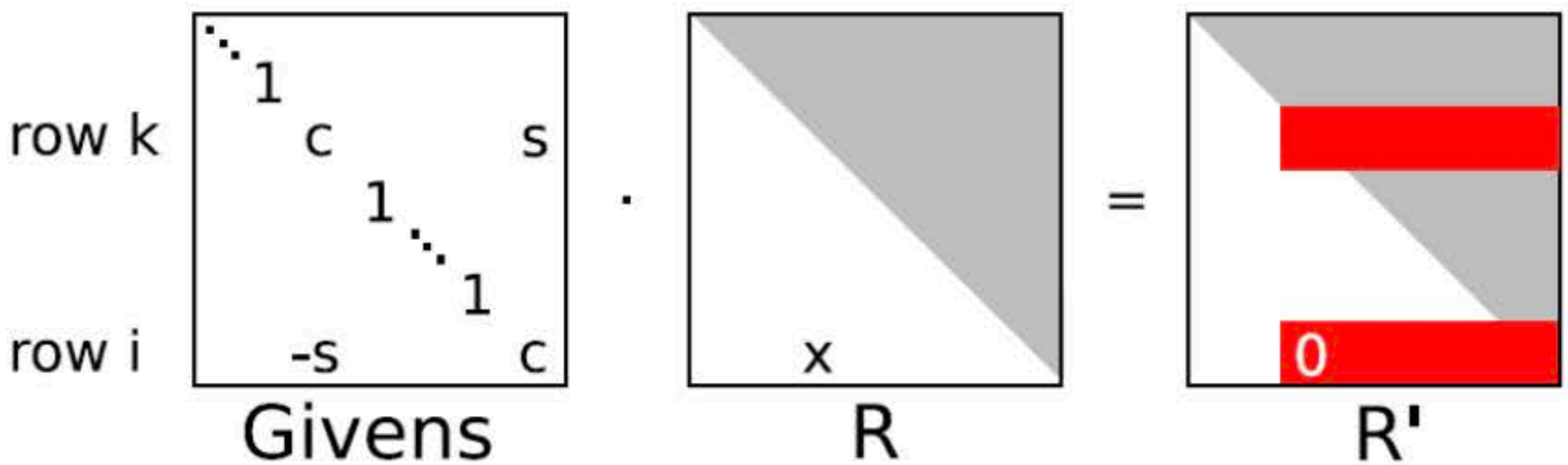
Incremental Smoothing and Mapping (iSAM)

iSAM - key ideas:



- Append new measurements to existing factorization (R)
- "Repair" using *Givens* rotations

$$\|Rx - \mathbf{d}\|^2$$



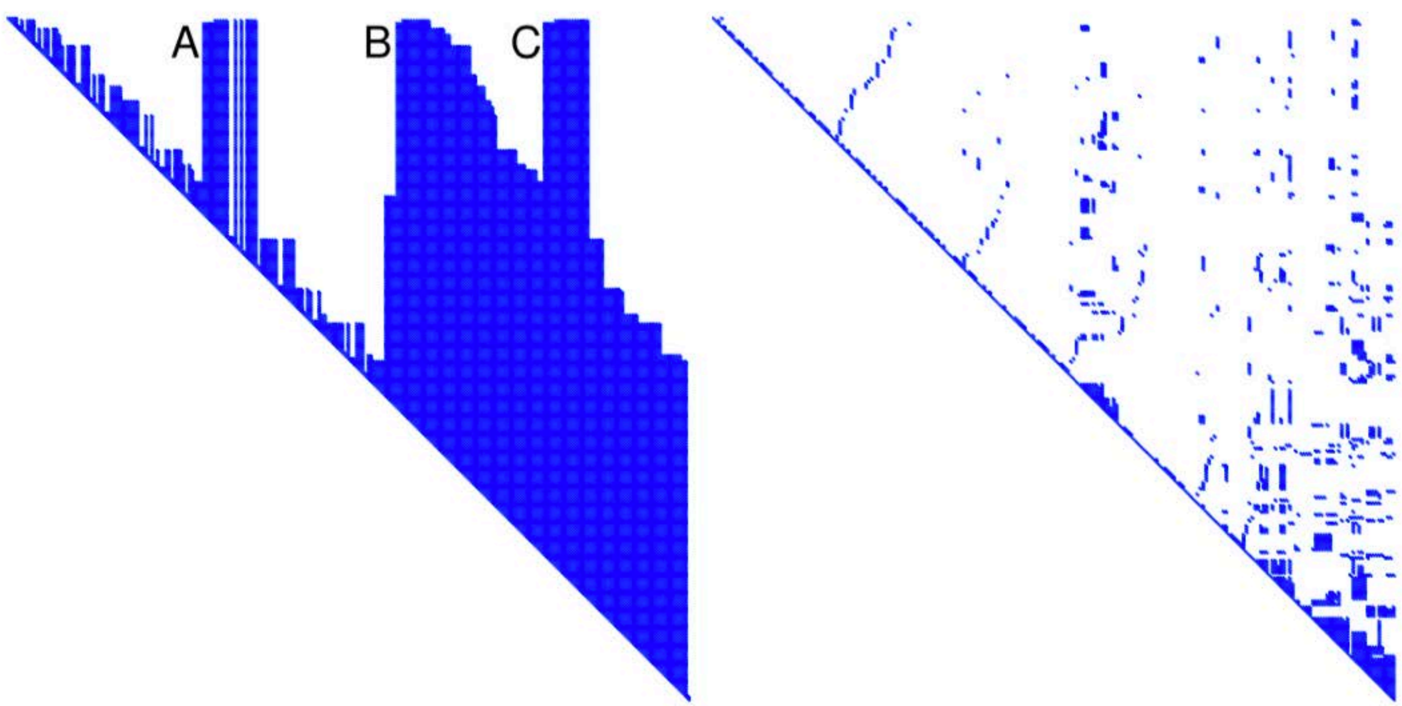
ISAM - issues

- The world is nonlinear

$$\operatorname{argmin}_{\mathrm{x}} \sum_{i} \|h_i(\mathrm{x})\|_{\Xi}{}^2$$

Periodic re-linearization

- Fill-in: ordering influences sparsity of sqrt info matrix R

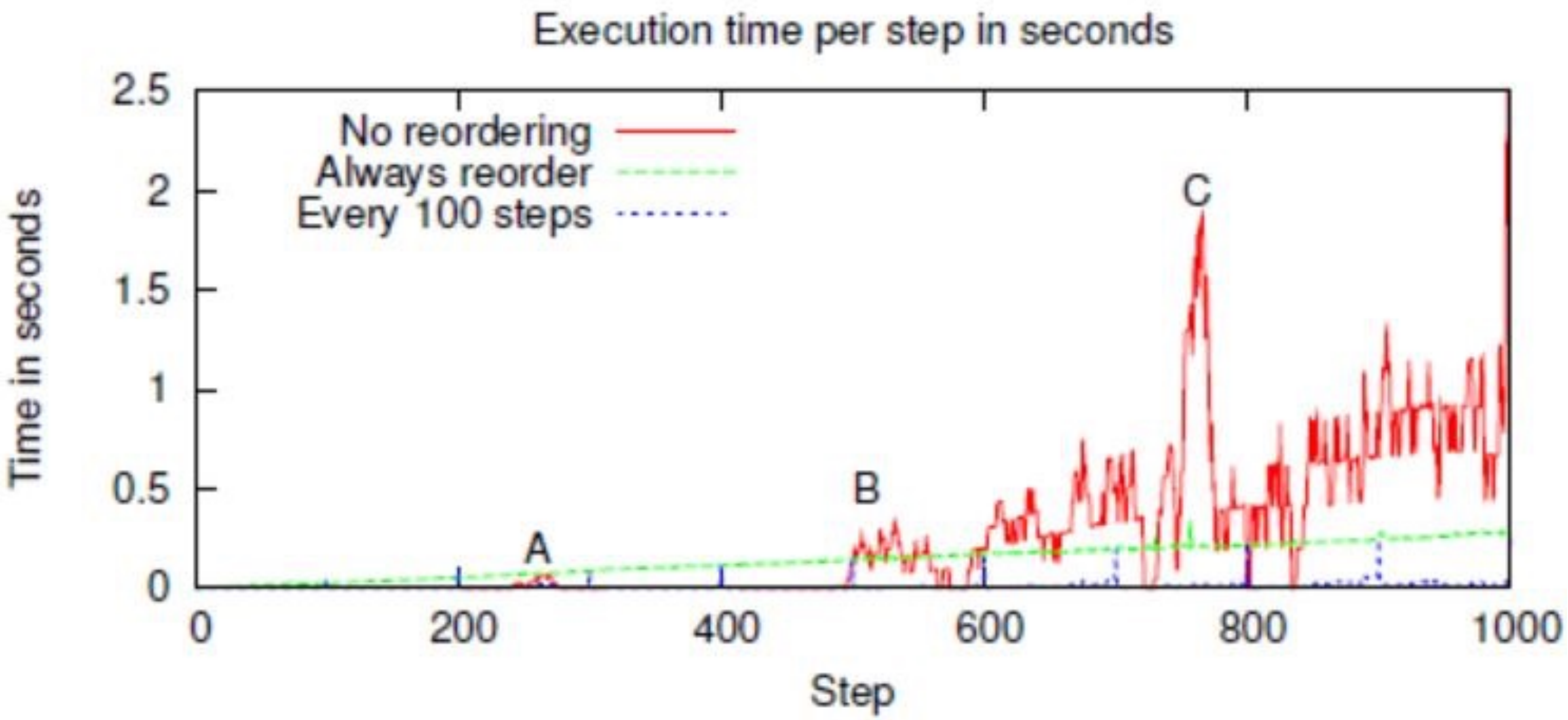


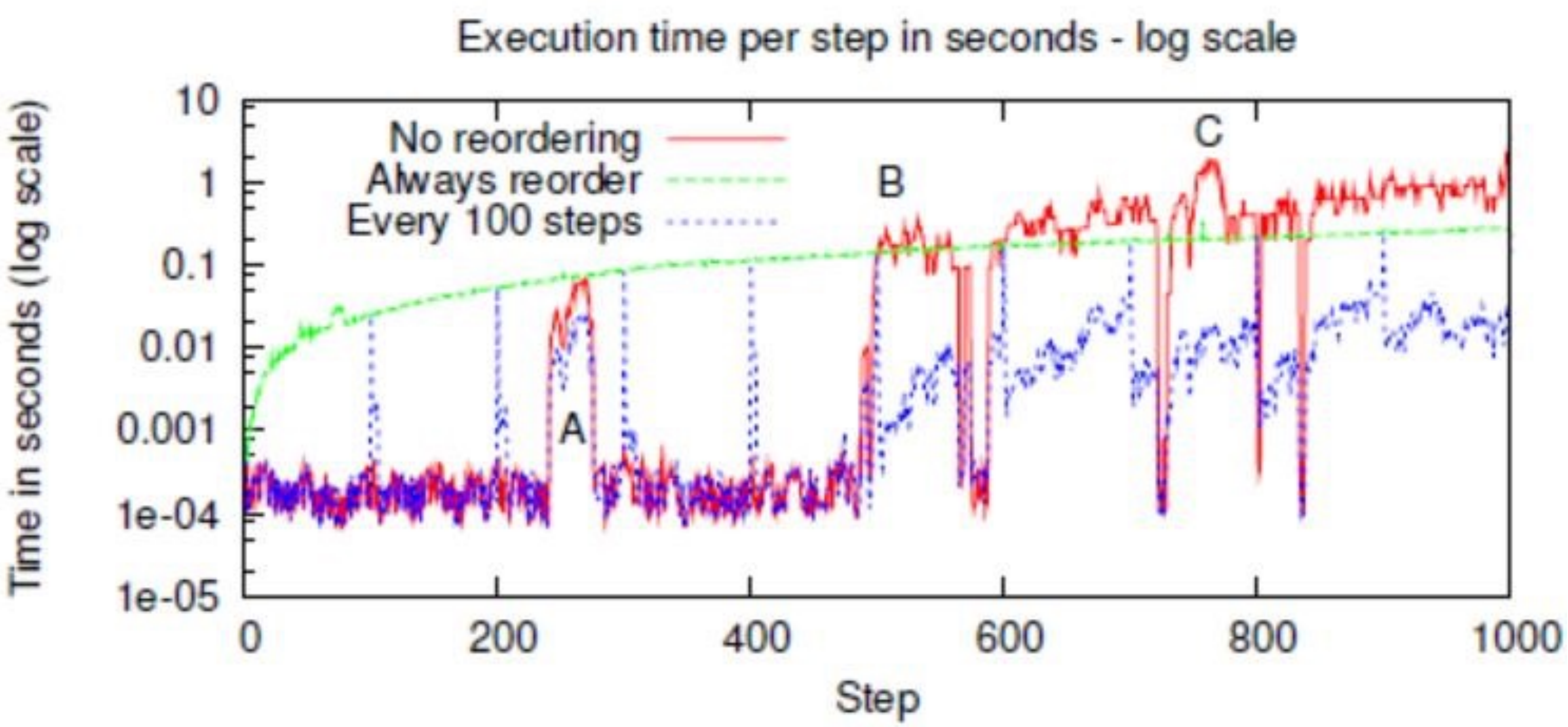
(b) Factor R.

(c) The same factor R after variable reordering.

Periodic re-ordering

iSAM - results





During re-linearization and reordering we still have to recompute everything from scratch

Not O(1),
We need iSAM2!

Today: Incremental Solvers for SLAM

- iSAM (Incremental Smoothing And Mapping)

- Bayes Tree and iSAM2

T-RO 2008

IEEE TRANSACTIONS ON ROBOTICS, MANUSCRIPT SEPTEMBER 7, 2008 1

iSAM: Incremental Smoothing and Mapping

Michael Kaess, *Student Member, IEEE,* Ananth Ranganathan, *Student Member, IEEE,* and Frank Dellaert, *Member, IEEE*

***Abstract*—We present incremental smoothing and mapping (iSAM), a novel approach to the simultaneous localization and mapping problem that is based on fast incremental matrix factorization. iSAM provides an efficient and exact solution by updating a QR factorization of the naturally sparse smoothing information matrix, therefore recalculating only the matrix entries that actually change. iSAM is efficient even for robot trajectories**

counterintuitive at first, because more variables are added to the estimation problem, the simplification arises from the fact that the smoothing information matrix is naturally sparse. In contrast, in filtering approaches the information matrix becomes dense when marginalizing out robot poses. As a consequence of applying smoothing, we are able to

iSAM2: Incremental Smoothing and Mapping Using the Bayes Tree

Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John Leonard, and Frank Dellaert

IJRR 2011

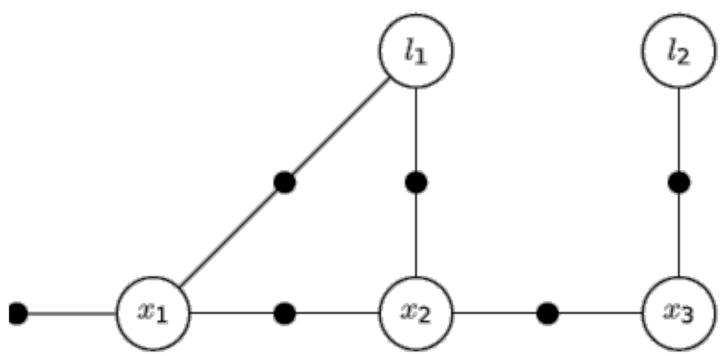
Abstract

We present a novel data structure, the Bayes tree, that provides an algorithmic foundation enabling a better understanding of existing graphical model inference algorithms and their connection to sparse matrix factorization methods. Similar to a clique tree, a Bayes tree encodes a factored probability density, but unlike the clique tree it is directed and maps more naturally to the square root information matrix of the simultaneous localization and mapping (SLAM) problem. In this paper, we highlight three insights provided by our new data structure. First, the Bayes tree provides a better understanding of the matrix factorization in terms of probability densities. Second, we show how the fairly abstract updates to a matrix factorization translate to a simple editing of the Bayes tree and its conditional densities. Third, we apply the Bayes tree to obtain a completely novel algorithm for sparse nonlinear incremental optimization, named iSAM2, which achieves improvements in efficiency through incremental variable re-ordering and fluid relinearization, eliminating the need for periodic batch steps. We analyze various properties of iSAM2 in detail, and show on a range of real and simulated datasets that our algorithm compares favorably with other recent mapping algorithms in both quality and efficiency.

Keywords: graphical models, clique tree, junction tree, probabilistic inference, sparse linear algebra, nonlinear optimization, smoothing and mapping, SLAM

iSAM2 - key idea 1

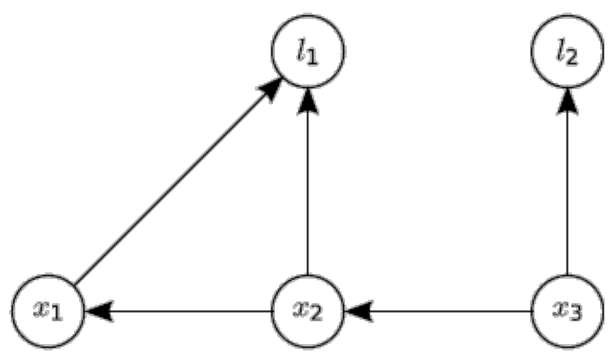
Interpret linear algebra operations (e.g., factorization) as operations on a graphical model



$$A = \begin{array}{c} \begin{array}{ccccc} l_1 & l_2 & x_1 & x_2 & x_3 \end{array} \\ \begin{bmatrix} X & & X & & \\ X & & & X & \\ & X & & & X \\ & & X & & \\ & & X & X & \\ & & & X & X \end{bmatrix} \end{array}$$

(a)

Jacobian
(= Factor Graph)



$$R = \begin{array}{c} \begin{array}{ccccc} l_1 & l_2 & x_1 & x_2 & x_3 \end{array} \\ \begin{bmatrix} X & & X & X & \\ & X & & & X \\ & & X & X & \\ & & & X & X \\ & & & & X \end{bmatrix} \end{array}$$

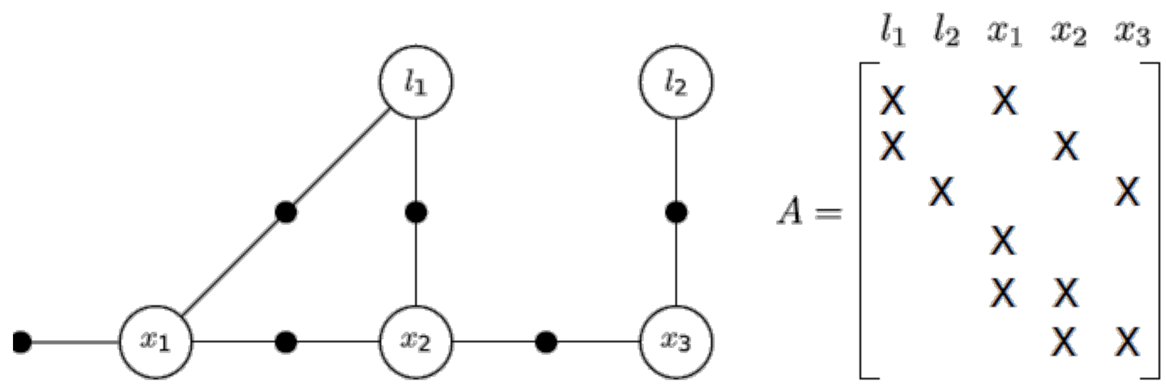
(b)

Square root
information matrix
(= Bayes Net)

- factorization can be “understood as converting the factor graph to a Bayes Net using the elimination algorithm”

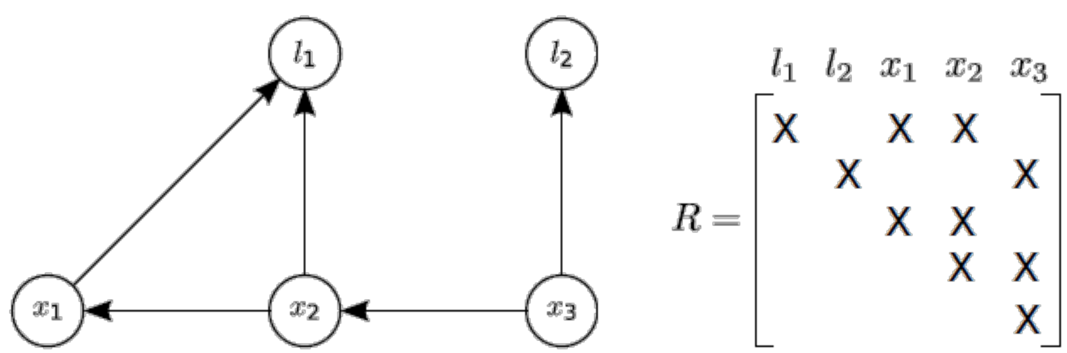
iSAM2 - key idea 2

Introduce a new graphical model (the “Bayes Tree”) and show we can “add” new states and measurements by local operations on the Bayes Tree



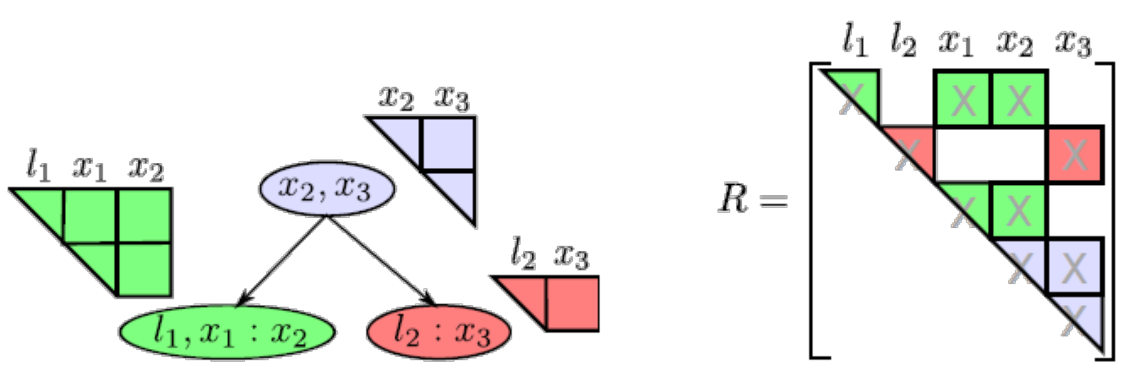
(a)

Jacobian
(= Factor Graph)



(b)

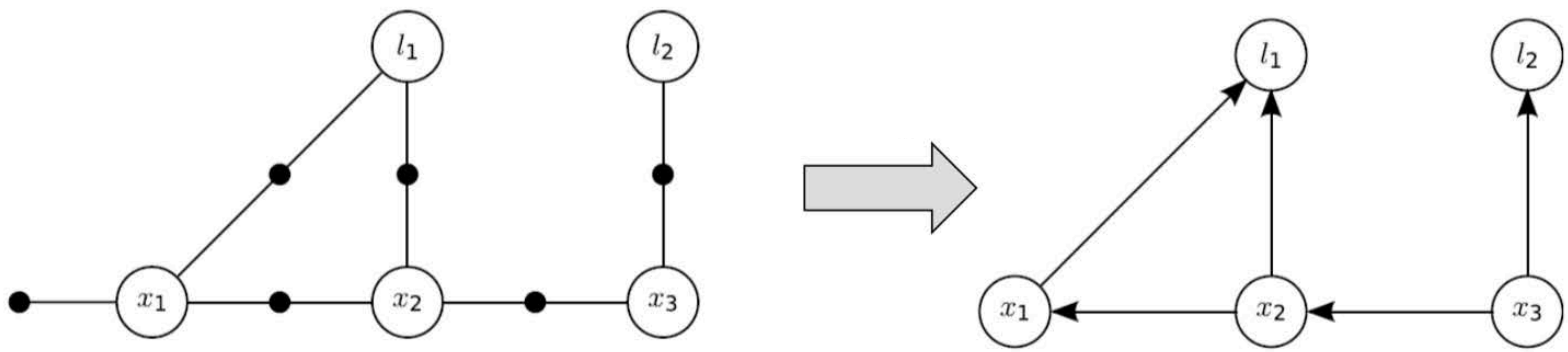
Square root
information matrix
(= Bayes Net)



(c)

Bayes Tree

Variable Elimination and Bayes Net



Bayesian network (a.k.a. Bayes network, belief network): probabilistic graphical model that represents a set of variables and their conditional dependencies via a directed acyclic graph [wiki]

In linear Gaussian case, elimination is equivalent to sparse QR factorization of the measurement Jacobian

$$A = \begin{array}{c} \begin{array}{ccccc} l_1 & l_2 & x_1 & x_2 & x_3 \end{array} \\ \left[\begin{array}{ccccc} X & & X & & \\ X & & & X & \\ & X & & & X \\ & & X & & \\ & & X & X & \\ & & & X & X \end{array}\right] \end{array} \Rightarrow R = \begin{array}{c} \begin{array}{ccccc} l_1 & l_2 & x_1 & x_2 & x_3 \end{array} \\ \left[\begin{array}{ccccc} X & & X & X & \\ & X & & & X \\ & & X & X & \\ & & & X & X \\ & & & & X \end{array}\right] \end{array}$$

Rx = d

Backsubstitution: solve from root of Bayes Net

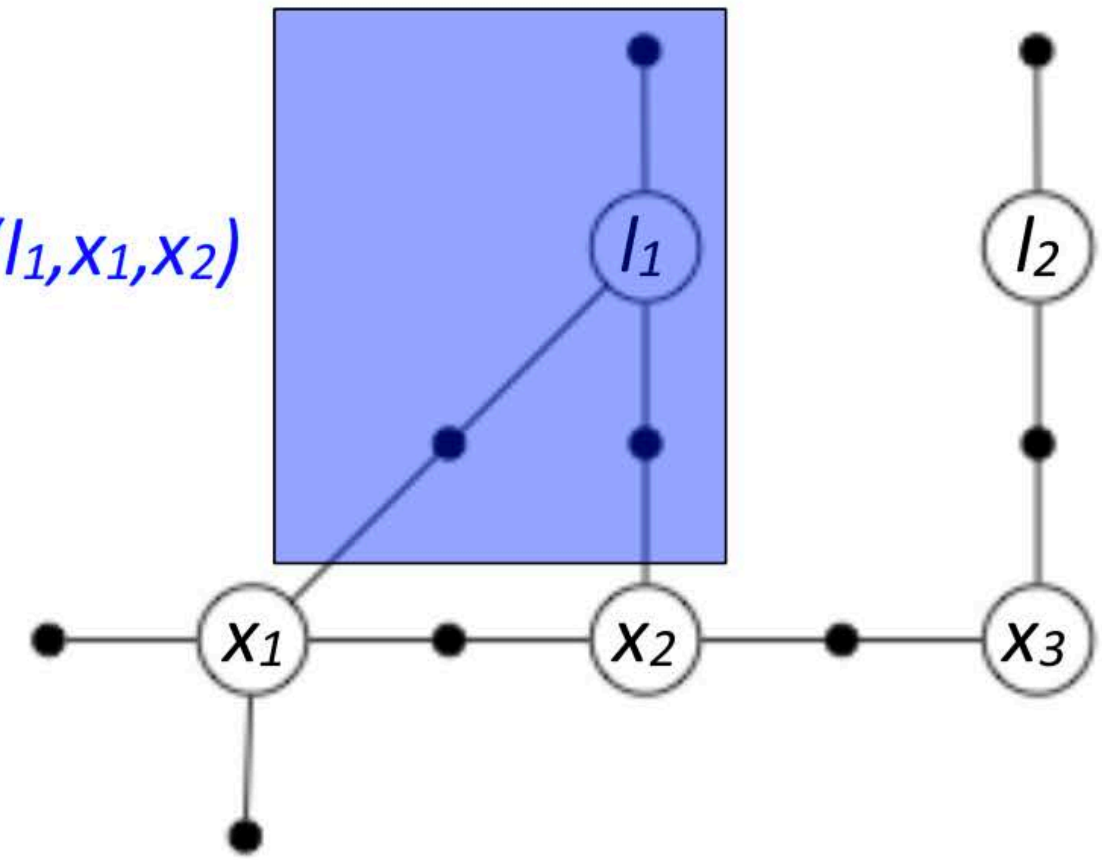
Variable Elimination and Bayes Net

- Choose ordering: l_1, l_2, x_1, x_2, x_3
- Eliminate one node at a time

Alg. 2 Eliminating a variable θ_j from the factor graph.

1. Remove from the factor graph all factors $f_i(\Theta_i)$ that are adjacent to θ_j. Define the *separator* S_j as all variables involved in those factors, excluding θ_j.
2. Form the (unnormalized) joint density $f_{joint}(\theta_j, S_j) = \prod_i f_i(\Theta_i)$ as the product of those factors.
3. Using the chain rule, factorize the joint density $f_{joint}(\theta_j, S_j) = P(\theta_j|S_j) f_{new}(S_j)$. Add the conditional $P(\theta_j|S_j)$ to the Bayes net and the factor $f_{new}(S_j)$ back into the factor graph.

$p(l_1, x_1, x_2)$



$$p(l_1, x_1, x_2) = p(l_1 | x_1, x_2)\, p(x_1, x_2)$$

Variable Elimination and Bayes Net

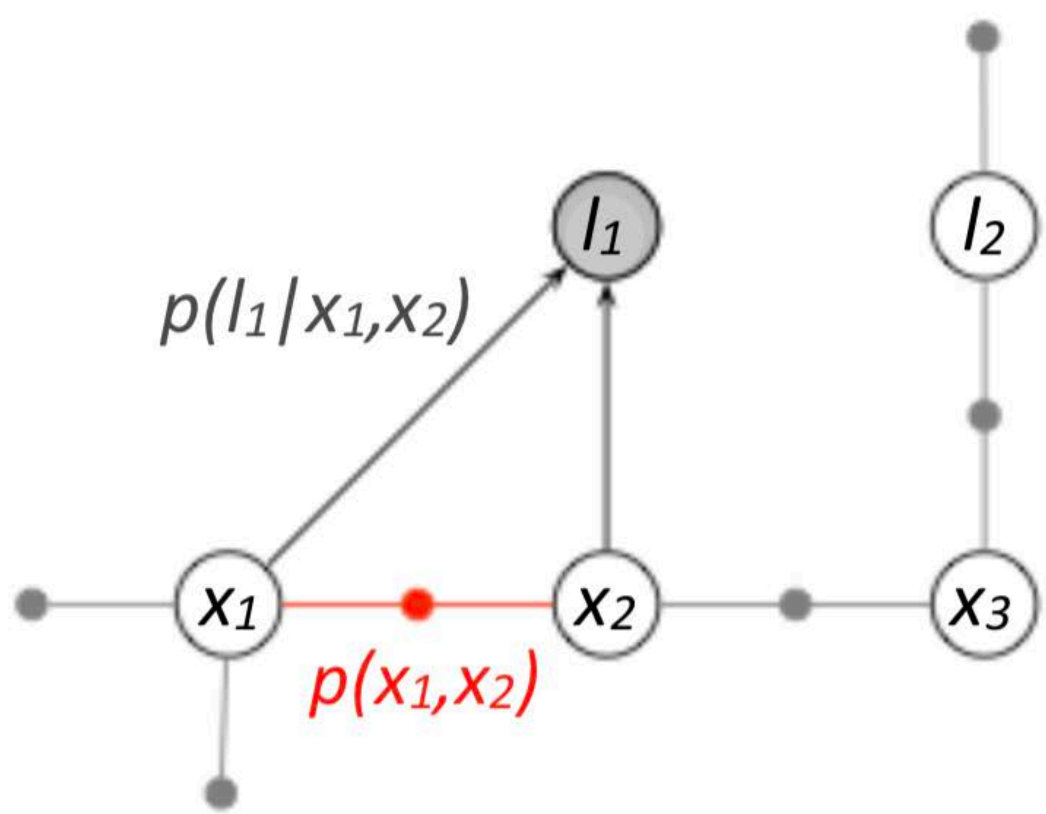
- Choose ordering: l_1, l_2, x_1, x_2, x_3
- Eliminate one node at a time

Alg. 2 Eliminating a variable θ_j from the factor graph.

1. Remove from the factor graph all factors $f_i(\Theta_i)$ that are adjacent to θ_j. Define the *separator* S_j as all variables involved in those factors, excluding θ_j.
2. Form the (unnormalized) joint density $f_{joint}(\theta_j, S_j) = \prod_i f_i(\Theta_i)$ as the product of those factors.
3. Using the chain rule, factorize the joint density $f_{joint}(\theta_j, S_j) = P(\theta_j|S_j) f_{new}(S_j)$. Add the conditional $P(\theta_j|S_j)$ to the Bayes net and the factor $f_{new}(S_j)$ back into the factor graph.

Alg. 2 in M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard and F. Dellaert, "iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering," 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 2011, pp. 3281-3288, doi: 10.1109/ICRA.2011.5979641

$$
\begin{array}{c c}
 & \begin{array}{ccccc} l_1 & l_2 & x_1 & x_2 & x_3 \end{array} \\
A = & \left[\begin{array}{ccccc}
 & & & X & \\
 & X & & & X \\
 & & X & & \\
 & & X & X & \\
 & & & X & X
\end{array}\right]
\end{array}
$$



Can be implemented via Schur complement in linear gaussian graphs

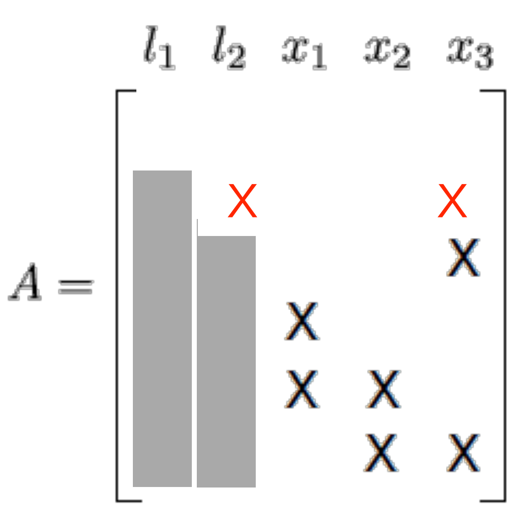
$$p(l_1,x_1,x_2) = p(l_1|x_1,x_2)\, p(x_1,x_2)$$

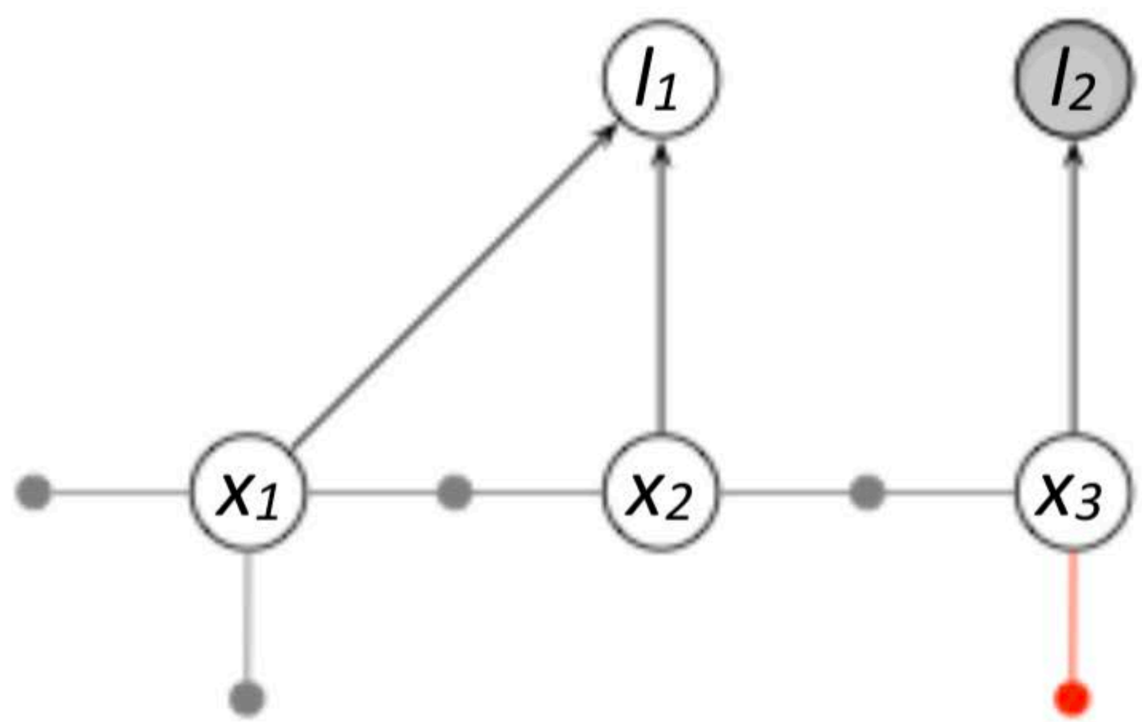
Variable Elimination and Bayes Net

- Choose ordering: l_1, l_2, x_1, x_2, x_3
- Eliminate one node at a time

Alg. 2 Eliminating a variable θ_j from the factor graph.

1. Remove from the factor graph all factors $f_i(\Theta_i)$ that are adjacent to θ_j. Define the *separator* S_j as all variables involved in those factors, excluding θ_j.
2. Form the (unnormalized) joint density $f_{joint}(\theta_j, S_j) = \prod_i f_i(\Theta_i)$ as the product of those factors.
3. Using the chain rule, factorize the joint density $f_{joint}(\theta_j, S_j) = P(\theta_j|S_j) f_{new}(S_j)$. Add the conditional $P(\theta_j|S_j)$ to the Bayes net and the factor $f_{new}(S_j)$ back into the factor graph.





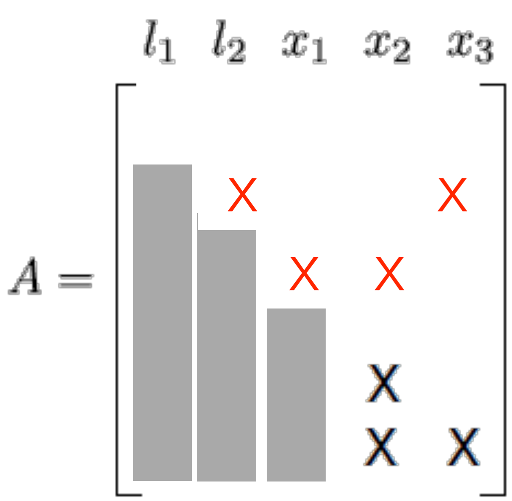
$p(l_2, x_3) = p(l_2|x_3)\, p(x_3)$

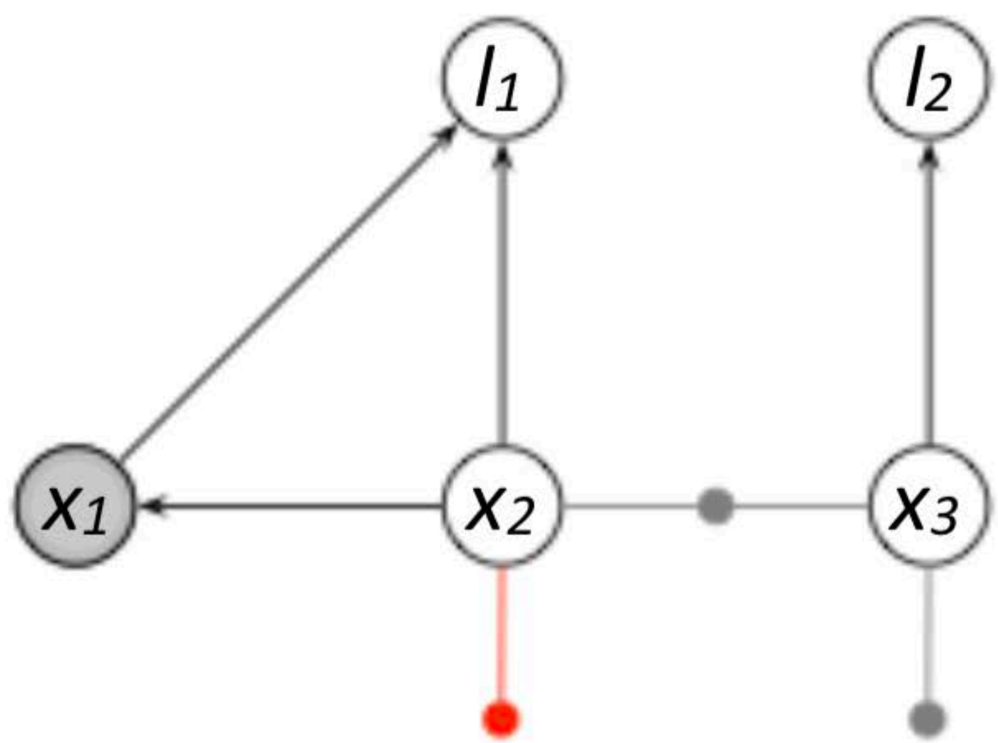
Variable Elimination and Bayes Net

- Choose ordering: l_1, l_2, x_1, x_2, x_3
- Eliminate one node at a time

Alg. 2 Eliminating a variable θ_j from the factor graph.

1. Remove from the factor graph all factors $f_i(\Theta_i)$ that are adjacent to θ_j. Define the *separator* S_j as all variables involved in those factors, excluding θ_j.
2. Form the (unnormalized) joint density $f_{joint}(\theta_j, S_j) = \prod_i f_i(\Theta_i)$ as the product of those factors.
3. Using the chain rule, factorize the joint density $f_{joint}(\theta_j, S_j) = P(\theta_j|S_j) f_{new}(S_j)$. Add the conditional $P(\theta_j|S_j)$ to the Bayes net and the factor $f_{new}(S_j)$ back into the factor graph.





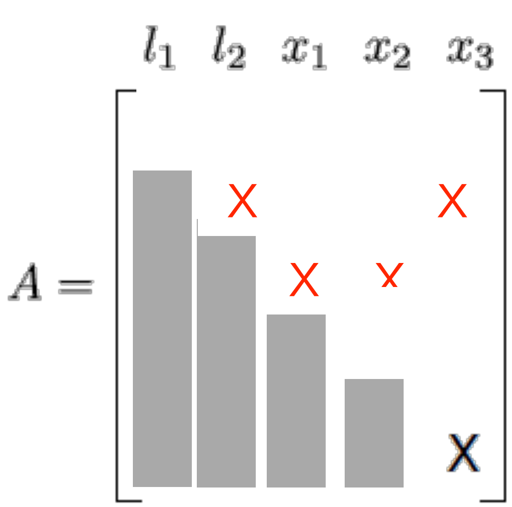
$$p(x_1, x_2) = p(x_1|x_2)\, p(x_2)$$

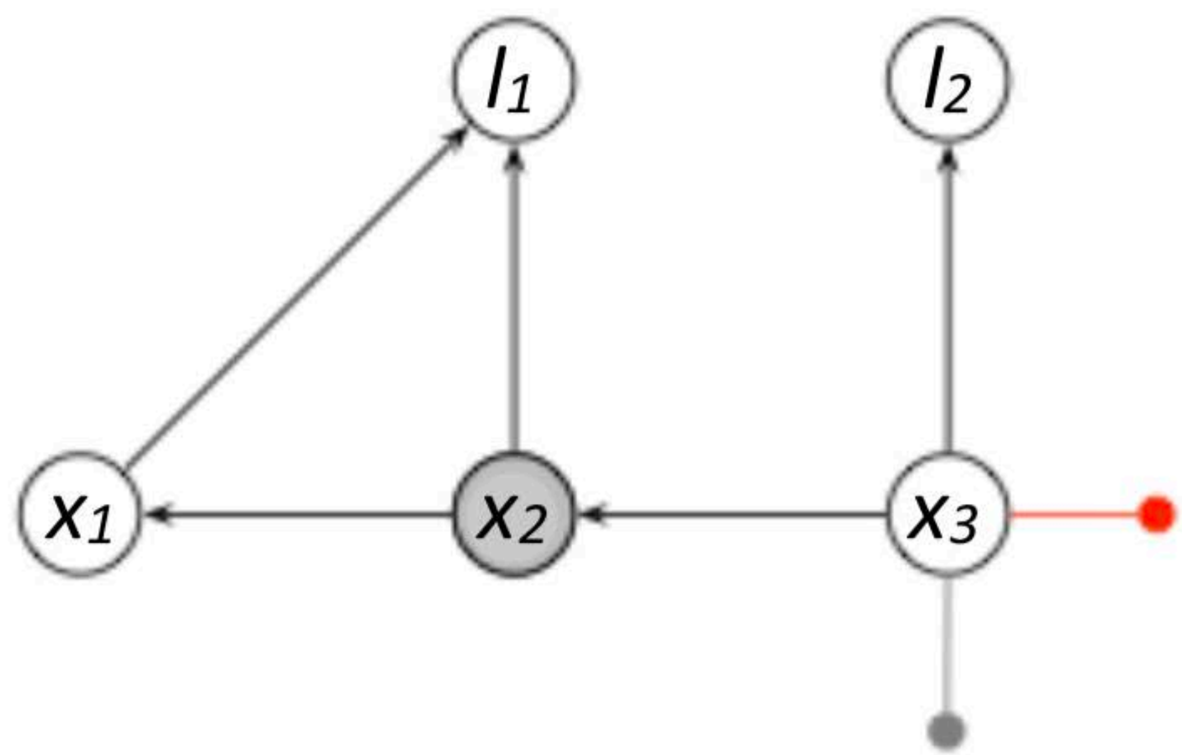
Variable Elimination and Bayes Net

- Choose ordering: l_1, l_2, x_1, x_2, x_3
- Eliminate one node at a time

Alg. 2 Eliminating a variable θ_j from the factor graph.

1. Remove from the factor graph all factors $f_i(\Theta_i)$ that are adjacent to θ_j. Define the *separator* S_j as all variables involved in those factors, excluding θ_j.
2. Form the (unnormalized) joint density $f_{joint}(\theta_j, S_j) = \prod_i f_i(\Theta_i)$ as the product of those factors.
3. Using the chain rule, factorize the joint density $f_{joint}(\theta_j, S_j) = P(\theta_j|S_j) f_{new}(S_j)$. Add the conditional $P(\theta_j|S_j)$ to the Bayes net and the factor $f_{new}(S_j)$ back into the factor graph.





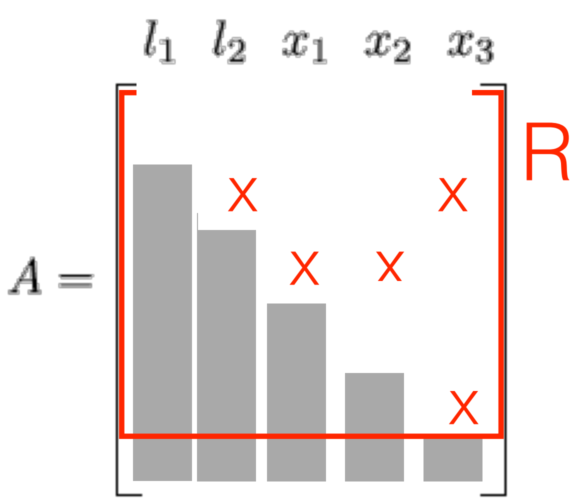
$p(x_2, x_3) = p(x_2|x_3)\, p(x_3)$

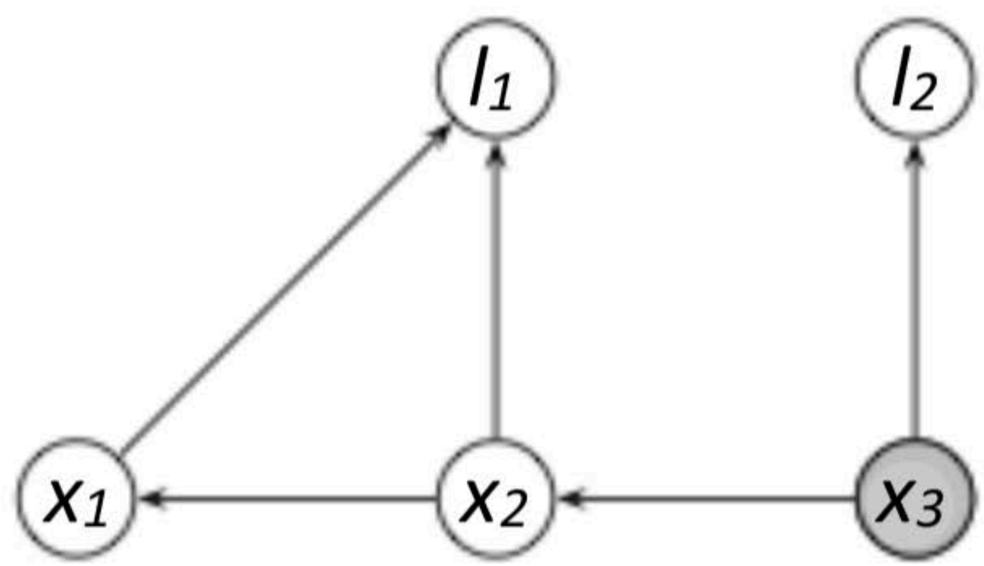
Variable Elimination and Bayes Net

- Choose ordering: l_1, l_2, x_1, x_2, x_3
- Eliminate one node at a time

Alg. 2 Eliminating a variable θ_j from the factor graph.

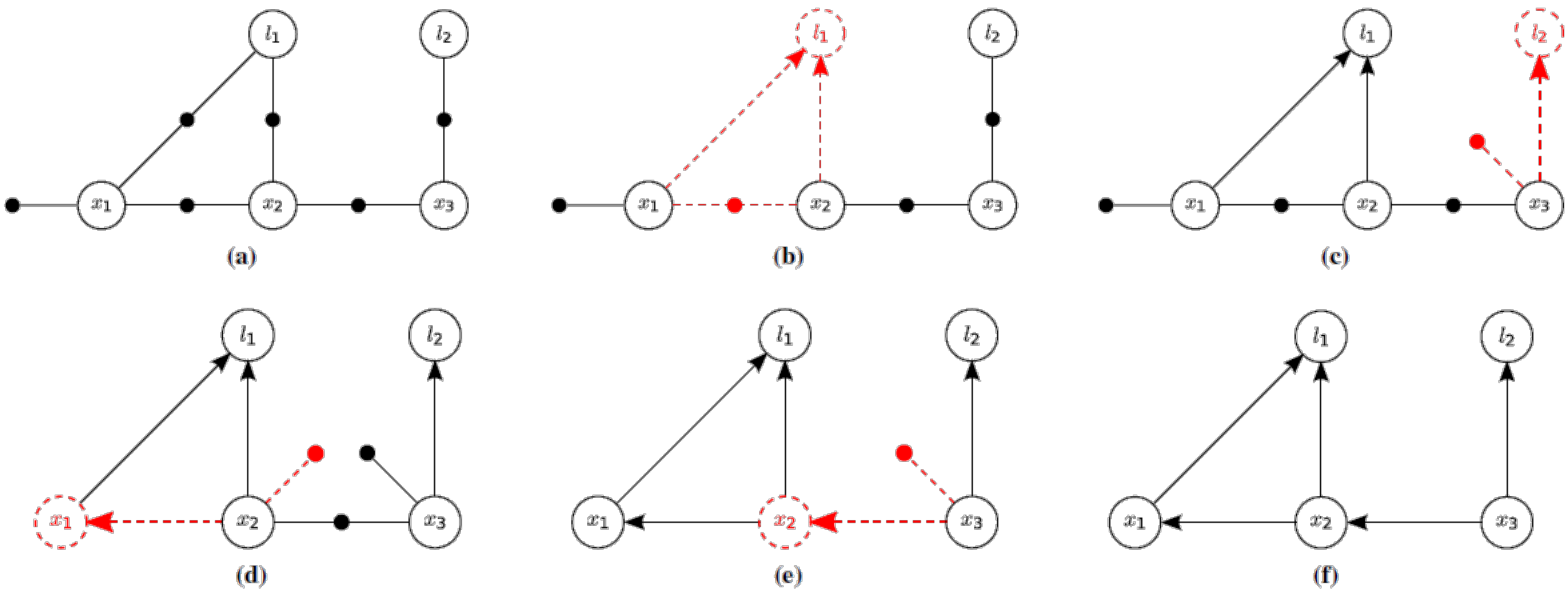
1. Remove from the factor graph all factors $f_i(\Theta_i)$ that are adjacent to θ_j. Define the *separator* S_j as all variables involved in those factors, excluding θ_j.
2. Form the (unnormalized) joint density $f_{joint}(\theta_j, S_j) = \prod_i f_i(\Theta_i)$ as the product of those factors.
3. Using the chain rule, factorize the joint density $f_{joint}(\theta_j, S_j) = P(\theta_j|S_j) f_{new}(S_j)$. Add the conditional $P(\theta_j|S_j)$ to the Bayes net and the factor $f_{new}(S_j)$ back into the factor graph.





$p(x_3)$

Variable Elimination and Bayes Net - Summary



Linear algebra perspective

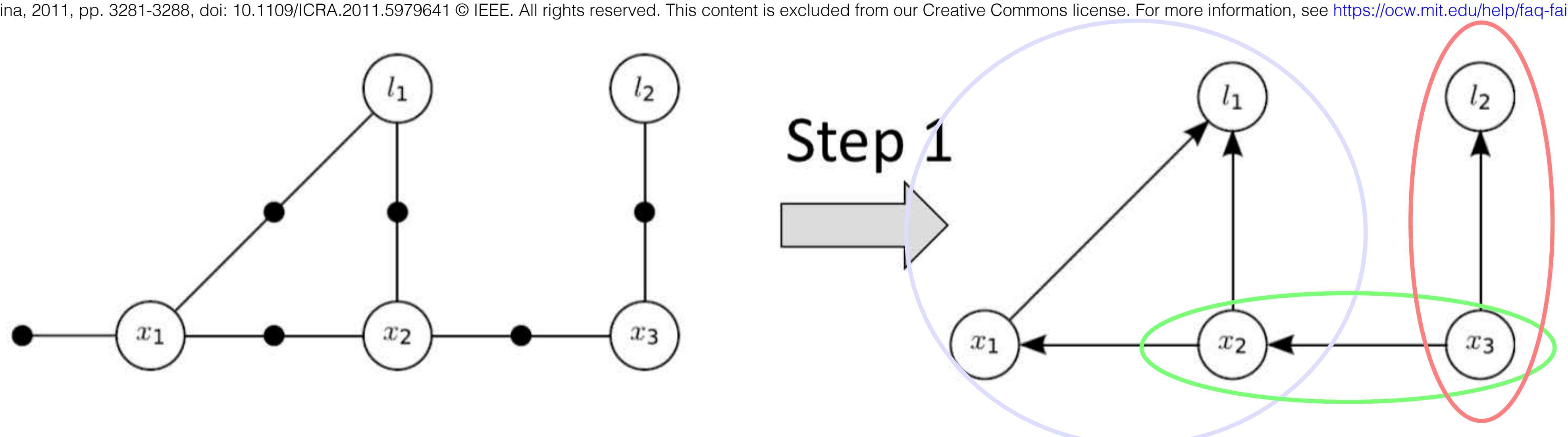
$$A \rightarrow R$$

Graphical model perspective

$$P(\Theta) = \prod_j P(\theta_j | S_j)$$

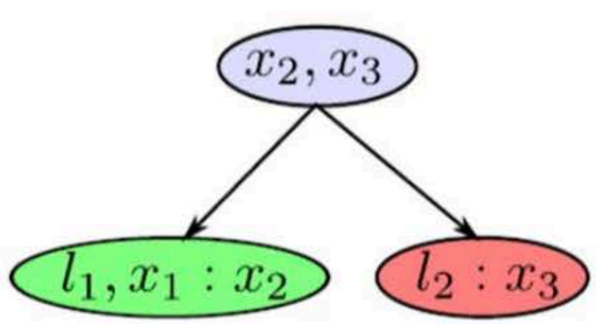
Bayes Tree

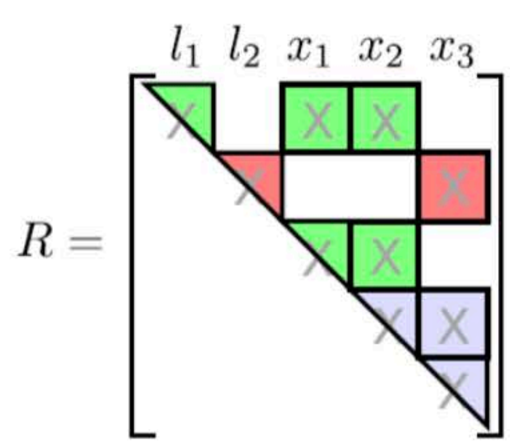
- Another directed graph: collects the cliques (fully connected subgraphs) of the Bayes Net
- Similar to a junction tree, but directed



l_1, l_2, x_1, x_2, x_3

Step 2: Find cliques in reverse elimination order:





Frontal variables

$$P(\Theta) = \prod_k P(F_k | S_k)$$

Separators

iSAM2: Incremental Solver based on Bayes Tree

- New variables and measurements only affect part of the Bayer Tree
- No need for global re-linearization and re-ordering
- “When a new measurement is added, for example a factor *f*(*x1*,*x2*), only the paths between the cliques containing x1 and x2 (respectively) and the root are affected.” [Kaess et al.]

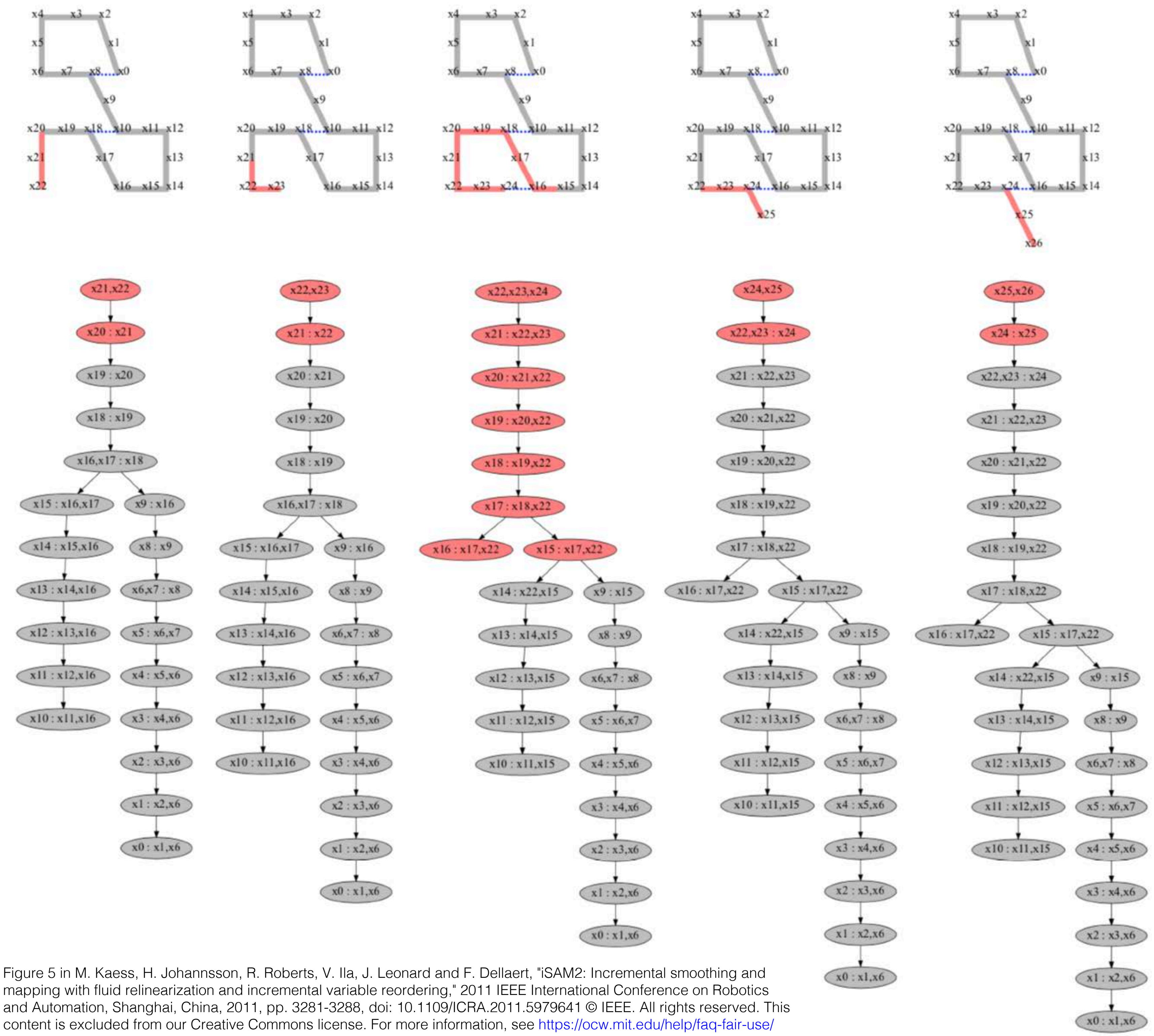
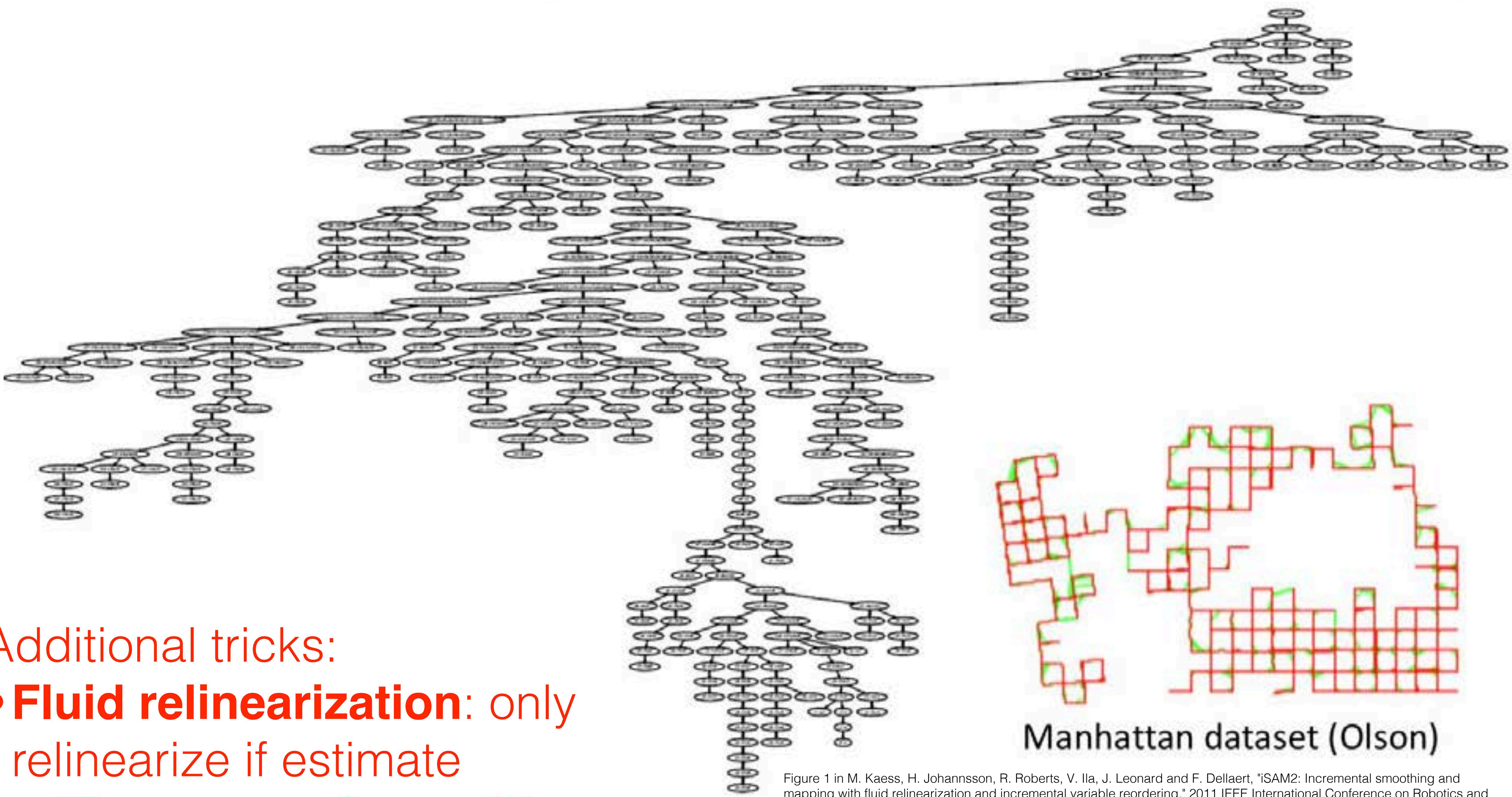


Figure 5 in M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard and F. Dellaert, "iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering," 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 2011, pp. 3281-3288, doi: 10.1109/ICRA.2011.5979641

Re-linearization, backsubstitution, reordering, etc. can be performed locally!

iSAM2: Incremental Solver based on Bayes Tree



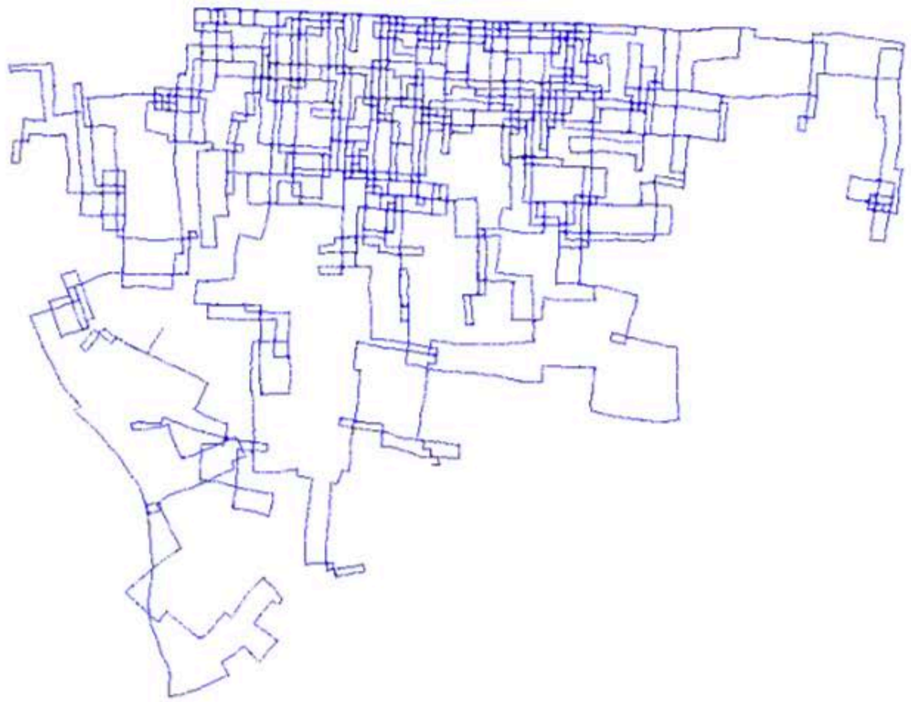
Manhattan dataset (Olson)

Additional tricks:

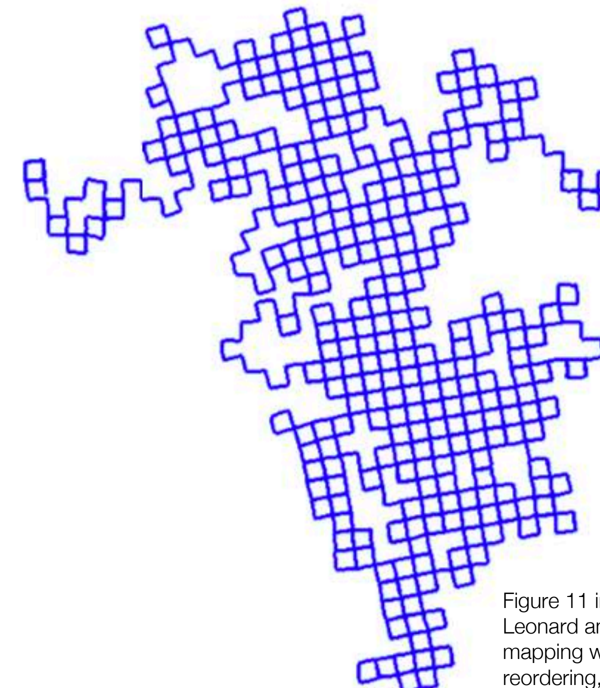
- **Fluid relinearization**: only relinearize if estimate changes beyond a threshold
- **Partial state updates**: early stopping of backsubstitution

O(1) complexity
when no loop closure

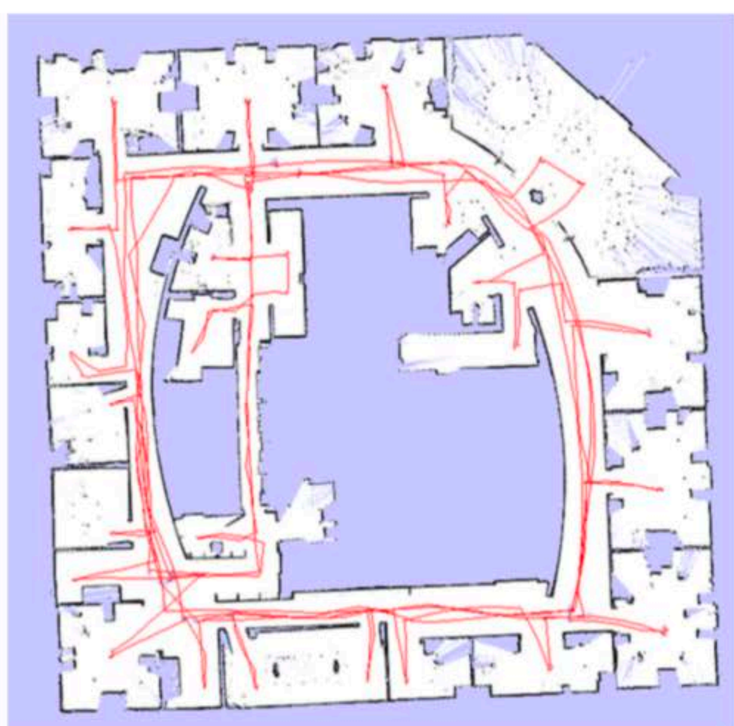
iSAM2 - Results



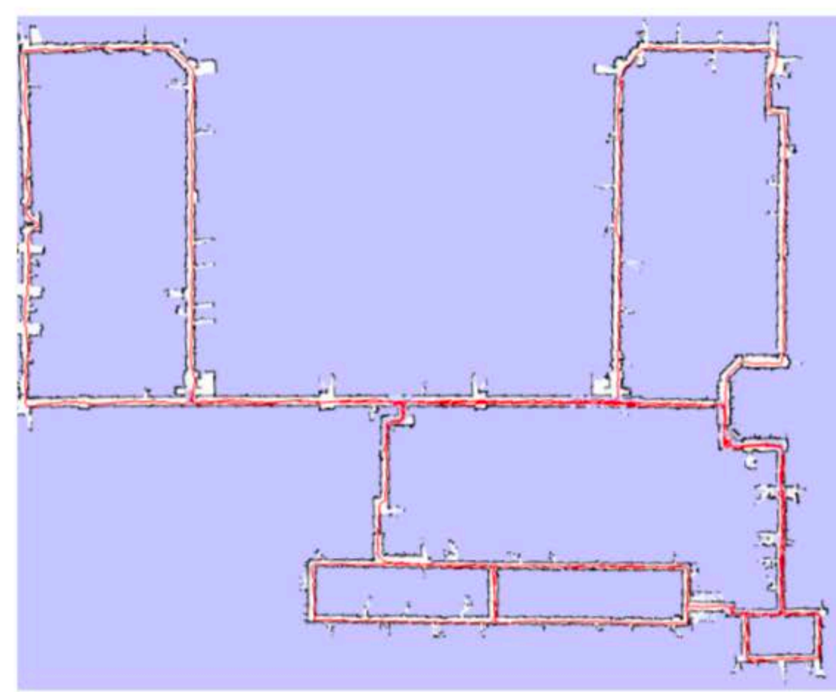
(a) City20000



(b) W10000



(c) Intel



(d) Killian Court

Fig. 11: 2D pose-graph datasets, including simulated data (City20000, W10000), and laser range data (Killian Court, Intel). See Fig. 8 for the Manhattan sequence.

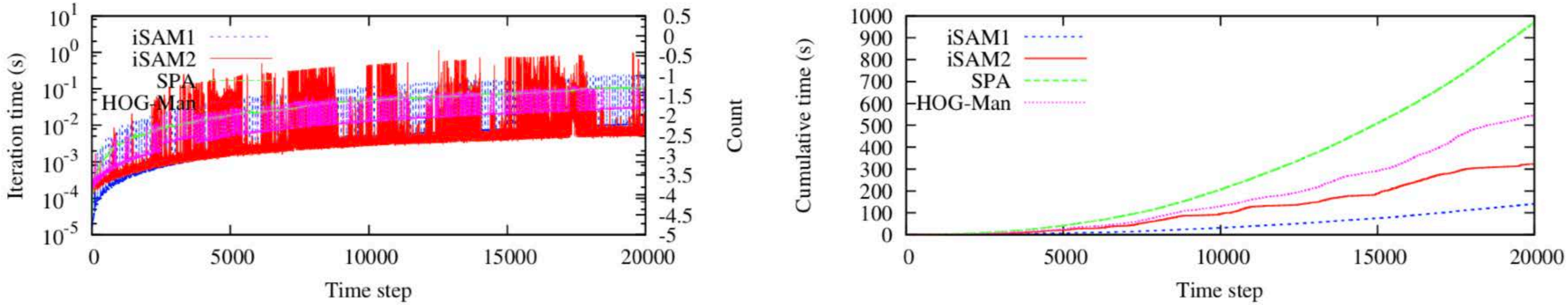
iSAM2:
Incremental Smoothing and Mapping Using the Bayes Tree

Michael Kaess, Hordur Johannsson, Richard Roberts,
Viorela Ila, John Leonard, Frank Dellaert

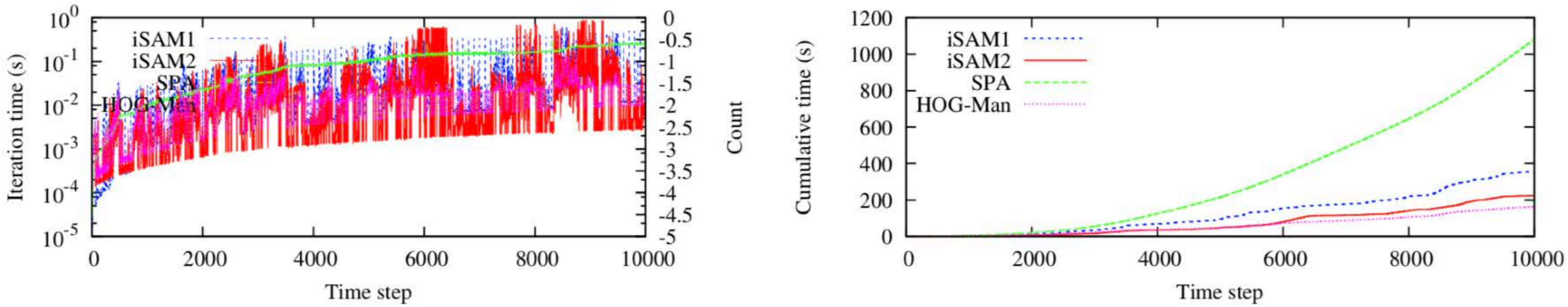
Seq: Manhattan dataset

IJRR Multimedia Extension

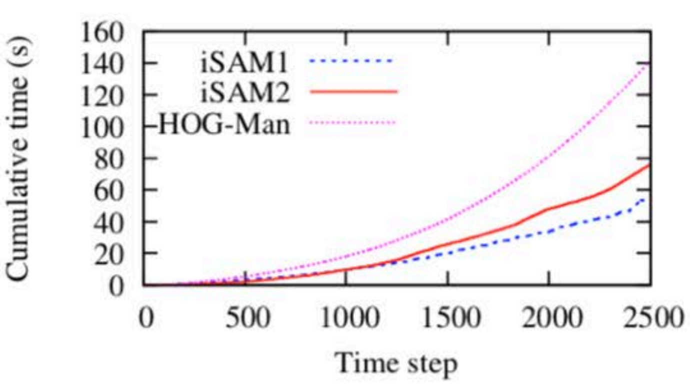
iSAM2 - Results



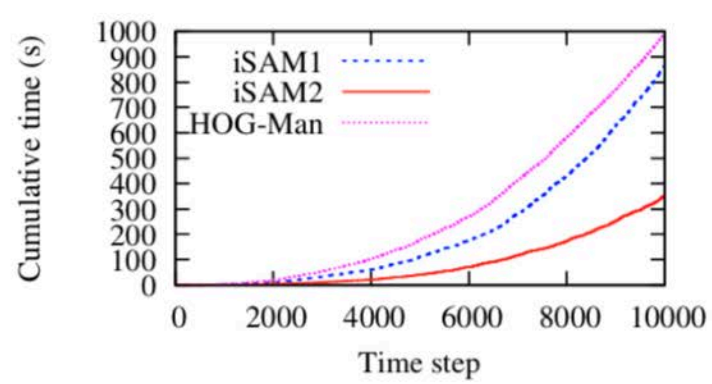
(a) City20000



(b) W10000



(g) Sphere2500



(h) Torus10000

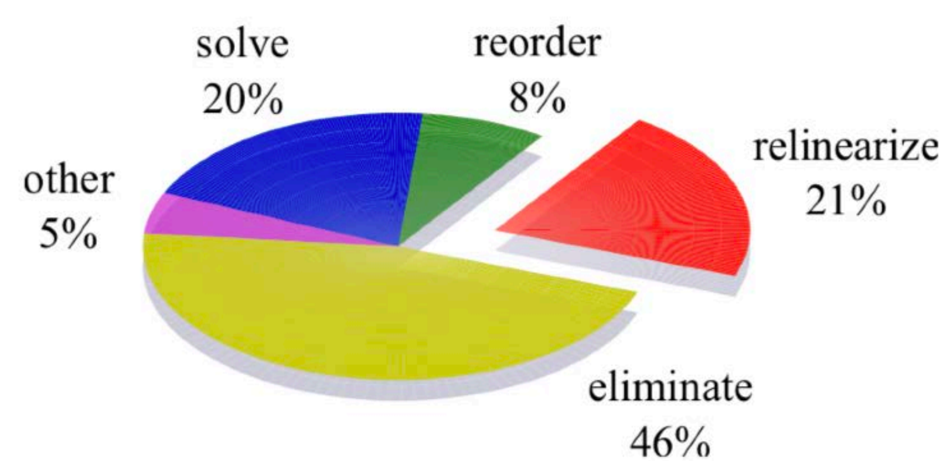
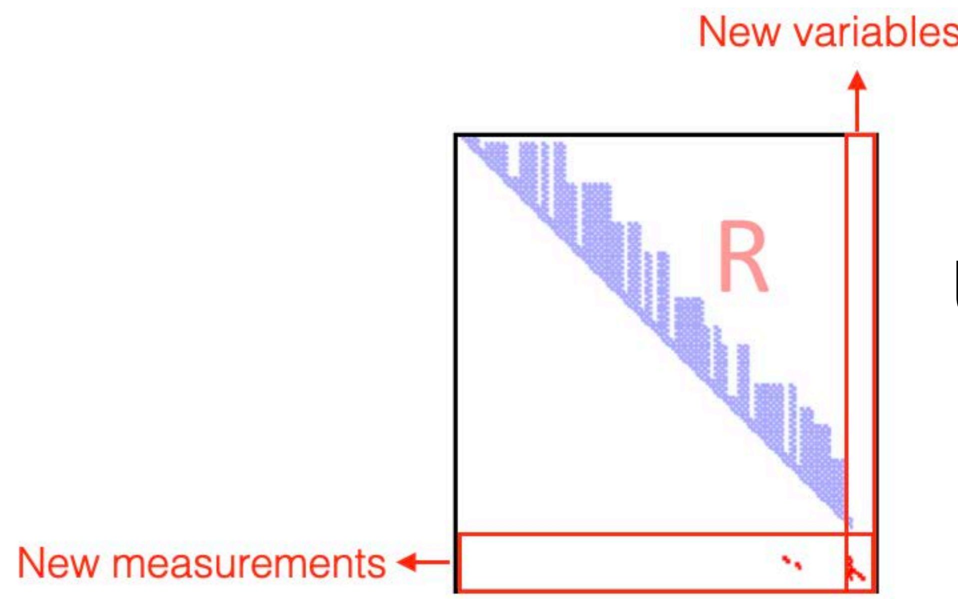


Fig. 15: How time is spent in iSAM2: Percentage of time spent in various components of the algorithm for the W10000 dataset.

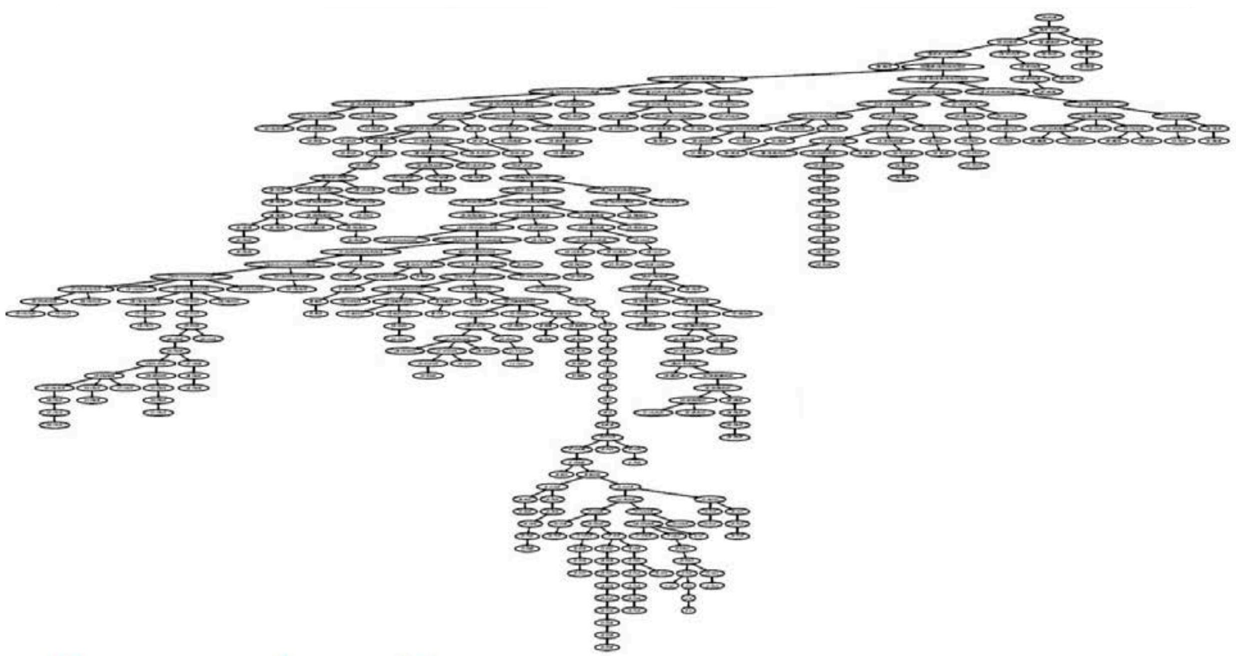
Today: Incremental Solvers for SLAM

- iSAM (Incremental Smoothing And Mapping)



Updates based on Givens rotations

- Bayes Tree and iSAM2



Updates based on local operations on Bayes Tree

iSAM2 - Fluid Relinearization

Alg. 5 Fluid relinearization: The linearization points of select variables are updated based on the current delta Δ.

In: linearization point Θ, delta Δ
Out: updated linearization point Θ, marked cliques M

1. Mark variables in Δ above threshold β: $J = \{\Delta_j \in \Delta \,|\, |\Delta_j| \geq \beta\}$.
2. Update linearization point for marked variables: $\Theta_J := \Theta_J \oplus \Delta_J$.
3. Mark all cliques M that involve marked variables Θ_J and all their ancestors.

Alg. 6 Updating the Bayes tree inclusive of fluid relinearization by recalculating all affected cliques. Note that the algorithm differs from Alg. 4 as it also includes the fluid relinearization; combining both steps is more efficient.

In: Bayes tree $\mathscr{T}$, nonlinear factors $\mathscr{F}$, affected variables $\mathscr{J}$
Out: modified Bayes tree $\mathscr{T}$'

1. Remove top of Bayes tree:
   (a) For each affected variable in $\mathscr{J}$ remove the corresponding clique and all parents up to the root.
   (b) Store orphaned sub-trees $\mathscr{T}_{orph}$ of removed cliques.
2. Relinearize all factors required to recreate top.
3. Add cached linear factors from orphans $\mathscr{T}_{orph}$.
4. Re-order variables, see Section 3.4.
5. Eliminate the factor graph (Alg. 2) and create a new Bayes tree (Alg. 3).
6. Insert the orphans $\mathscr{T}_{orph}$ back into the new Bayes tree.

iSAM2 - Partial State Updates

Alg. 7 Partial state update: Solving the Bayes tree in the nonlinear case returns an update Δ to the current linearization point Θ.

In: Bayes tree $\mathscr{T}$

Out: update Δ

Starting from the root clique $C_r = F_r$:

1. For current clique $C_k = F_k : S_k$
   compute update Δ_k of frontal variables F_k from the local conditional density $P(F_k|S_k)$.

2. For all variables Δ_{k_j} in Δ_k that change by more than threshold α: recursively process each descendant containing such a variable.

16.485 Visual Navigation for Autonomous Vehicles (VNAV)
Fall 2020