Carnegie Mellon
THE ROBOTICS INSTITUTE

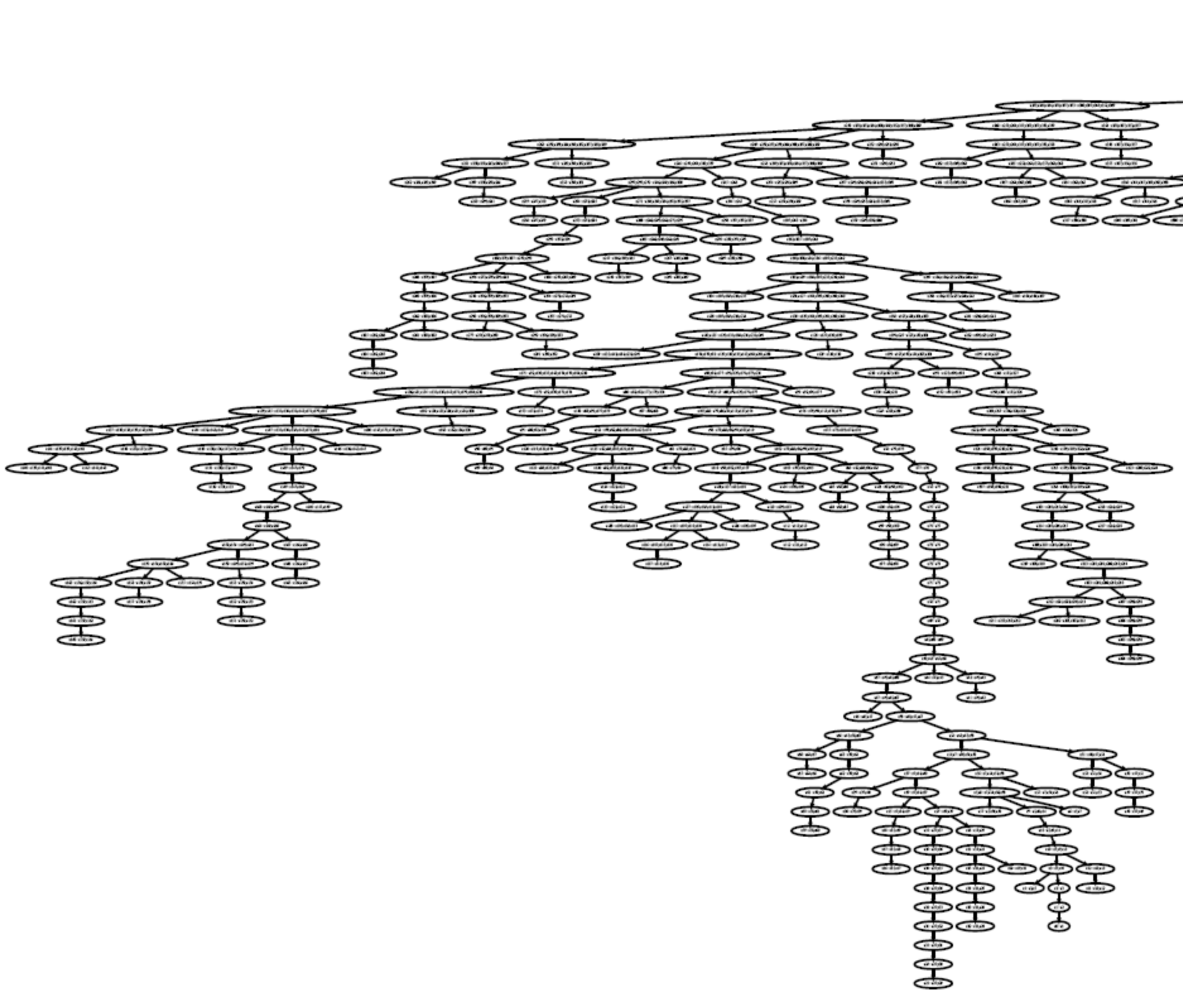
Incrementally Updating the Bayes Tree

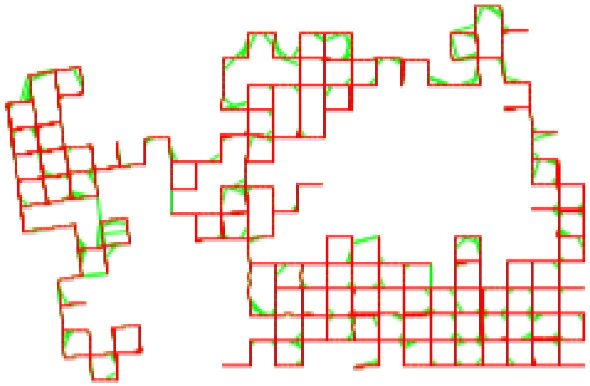
Robot Localization and Mapping
16-833

Michael Kaess

November 13+18, 2024

iSAM2: Bayes Tree Example

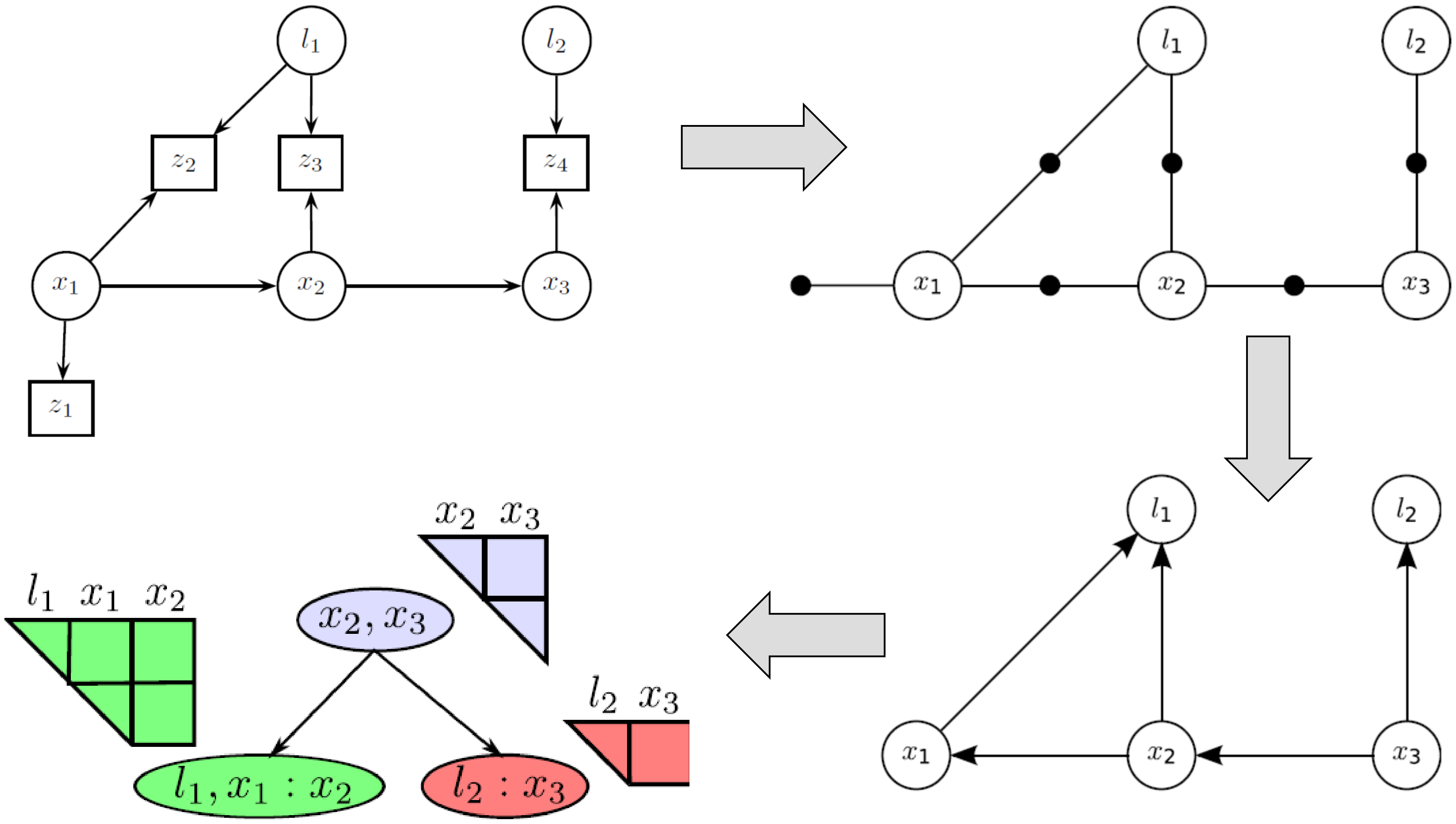




Manhattan dataset (Olson)

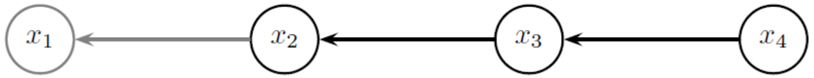
How to update with new measurements / add variables?

From Bayes Net to Bayes Tree

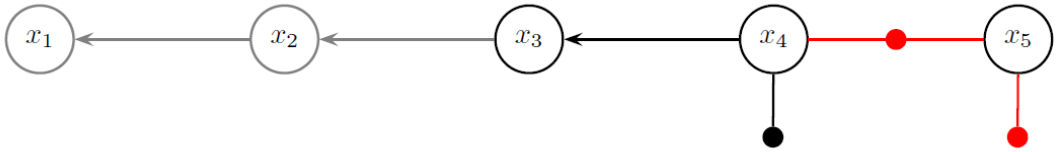


Fixed-lag Smoothing (Linear)

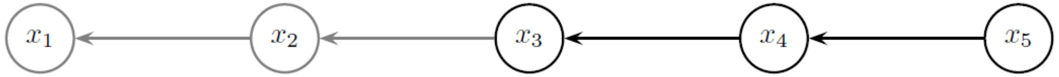
Fully eliminated Bayes net. Dropping x_1 is equivalent to marginalization.



Next time step with new measurements added:



After elimination:



Kalman Filter: Fixed-lag smoother with lag of 1.

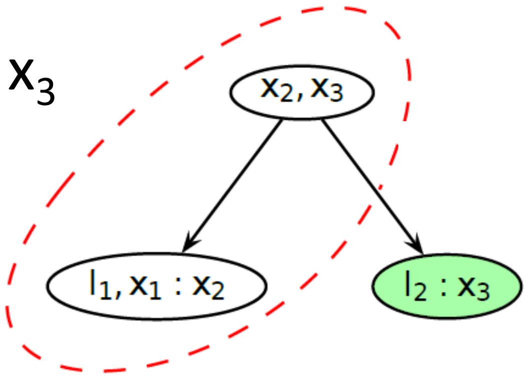
Kalman Filter Algorithm

$$
\begin{aligned}
&\textbf{Kalman\_filter}(\boldsymbol{\mu}_{t-1}, \Sigma_{t-1}, \mathbf{u}_t, \mathbf{z}_t): \\
&\quad \bar{\boldsymbol{\mu}}_t = A_t\, \boldsymbol{\mu}_{t-1} + B_t\, \mathbf{u}_t \\
&\quad \bar{\Sigma}_t = A_t\, \Sigma_{t-1}\, A_t^\top + R_t \\
&\quad K_t = \bar{\Sigma}_t\, C_t^\top (C_t\, \bar{\Sigma}_t\, C_t^\top + Q_t)^{-1} \\
&\quad \boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t + K_t(\mathbf{z}_t - C_t\, \bar{\boldsymbol{\mu}}_t) \\
&\quad \Sigma_t = (I - K_t\, C_t)\, \bar{\Sigma}_t \\
&\quad \mathit{return}\ \boldsymbol{\mu}_t, \Sigma_t
\end{aligned}
$$

The elimination algorithm implements the square root form, also known as square root information filter (SRIF) and smoother (SRIS)

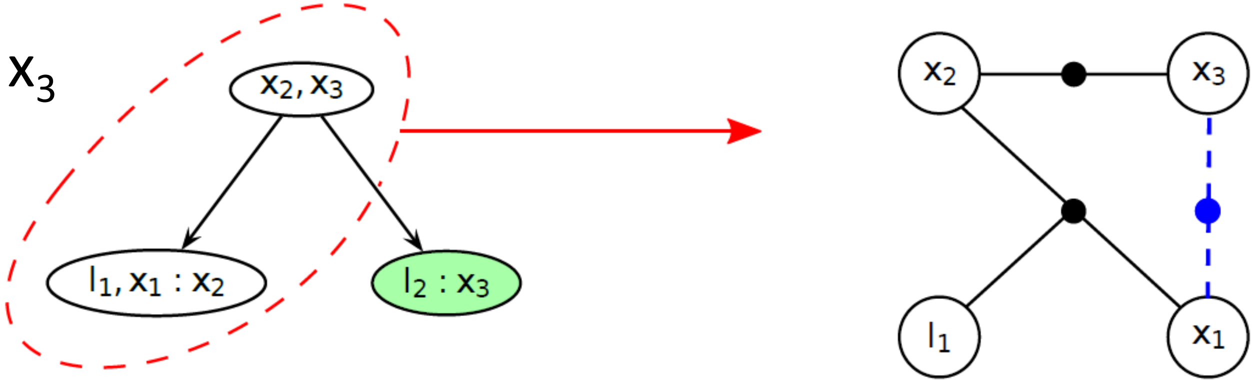
iSAM2: Updating the Bayes Tree

Add new factor between x_1 and x_3



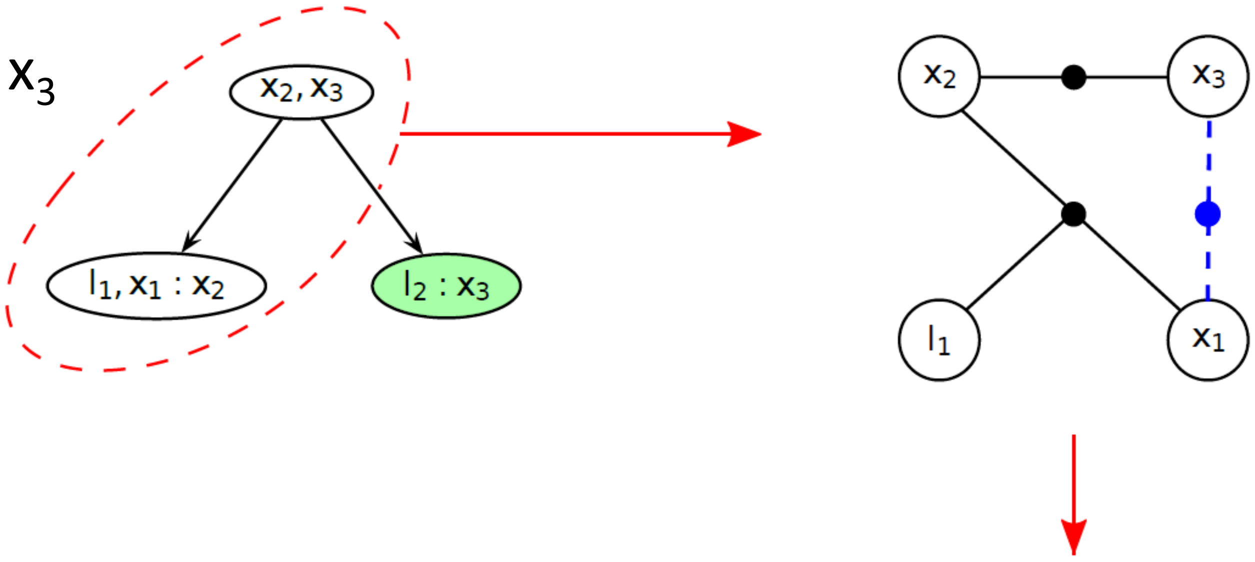
iSAM2: Updating the Bayes Tree

Add new factor
between x_1 and x_3



iSAM2: Updating the Bayes Tree

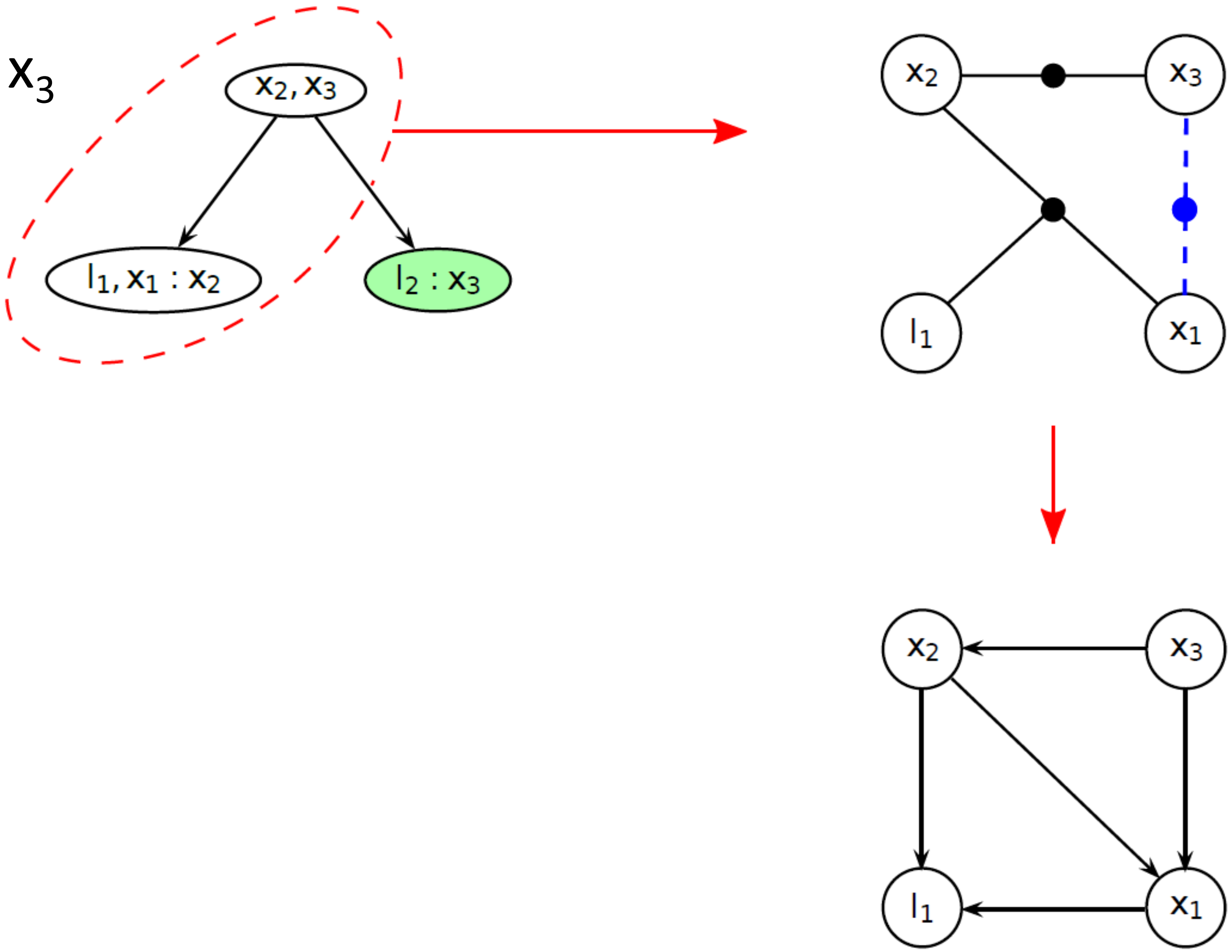
Add new factor between x_1 and x_3



On the board

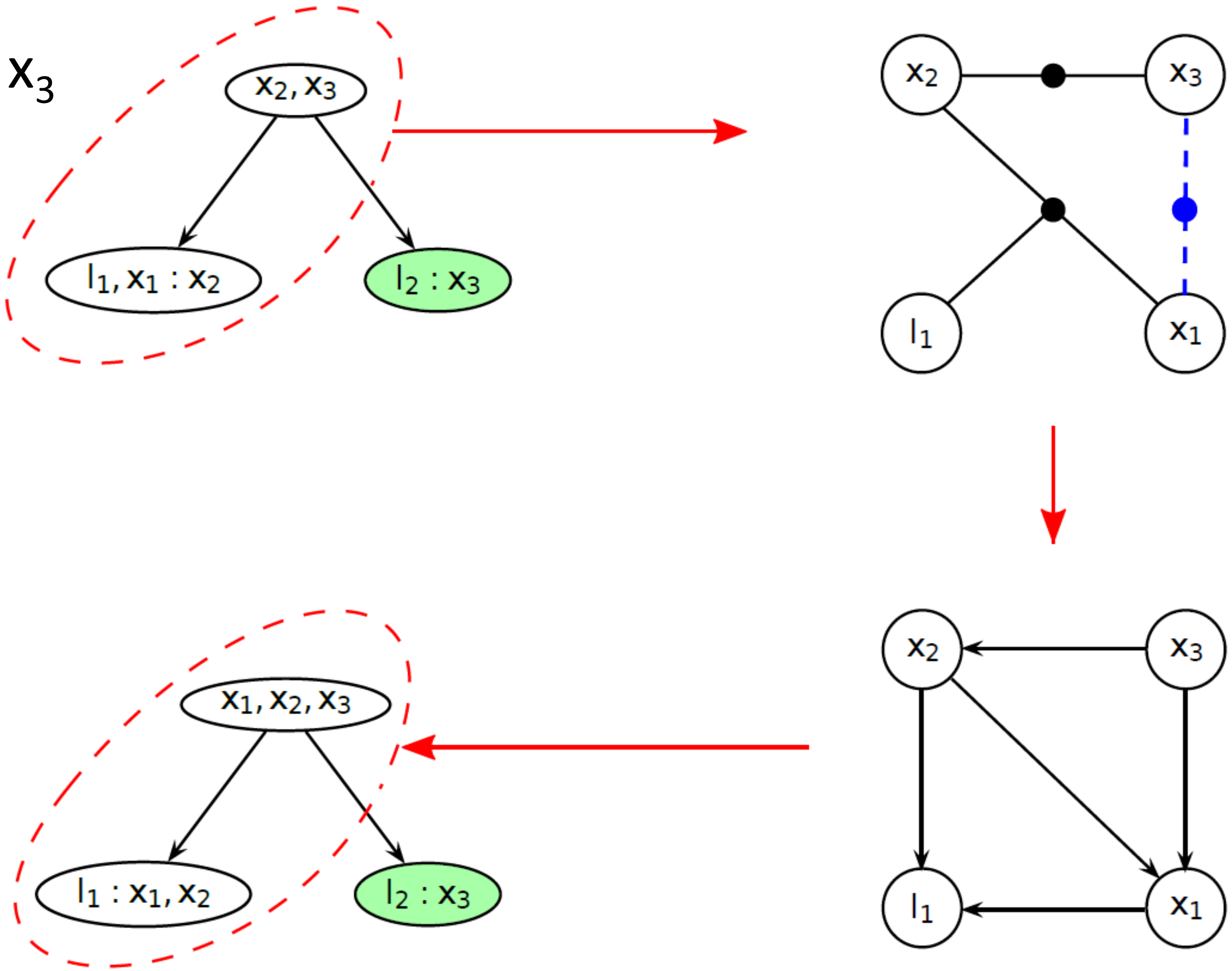
iSAM2: Updating the Bayes Tree

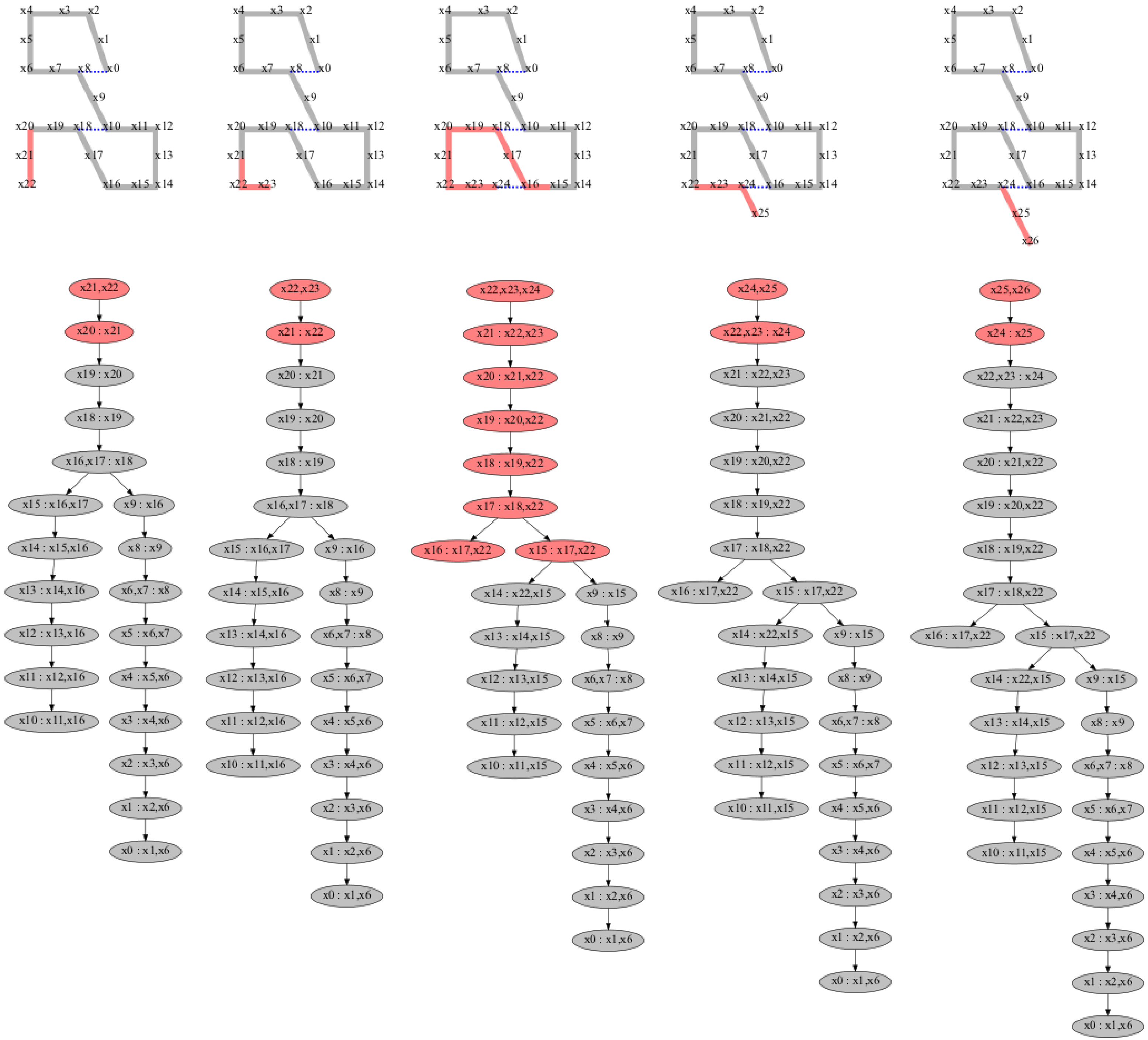
Add new factor between x_1 and x_3



iSAM2: Updating the Bayes Tree

Add new factor
between x_1 and x_3



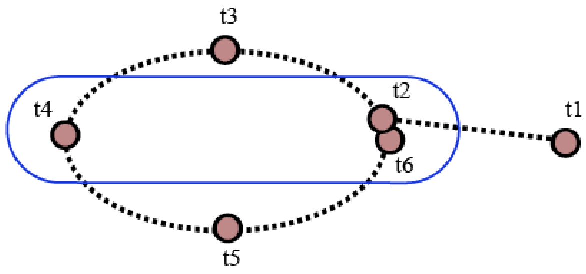

x21,x22
x20 : x21
x19 : x20
x18 : x19
x16,x17 : x18
x15 : x16,x17
x9 : x16
x14 : x15,x16
x8 : x9
x13 : x14,x16
x6,x7 : x8
x12 : x13,x16
x5 : x6,x7
x11 : x12,x16
x4 : x5,x6
x10 : x11,x16
x3 : x4,x6
x2 : x3,x6
x1 : x2,x6
x0 : x1,x6
x22,x23
x21 : x22
x20 : x21
x19 : x20
x18 : x19
x16,x17 : x18
x15 : x16,x17
x9 : x16
x14 : x15,x16
x8 : x9
x13 : x14,x16
x6,x7 : x8
x12 : x13,x16
x5 : x6,x7
x11 : x12,x16
x4 : x5,x6
x10 : x11,x16
x3 : x4,x6
x2 : x3,x6
x1 : x2,x6
x0 : x1,x6
x22,x23,x24
x21 : x22,x23
x20 : x21,x22
x19 : x20,x22
x18 : x19,x22
x17 : x18,x22
x16 : x17,x22
x15 : x17,x22
x14 : x22,x15
x9 : x15
x13 : x14,x15
x8 : x9
x12 : x13,x15
x6,x7 : x8
x11 : x12,x15
x5 : x6,x7
x10 : x11,x15
x4 : x5,x6
x3 : x4,x6
x2 : x3,x6
x1 : x2,x6
x0 : x1,x6
x24,x25
x22,x23 : x24
x21 : x22,x23
x20 : x21,x22
x19 : x20,x22
x18 : x19,x22
x17 : x18,x22
x16 : x17,x22
x15 : x17,x22
x14 : x22,x15
x9 : x15
x13 : x14,x15
x8 : x9
x12 : x13,x15
x6,x7 : x8
x11 : x12,x15
x5 : x6,x7
x10 : x11,x15
x4 : x5,x6
x3 : x4,x6
x2 : x3,x6
x1 : x2,x6
x0 : x1,x6
x25,x26
x24 : x25
x22,x23 : x24
x21 : x22,x23
x20 : x21,x22
x19 : x20,x22
x18 : x19,x22
x17 : x18,x22
x16 : x17,x22
x15 : x17,x22
x14 : x22,x15
x9 : x15
x13 : x14,x15
x8 : x9
x12 : x13,x15
x6,x7 : x8
x11 : x12,x15
x5 : x6,x7
x10 : x11,x15
x4 : x5,x6
x3 : x4,x6
x2 : x3,x6
x1 : x2,x6
x0 : x1,x6

Incremental Variable Reordering

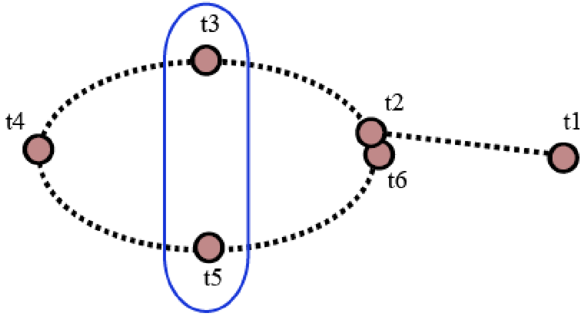
For a small loop, what constitutes a "good" ordering?

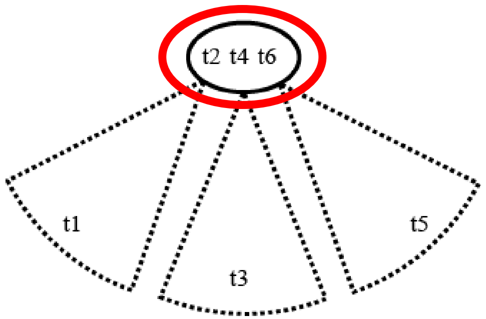
Include loop closing into cut

Loop closing not part of cut



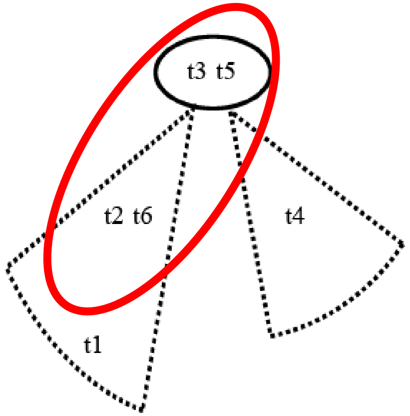
Trajectory





Affected by next update

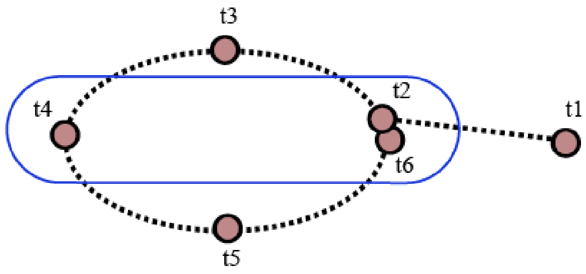
Bayes tree

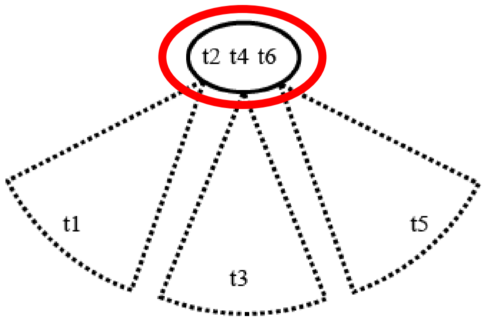


Incremental Variable Reordering

Most recent variable at the end

expected to make future updates cheaper





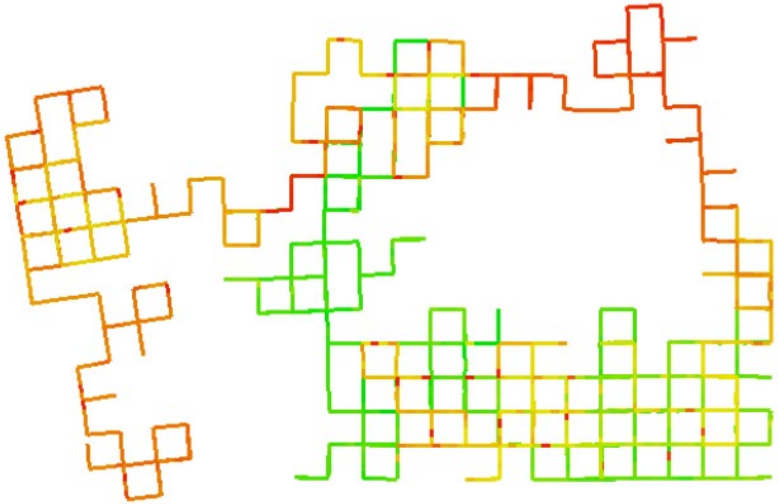
- Force most recent variables to the end
- Find best ordering for remaining variables

Using constrained version of COLAMD algorithm (CCOLAMD)

Variable Reordering – Constrained COLAMD

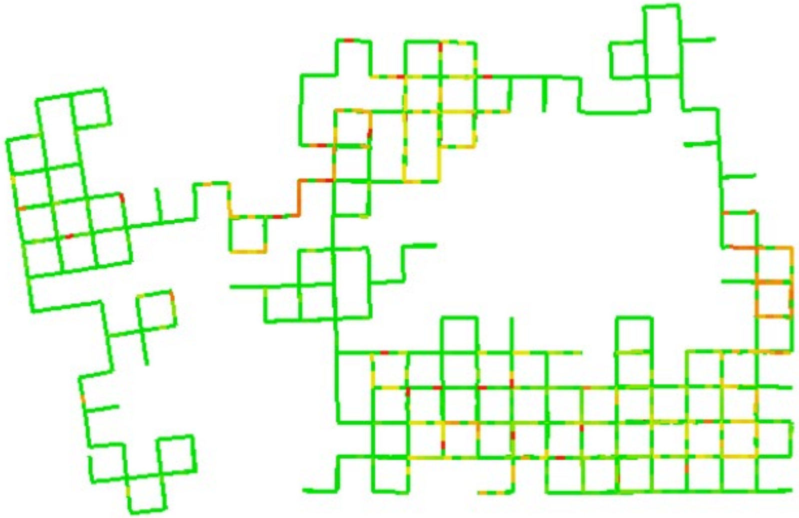
Greedy approach

Arbitrary placement of newest variable



Constrained Ordering

Newest variables forced to the end



Number of affected variables:

low high

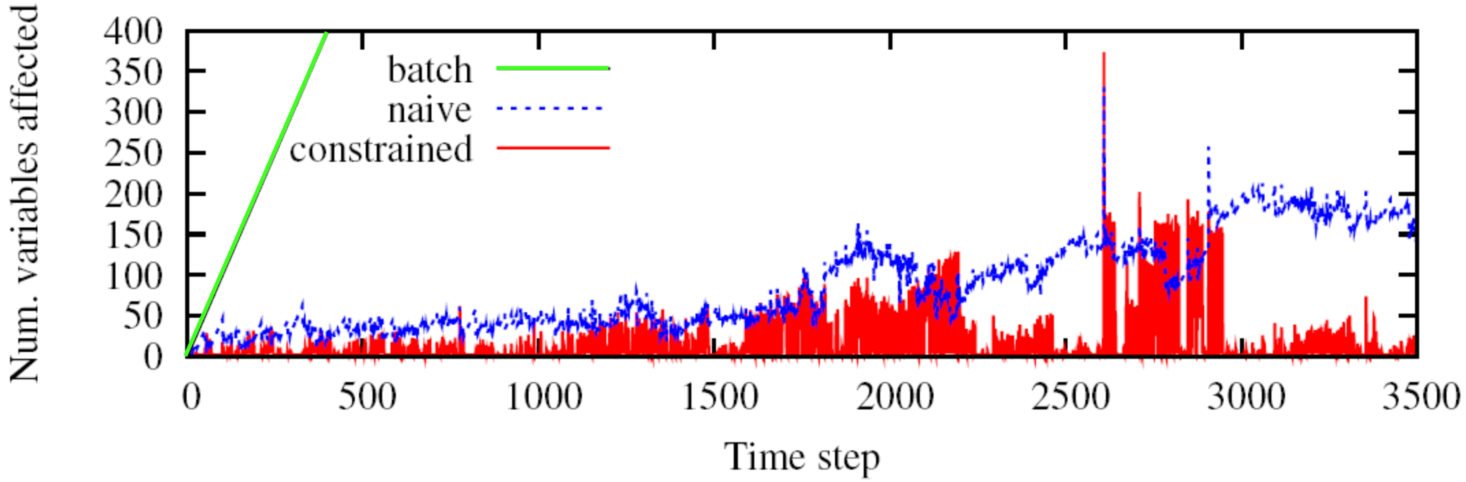
Much cheaper!

iSAM2: Incremental Update + Variable Ordering

Variable ordering changes incrementally during update

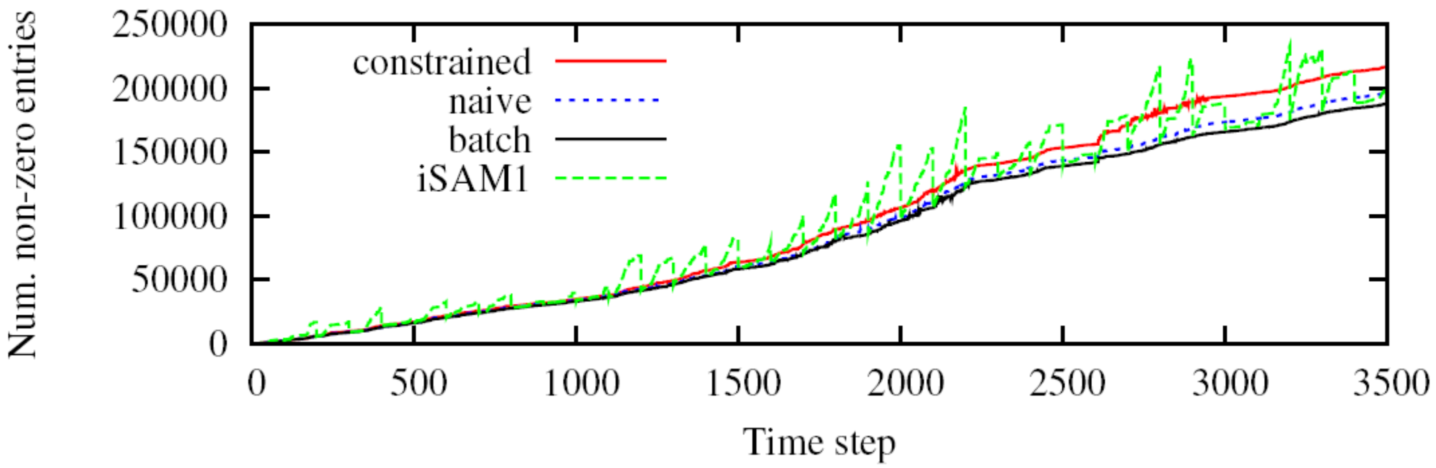
- Not understood in matrix version
- Sparse matrix data structure not suitable

Large savings in computation



Variable Reordering – Fill-in

Incremental ordering still yields good overall ordering



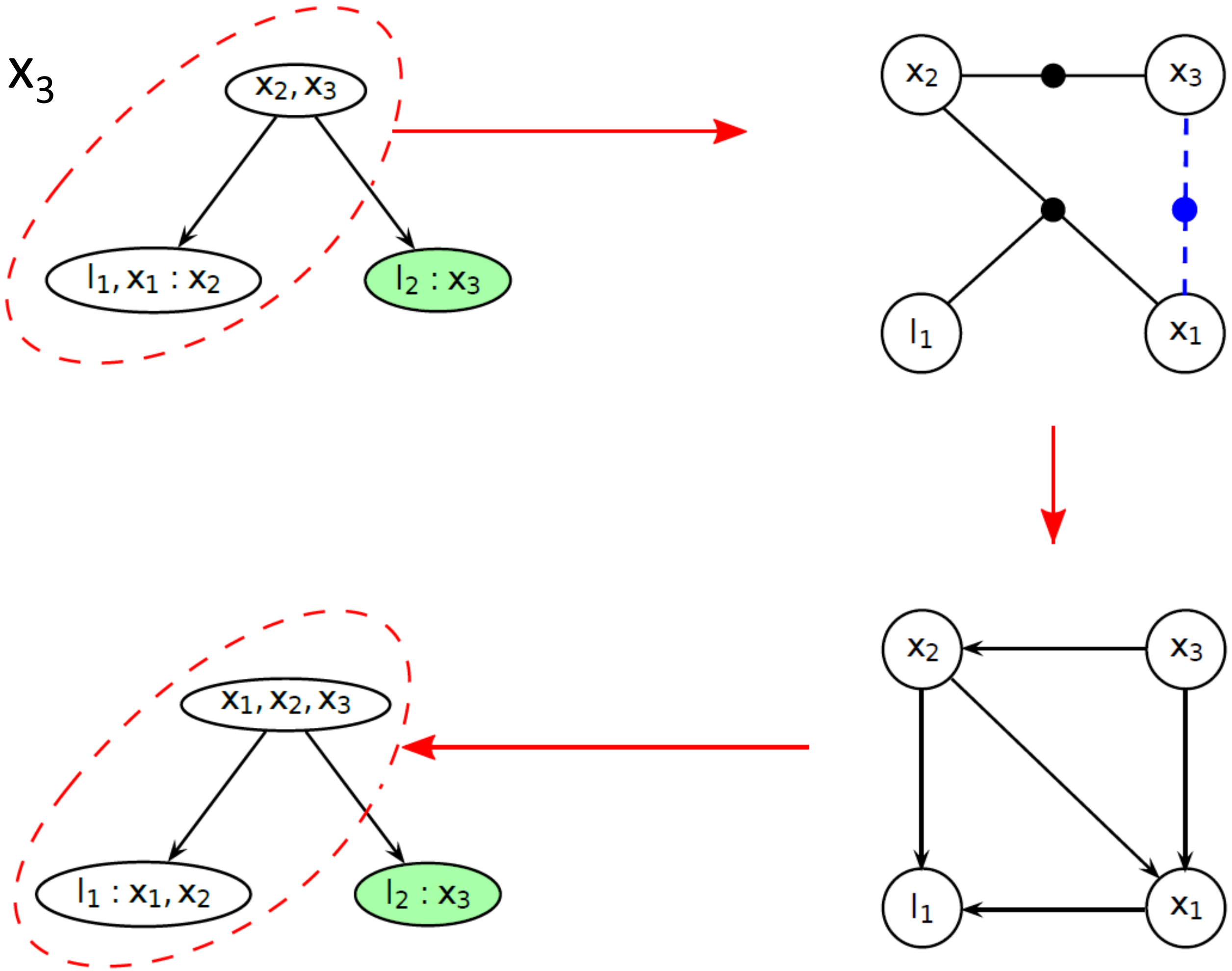
- Only slightly more fill-in than batch COLAMD ordering
- Constrained ordering is worse than naïve/greedy:
  - Suboptimal ordering because of partial constraint,
    but cheaper to update!

iSAM2: Nonlinear Updates

- The Bayes tree contains linearized information
- We have to re-linearize the original nonlinear factors!

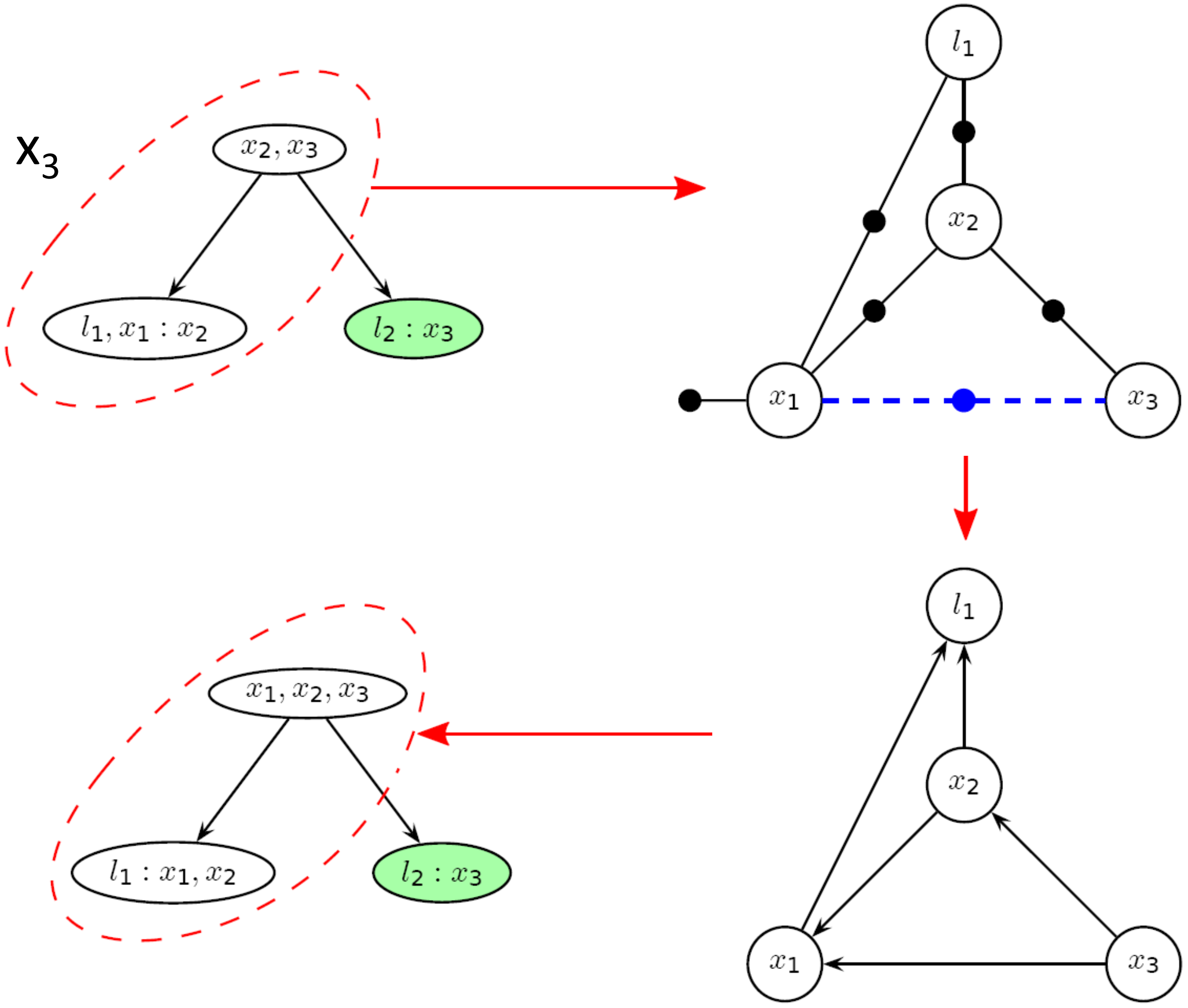
iSAM2: Updating the Bayes Tree (Linear)

Add new factor between x_1 and x_3



iSAM2: Updating the Bayes Tree (Nonlinear)

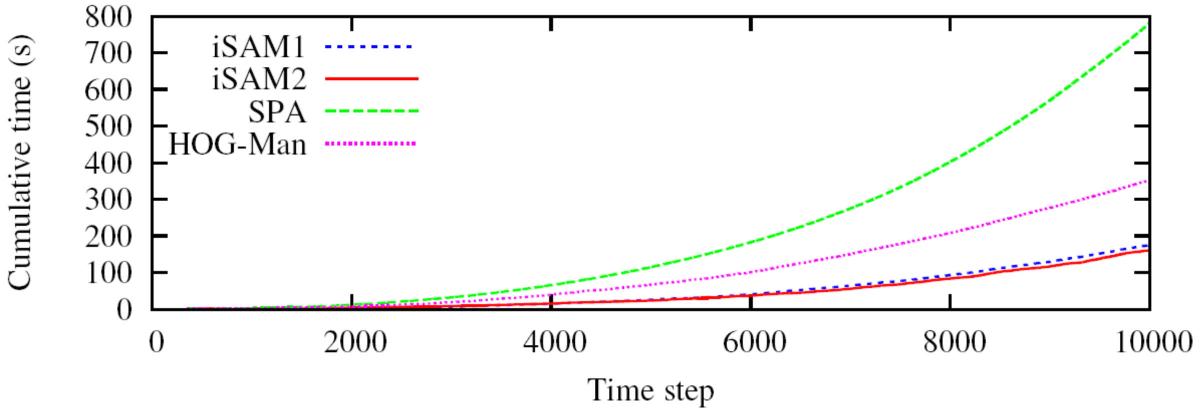
Add new factor between x_1 and x_3

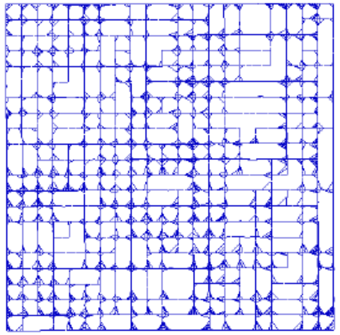


iSAM2: Fluid Relinearization

Relinearize select variables only

- Changes in map estimates are often local
- Most variables do not need to be updated
- Can be combined with updates





City 10000 dataset

iSAM1: Kaess et al., TRO 08
iSAM2: Kaess et al., IJRR 12
SPA: Konolige et al., IROS 2010
HOG-Man: Grisetti et al., ICRA 2010

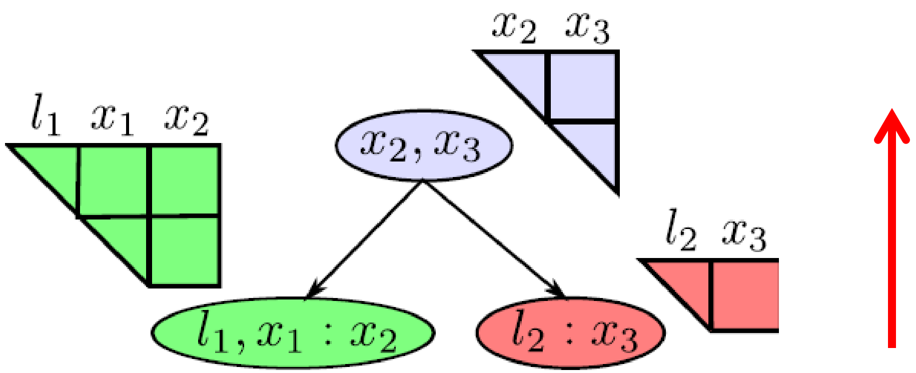
Running Intersection Property

If variable x is part of two cliques C_1 and C_2, then x is part of every clique on the unique path between C_1 and C_2

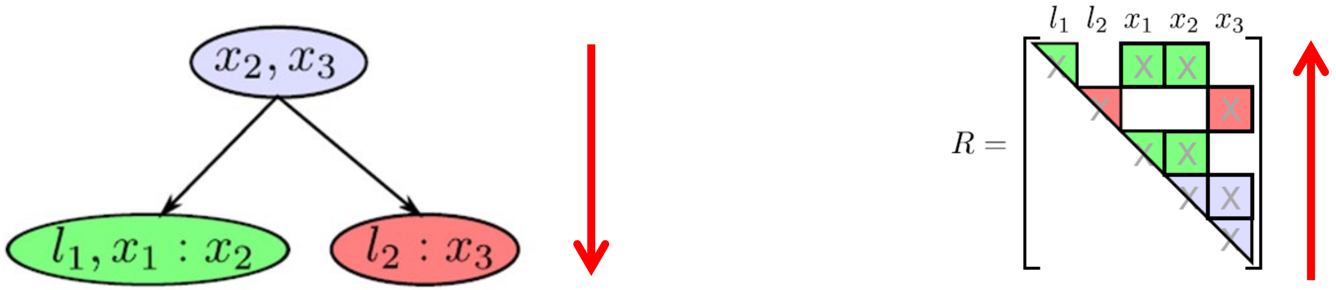
We can efficiently find any occurrence of a variable x in the tree by starting at the clique where it is eliminated and recursively traversing each subtree until the variable disappears

Backsubstitution in the Graph

- Inference is a two-step process:
  - Elimination starts at leaves and proceeds to the root

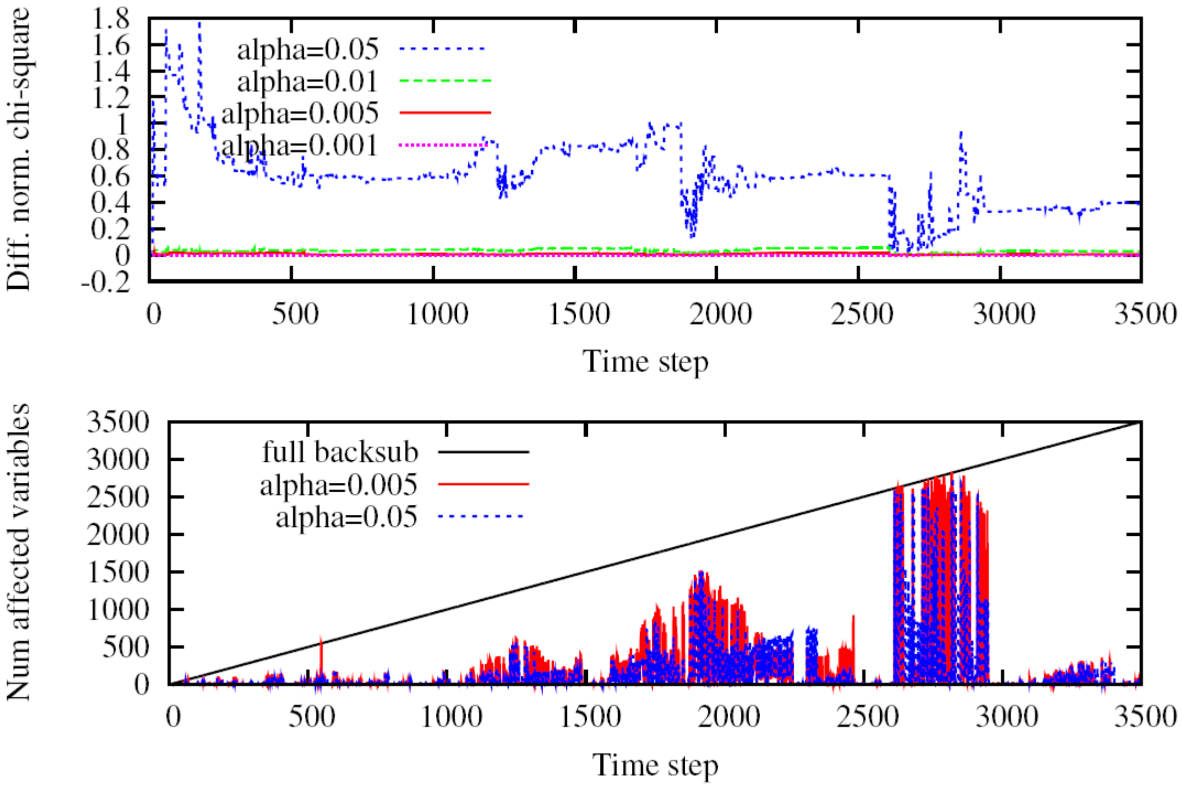


  - Solving starts at root and proceeds to the leaves

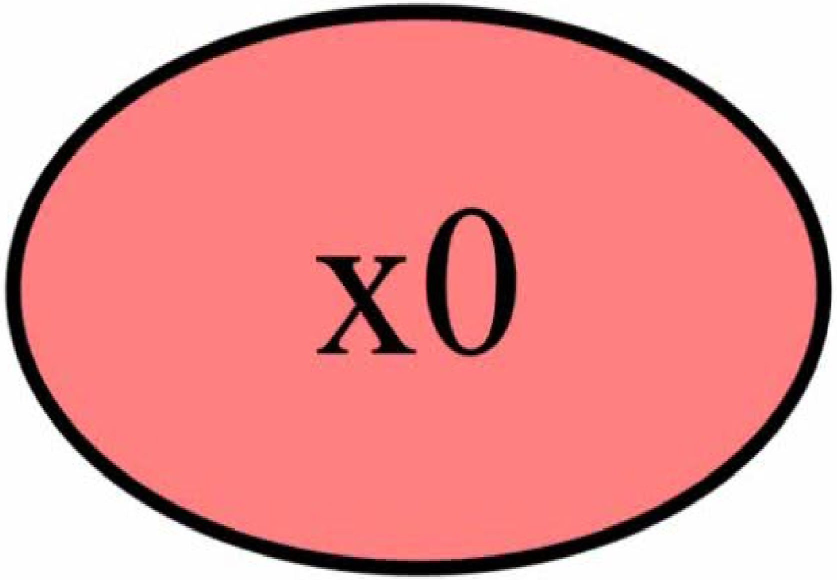


Selective Variable Recovery

Again good quality and low cost are achievable:

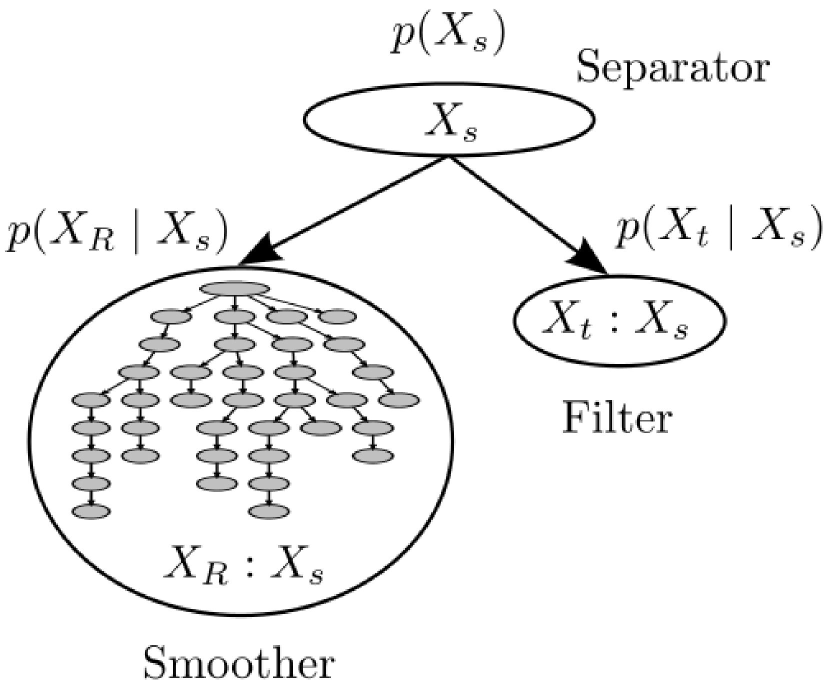


iSAM2: Bayes Tree for Manhattan Sequence



Custom Variable Ordering for Parallelization

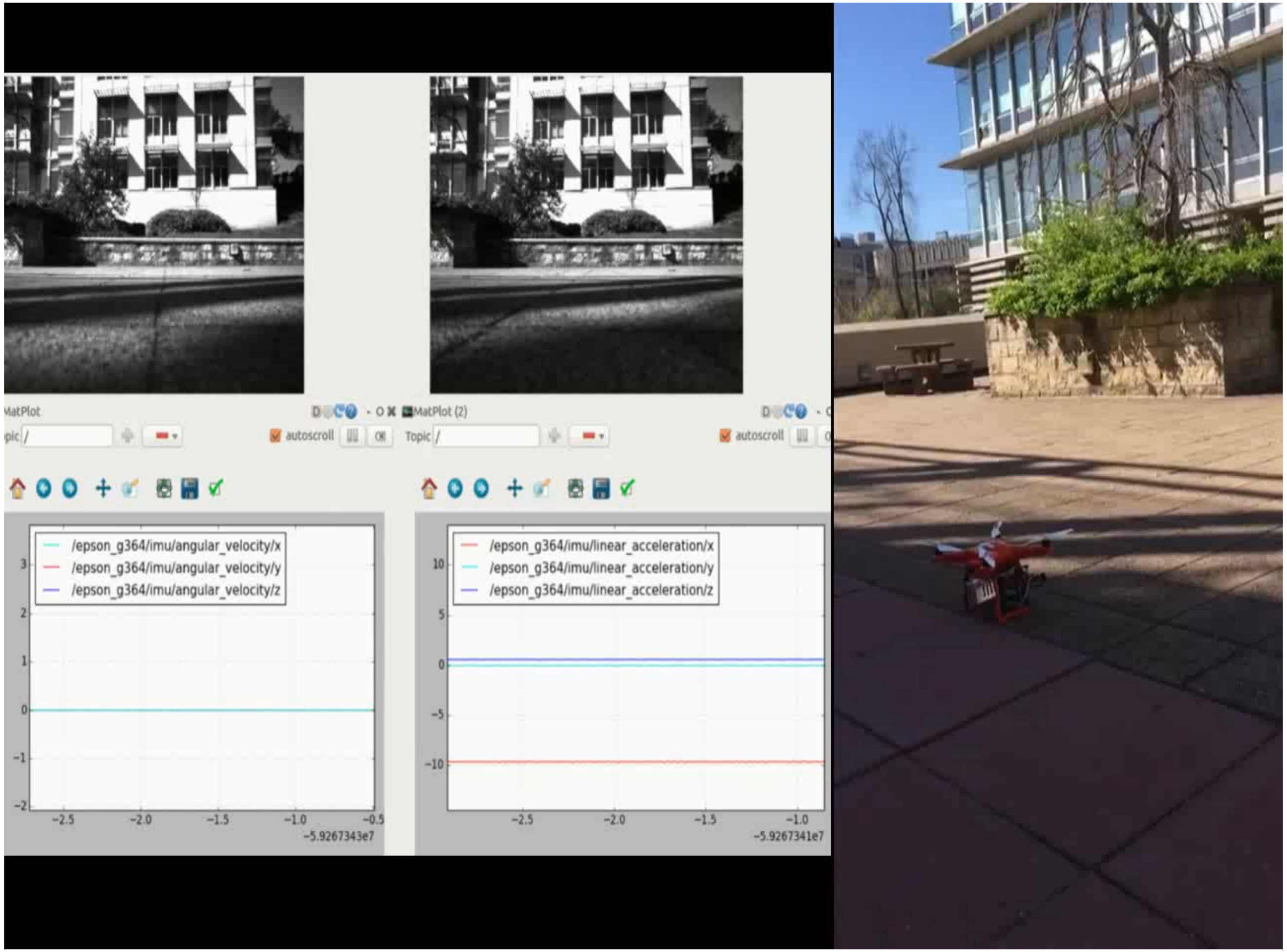
- Combining filtering (constant time updates) and smoothing (loop closure capabilities)
- Concurrent updates to single Bayes tree formulation



Open-Source Libraries

- iSAM1
  - Matrix-based
  - LGPL licensed C++ library with minimal dependencies
  - http://people.csail.mit.edu/kaess/isam/
- GTSAM
  - Graph-based
  - BSD-licensed C++ library
  - Implements iSAM2
  - https://github.com/borglab/gtsam
- OpenSAM Foundation announced November 2019
  - GTSAM compatible
  - For embedded systems
  - Strict coding standards for industry distribution
- Related libraries
  - g2o: https://openslam-org.github.io/g2o
  - Ceres Solver: https://github.com/ceres-solver/ceres-solver
  - SLAM++: https://sourceforge.net/projects/slam-plus-plus/

Application: Visual-Inertial Odometry (VIO)



Application: Visual-Inertial Odometry (VIO)

- Fundamental algorithm for state estimation of mobile devices, robots, VR, AR,…
- Track pose (position+orientation) of rigidly mounted camera+IMU
- Combines complementary advantages of two sensors:

Camera

Measures linear (up to scale) and angular velocity

+

Inertial Measurement Unit (IMU)

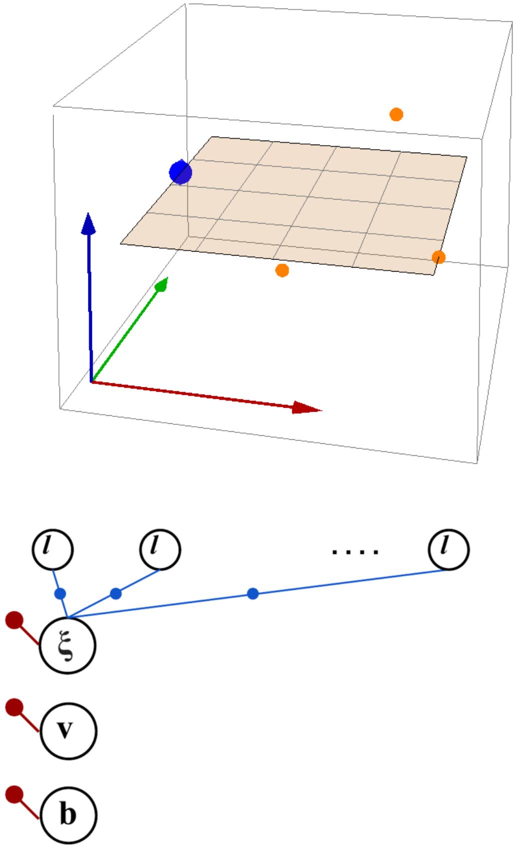
Measures linear acceleration and angular velocity

Allows continuous calibration of IMU

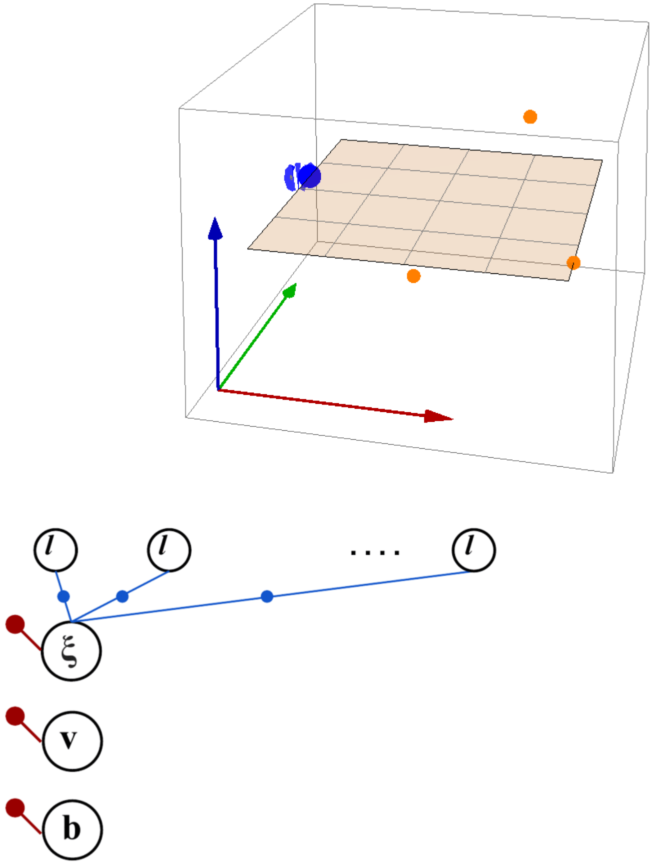
Works even when vision fails (motion blur, lack of texture)

Fixed-Lag Smoothing for Real-time VIO

Full (batch) smoothing



Fixed-Lag Smoothing

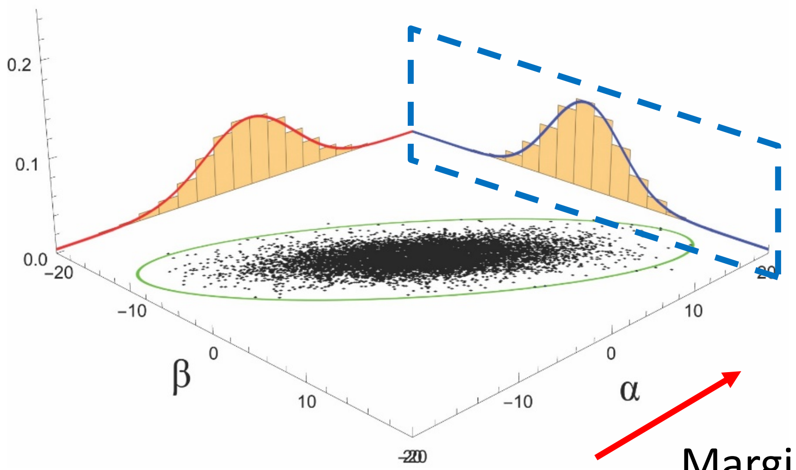


Slide courtesy of Jerry Hsiung

Marginalization 2D Example

$$\mu_{\alpha,\beta} = \begin{bmatrix} \mu_\alpha \\ \mu_\beta \end{bmatrix}, \quad \Sigma_{\alpha,\beta} = \begin{bmatrix} \Sigma_{\alpha\alpha} & \Sigma_{\alpha\beta} \\ \Sigma_{\beta\alpha} & \Sigma_{\beta\beta} \end{bmatrix}$$

$$\alpha, \beta \sim \mathcal{N}(\mu_{\alpha,\beta}, \Sigma_{\alpha,\beta})$$



Marginalize α

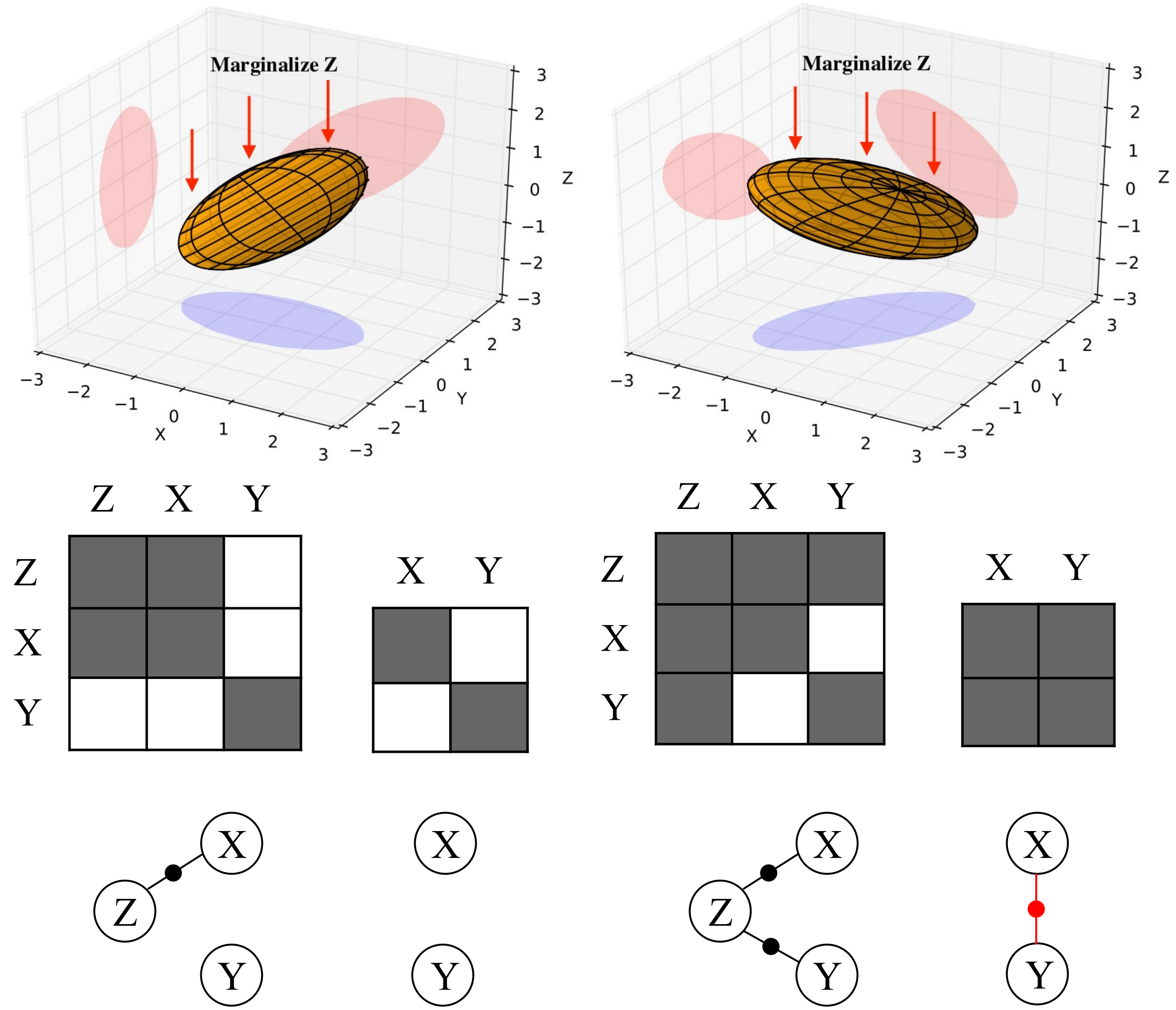
Covariance

Information

$$\Sigma_{\alpha,\beta} = \begin{bmatrix} \Sigma_{\alpha\alpha} & \Sigma_{\alpha\beta} \\ \Sigma_{\beta\alpha} & \Sigma_{\beta\beta} \end{bmatrix} = \begin{bmatrix} \Lambda_{\alpha\alpha} & \Lambda_{\alpha\beta} \\ \Lambda_{\beta\alpha} & \Lambda_{\beta\beta} \end{bmatrix}^{-1}$$

$$\Sigma_{\beta\beta} = \Lambda'^{-1}_{\beta\beta} = (\Lambda_{\beta\beta} - \Lambda_{\beta\alpha}\Lambda^{-1}_{\alpha\alpha}\Lambda_{\alpha\beta})^{-1}$$ "Schur complement"

Marginalization 3D Example



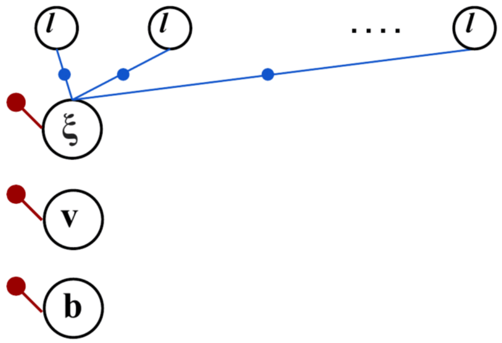
- Marginalization creates "fill-in"
- Only variables connected to Z affected (Markov blanket)

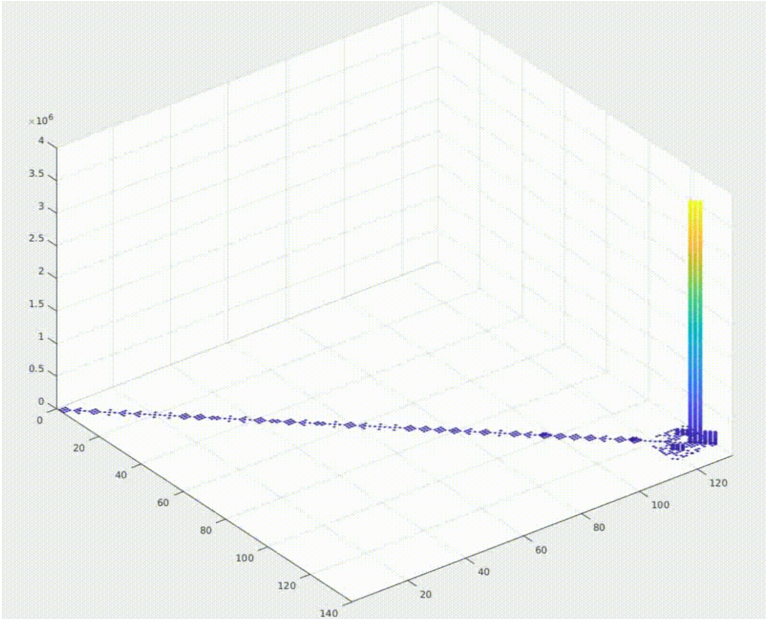
Information matrices

Factor graphs

Marginalization is Problematic

The information matrix is no longer sparse!

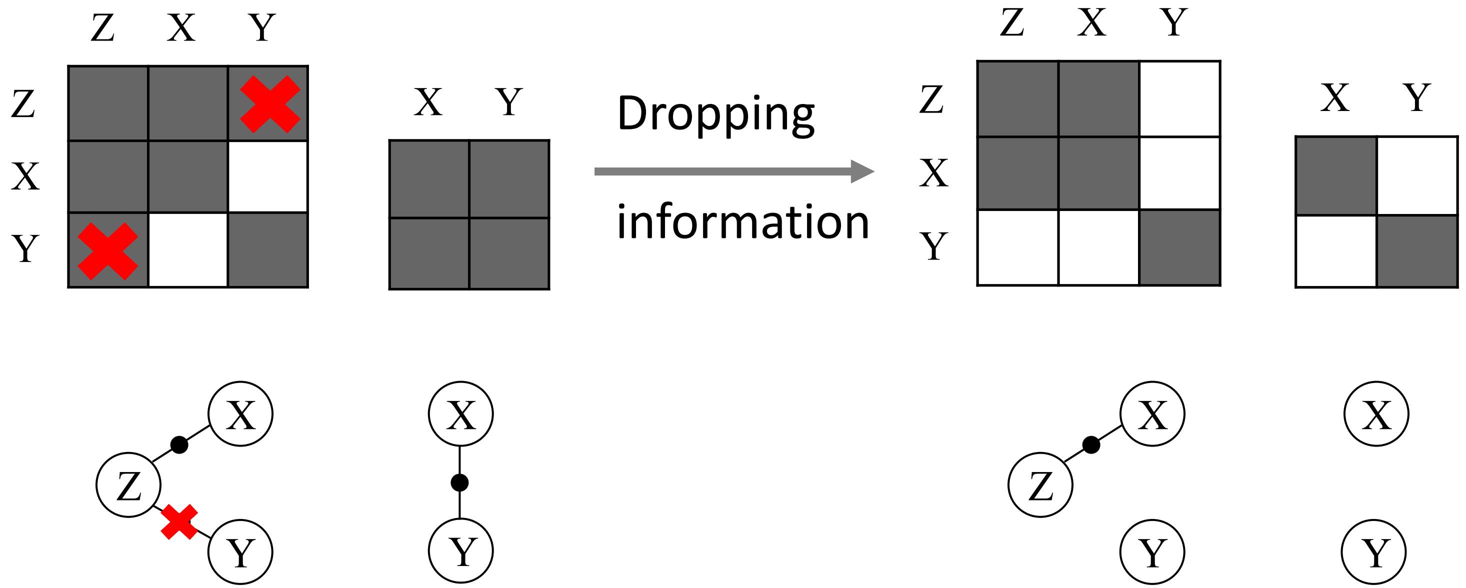




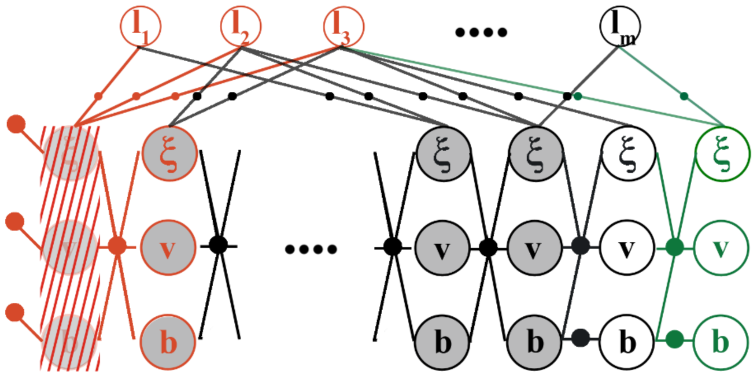
Too expensive for onboard state estimation

Existing Methods are not Optimal

Existing methods discard measurements or marginalize more variables



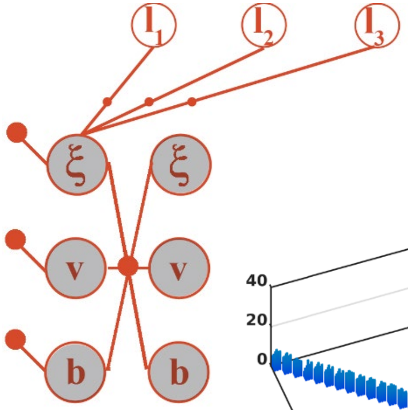
VIO Marginalization

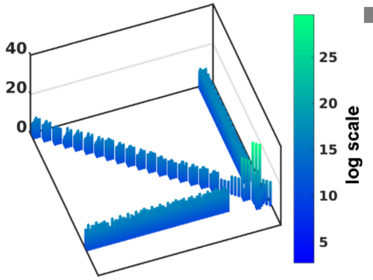


Marginalized states

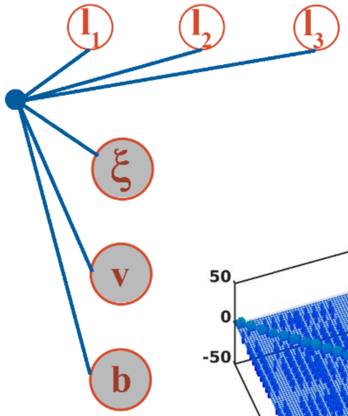
Incoming states

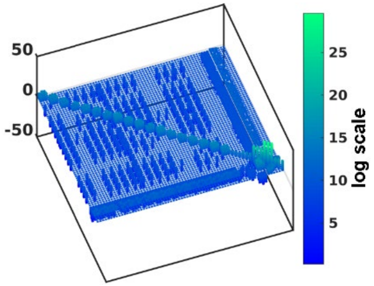
Markov Blanket





Marginalization

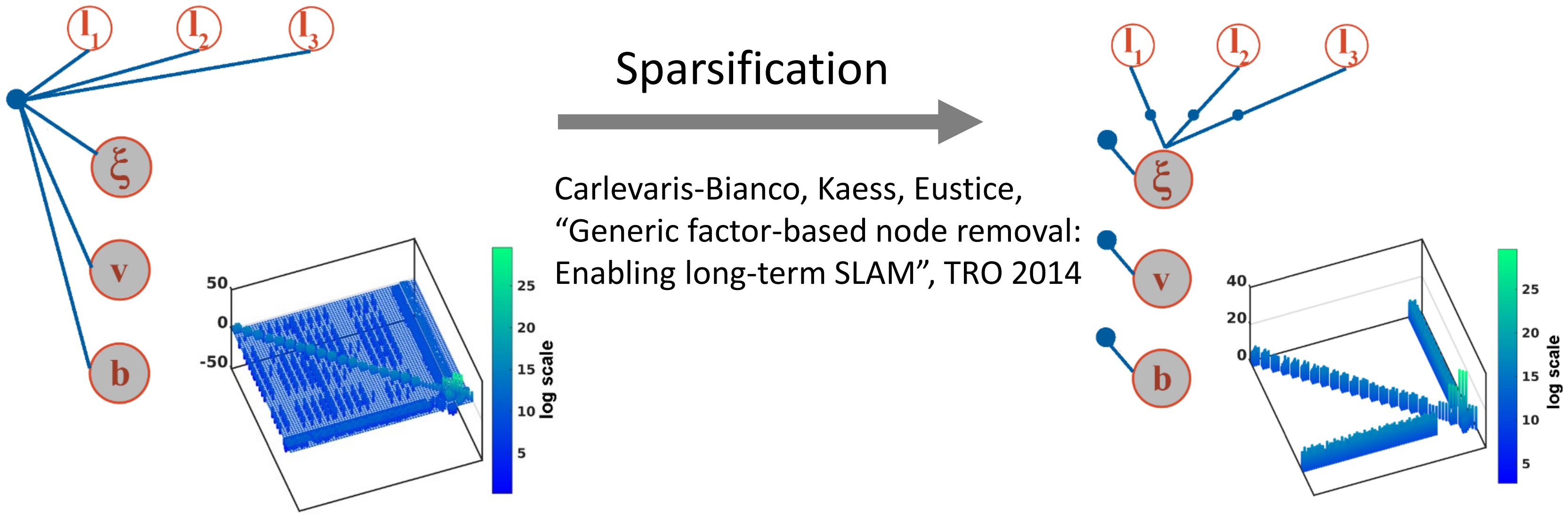




Information Sparsification

Research Question:

Can we use a sparse graph to approximate the dense one?



Carlevaris-Bianco, Kaess, Eustice, "Generic factor-based node removal: Enabling long-term SLAM", TRO 2014

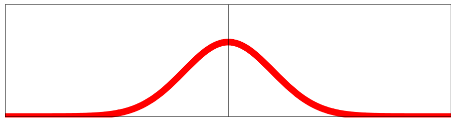
Slide courtesy of Jerry Hsiung

Information Sparsification

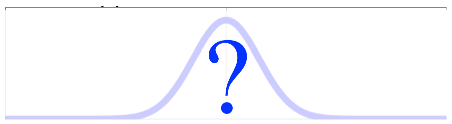
Minimizing Kullback-Leibler Divergence (KLD)

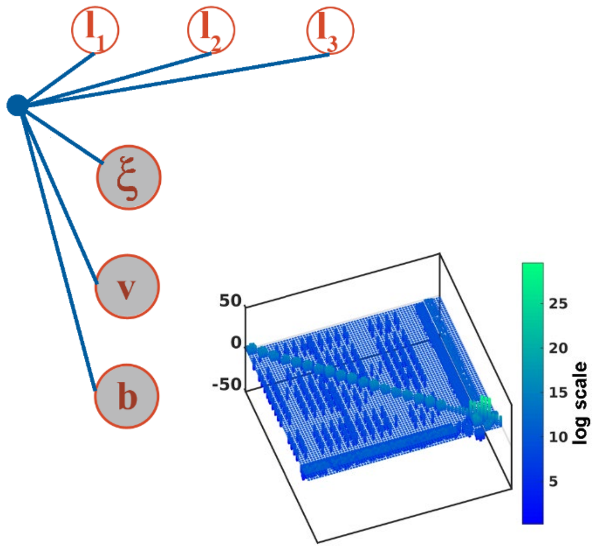
$$D_{KL}(p(\mathcal{X}_{\mathrm{t}})\|p_{\mathrm{s}}(\mathcal{X}_{\mathrm{t}})) = \frac{1}{2}\left(\langle\Lambda_{\mathrm{s}}, \Sigma_{\mathrm{t}}\rangle - \log\det(\Lambda_{\mathrm{s}}) + \|\Lambda_{\mathrm{s}}^{\frac{1}{2}}(\mu_{\mathrm{s}} - \mu_{\mathrm{t}})\|_2^2 - d\right)$$

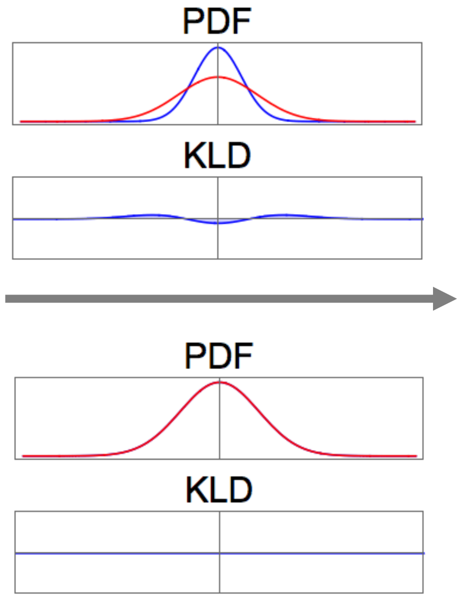
Target Distribution

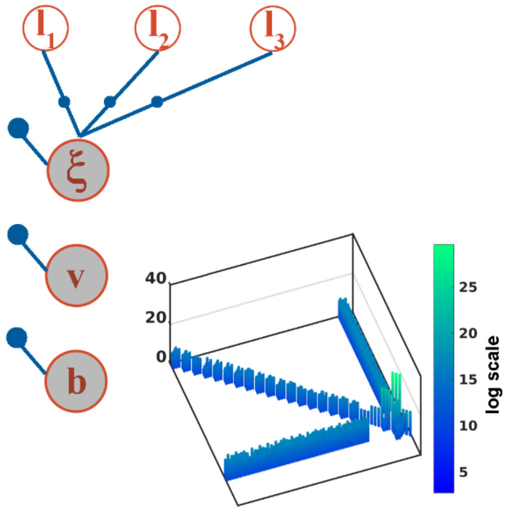


Approximate Distribution

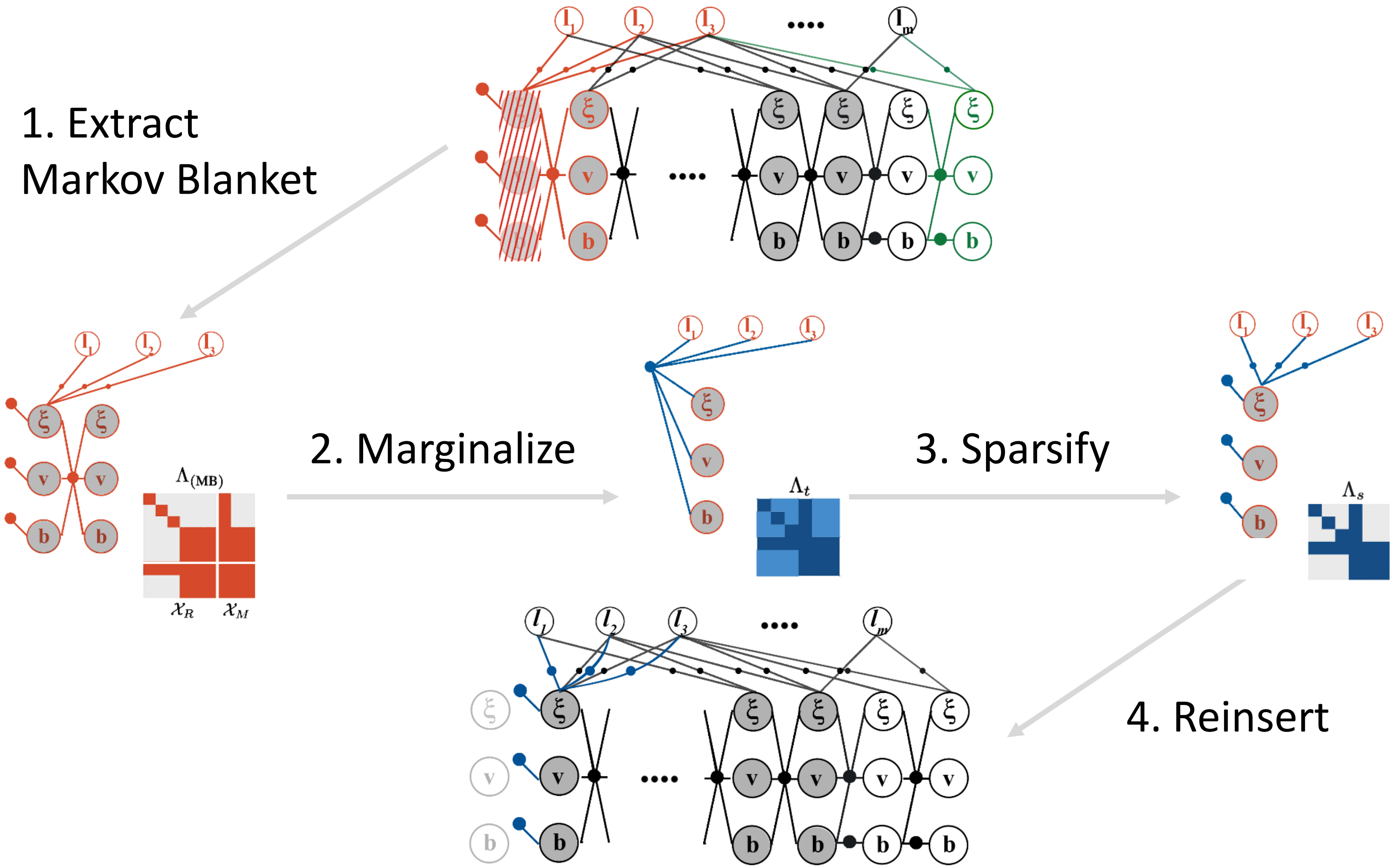








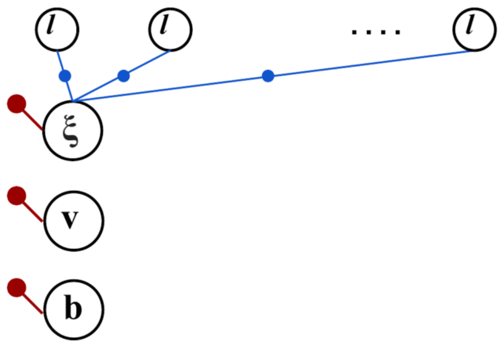
The Proposed VIO Sparsification Framework



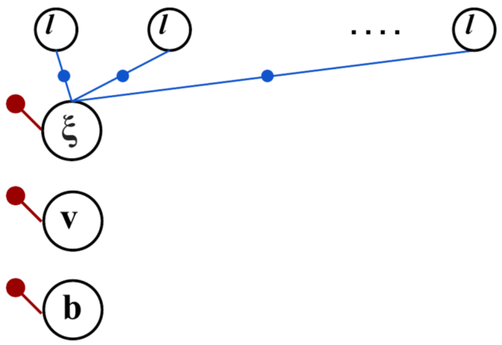
Jerry Hsiung, Ming Hsiao, Eric Westman, Rafael Valencia, Michael Kaess
"Information Sparsification in Visual-Inertial Odometry", IROS 2018

Sparsification in Visual Inertial Odometry

Fixed-Lag Smoothing



Sparsified Fixed-Lag Smoothing



Comparing to a regular Fixed-Lag Smoother:

+ Preserve sparsity.

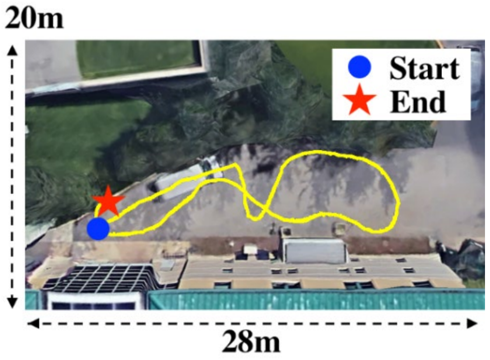
Comparing to OKVIS [1], VINS-MONO [2]:

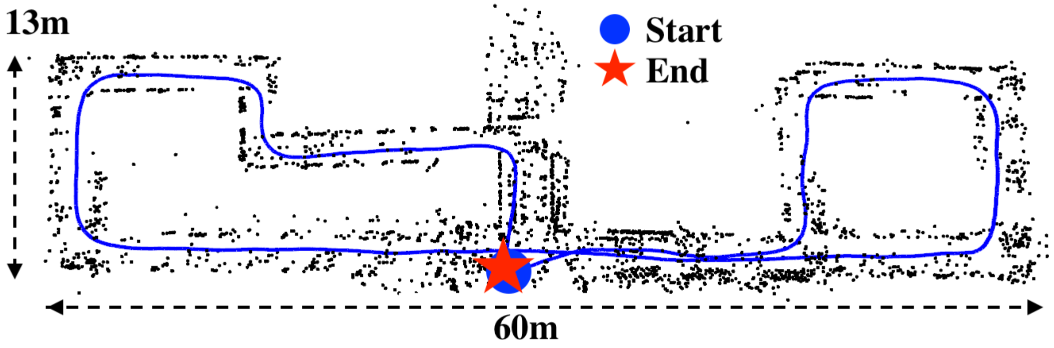
+ Preserve all measurements.
+ Variables remain optimizable.

[1] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual-inertial odometry using nonlinear optimization", IJRR 2015

[2] T. Qin, P. Li, and S. Shen, "VINS-Mono: A robust and versatile monocular visual-inertial state estimator", TRO 2018

Experiments – Flight Tests

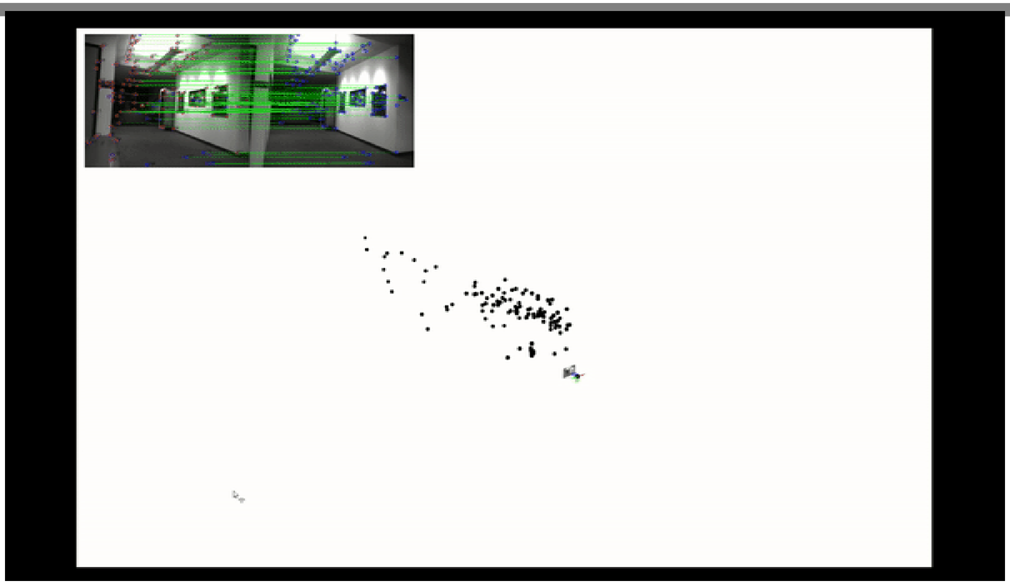




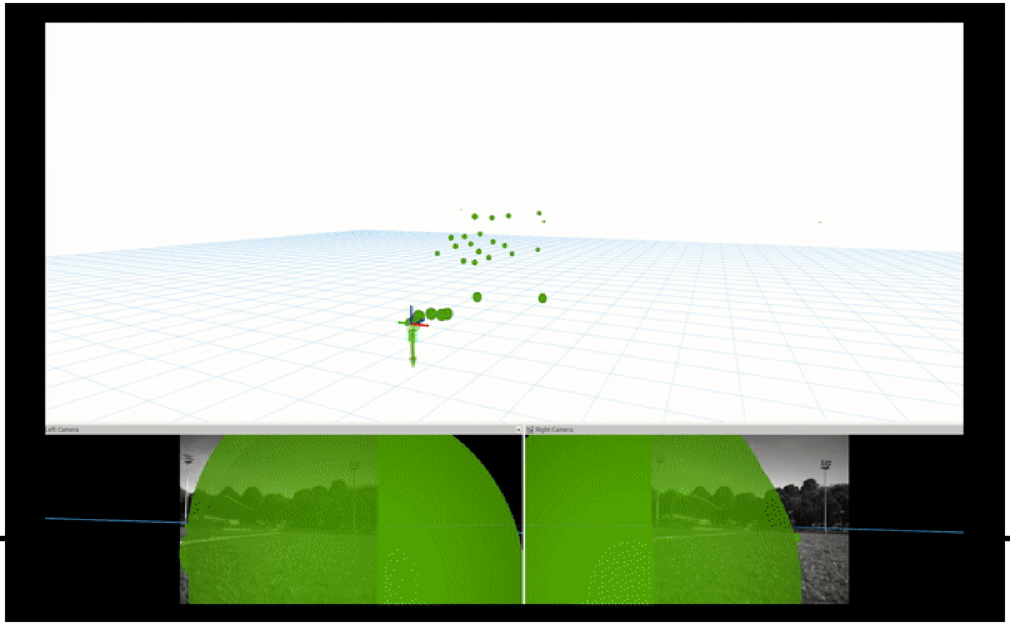
Jerry Hsiung, Ming Hsiao, Eric Westman, Rafael Valencia, Michael Kaess
"Information Sparsification in Visual-Inertial Odometry", IROS 2018

Experiments – Flight Tests

Indoor

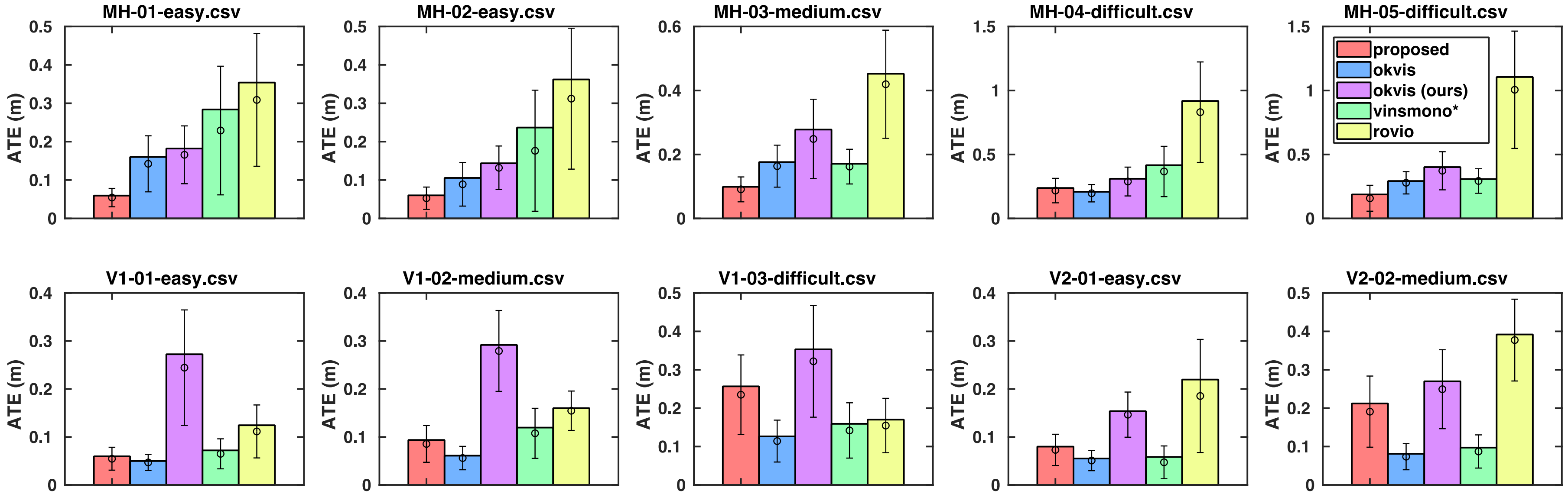


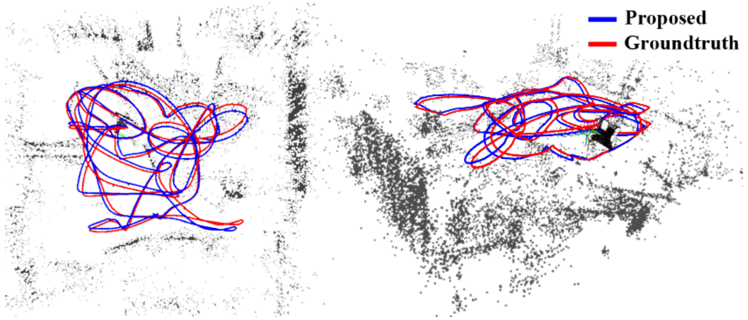
Outdoor



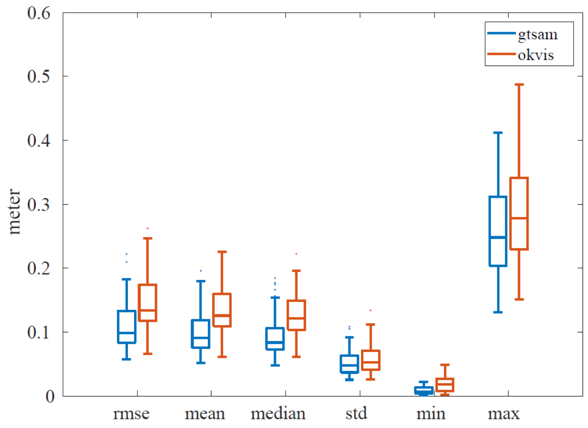
Experiments – Benchmark

EuRoC Dataset:





45 Monte Carlo simulations:



Jerry Hsiung, Ming Hsiao, Eric Westman, Rafael Valencia, Michael Kaess
“Information Sparsification in Visual-Inertial Odometry”, IROS 2018