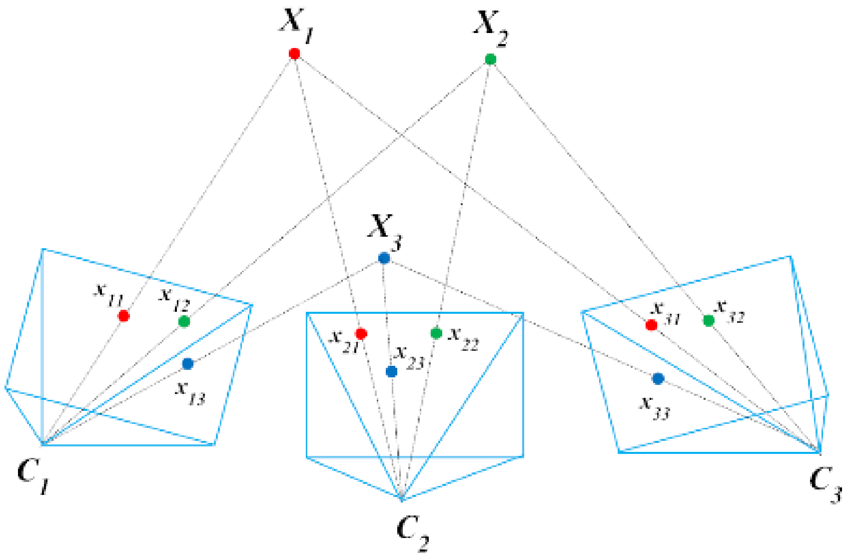
集束调整

章国锋
浙江大学CAD&CG国家重点实验室

Bundle Adjustment

- Jointly optimize all cameras and points

$$\arg\min_{C_1,\ldots C_{N_c}, X_1,\ldots,X_{N_p}} \sum \left\| \pi(X_i, C_j) - x_{ij} \right\|^2$$



Triggs, B., Mclauchlan, P., Hartley, R., and Fitzgibbon, A. 1999. Bundle adjustment—a modern synthesis. In Proceedings of the International Workshop on Vision Algorithms: Theory and Practice. 298–372.

Nonlinear Least Squares

- Gaussian Newton

$$x^* = \arg\min_x \|\varepsilon(x)\|^2$$

$$\varepsilon(x^*) = \varepsilon(\hat{x} + \delta_x) \approx \varepsilon(\hat{x}) + J\delta_x$$

$$J = \partial\varepsilon / \partial x\big|_{x=\hat{x}}$$ Jacobian matrix

$$\boxed{J^T J}\delta_x = -J^T \varepsilon(\hat{x})$$

first order approximation to Hessian

- Levenberg-Marquardt

$$(J^T J + \mu I)\delta x = -J^T \varepsilon(\hat{x})$$

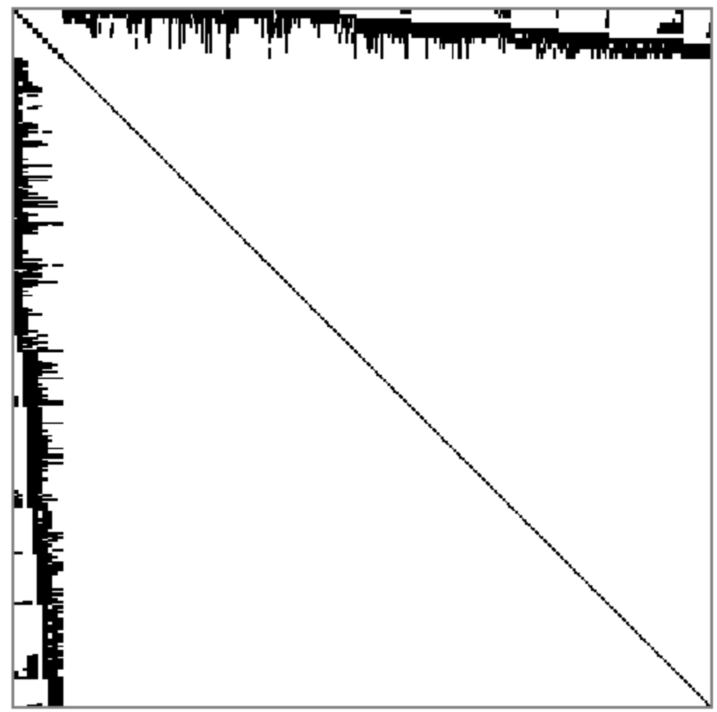
Sparse Bundle Adjustment

$$\underset{C_1,\ldots C_{N_c},X_1,\ldots,X_{N_p}}{\arg\min} \sum \left\| \pi(X_i, C_j) - x_{ij} \right\|^2$$

1 Point

1 Camera

Sparsity patten of Hessian



Manolis I. A. Lourakis, Antonis A. Argyros: SBA: A software package for generic sparse bundle adjustment. ACM Trans. Math. Softw. 36(1) (2009)

Sparse Bundle Adjustment

- An simple example
  - 4 points
  - 3 cameras
  - all points are visible in all cameras

Sparse Bundle Adjustment

$$J = \begin{pmatrix} A_{11} & 0 & 0 & B_{11} & 0 & 0 & 0 \\ 0 & A_{12} & 0 & B_{12} & 0 & 0 & 0 \\ 0 & 0 & A_{13} & B_{13} & 0 & 0 & 0 \\ A_{21} & 0 & 0 & 0 & B_{21} & 0 & 0 \\ 0 & A_{22} & 0 & 0 & B_{22} & 0 & 0 \\ 0 & 0 & A_{23} & 0 & B_{23} & 0 & 0 \\ A_{31} & 0 & 0 & 0 & 0 & B_{31} & 0 \\ 0 & A_{32} & 0 & 0 & 0 & B_{32} & 0 \\ 0 & 0 & A_{33} & 0 & 0 & B_{33} & 0 \\ A_{41} & 0 & 0 & 0 & 0 & 0 & B_{41} \\ 0 & A_{42} & 0 & 0 & 0 & 0 & B_{42} \\ 0 & 0 & A_{43} & 0 & 0 & 0 & B_{43} \end{pmatrix}, \varepsilon = \begin{pmatrix} \varepsilon_{11} \\ \varepsilon_{12} \\ \varepsilon_{13} \\ \varepsilon_{21} \\ \varepsilon_{22} \\ \varepsilon_{23} \\ \varepsilon_{31} \\ \varepsilon_{32} \\ \varepsilon_{33} \\ \varepsilon_{41} \\ \varepsilon_{42} \\ \varepsilon_{43} \end{pmatrix}$$

Sparse Bundle Adjustment

$$\boxed{J^T J}\delta_x = -J^T \varepsilon$$

$$J^T J = \begin{pmatrix} U & W \\ W^T & V \end{pmatrix} = \begin{pmatrix} U_1 & 0 & 0 & W_{11} & W_{21} & W_{31} & W_{41} \\ 0 & U_2 & 0 & W_{12} & W_{22} & W_{32} & W_{42} \\ 0 & 0 & U_3 & W_{13} & W_{23} & W_{33} & W_{43} \\ W_{11}^T & W_{12}^T & W_{13}^T & V_1 & 0 & 0 & 0 \\ W_{21}^T & W_{22}^T & W_{23}^T & 0 & V_2 & 0 & 0 \\ W_{31}^T & W_{32}^T & W_{33}^T & 0 & 0 & V_3 & 0 \\ W_{41}^T & W_{42}^T & W_{43}^T & 0 & 0 & 0 & V_4 \end{pmatrix}$$

$$U_j = \sum_{i=1}^{4} A_{ij}^T A_{ij}, V_i = \sum_{j=1}^{3} B_{ij}^T B_{ij}, W_{ij} = A_{ij}^T B_{ij}$$

Sparse Bundle Adjustment

$$J^T J \boxed{\delta_x} = -J^T \varepsilon$$

$$\delta_x = \begin{pmatrix} \delta_C \\ \delta_X \end{pmatrix} = \begin{pmatrix} \delta_{C_1}^T & \delta_{C_2}^T & \delta_{C_3}^T & \delta_{X_1}^T & \delta_{X_2}^T & \delta_{X_3}^T & \delta_{X_4}^T \end{pmatrix}^T$$

Sparse Bundle Adjustment

$$J^T J \delta_x = -\boxed{J^T \varepsilon}$$

$$J^T \varepsilon = \begin{pmatrix} \varepsilon_C \\ \varepsilon_X \end{pmatrix} = \begin{pmatrix} \varepsilon_{C_1}^T & \varepsilon_{C_2}^T & \varepsilon_{C_3}^T & \varepsilon_{X_1}^T & \varepsilon_{X_2}^T & \varepsilon_{X_3}^T & \varepsilon_{X_4}^T \end{pmatrix}^T$$

$$\varepsilon_{C_j} = \sum_{i=1}^{4} A_{ij}^T \varepsilon_{ij}$$

$$\varepsilon_{X_i} = \sum_{j=1}^{3} B_{ij}^T \varepsilon_{ij}$$

Sparse Bundle Adjustment

$$J^T J \delta_x = -J^T \varepsilon$$

$$\begin{pmatrix} U & W \\ W^T & V \end{pmatrix} \begin{pmatrix} \delta_C \\ \delta_X \end{pmatrix} = -\begin{pmatrix} \varepsilon_C \\ \varepsilon_X \end{pmatrix}$$

$$\begin{pmatrix} U - WV^{-1}W^T & 0 \\ W^T & V \end{pmatrix} \begin{pmatrix} \delta_C \\ \delta_X \end{pmatrix} = -\begin{pmatrix} \varepsilon_C - WV^{-1}\varepsilon_X \\ \varepsilon_X \end{pmatrix}$$

$$S = U - WV^{-1}W^T$$ Schur Complement

$$S\delta_C = -(\varepsilon_C - WV^{-1}\varepsilon_X)$$ Compute cameras first (# cameras << # points)

$$V\delta_X = -\varepsilon_X - W^T \delta_C$$ back substitution for points

Sparse Bundle Adjustment

- In general, NOT all points are visible in all cameras

$$U_j = \sum_{i=1}^{4} A_{ij}^T A_{ij}, V_i = \sum_{j=1}^{3} B_{ij}^T B_{ij}, W_{ij} = A_{ij}^T B_{ij}$$

  - $A_{ij} = B_{ij} = 0$ if i-th points is invisible (or not matched) in j-th camera
  - More sparse structure, more speed-up

Related Works

- Hierarchical BA
  - Steedly et al. 2003, Snavely et al. 2008, Frahm et al. 2010
- Segment-based BA
  - Zhu et al. 2014, Zhang et al. 2016 (ENFT)
- Incremental BA
  - Kaess et al. 2008 (iSAM), Kaess et al. 2011 (iSAM2), Indelman et al. 2012 (iLBA), Ila et al. 2017 (SLAM++), Liu et al. 2017 (EIBA)
- Parallel BA
  - Ni et al. 2007, Wu et al. 2011 (PBA)

Segment-based Bundle Adjustment

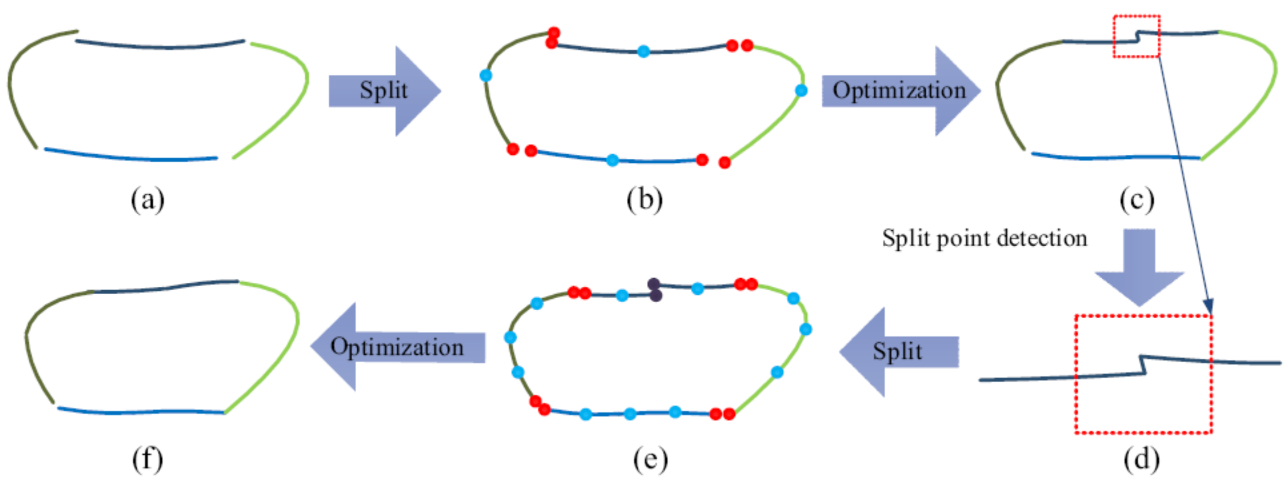
Zhang G, Liu H, Dong Z, et al. Efficient non-consecutive feature tracking for robust structure-from-motion[J]. IEEE Transactions on Image Processing, 2016, 25(12): 5957-5970.

The Difficulties for Large-Scale SfM

- Global Bundle Adjustment
  - Huge variables
  - Memory limit
  - Time-consuming
- Iterative Local Bundle Adjustment
  - Large error is difficult to be propagated to the whole sequence.
  - Easily stuck in a local optimum.
- Pose Graph Optimization
  - May not sufficiently minimize the error.

Segment-based Progressive SfM

- Split a long sequence to multiple short sequences.
- Perform SfM for each sequence and align them together.
- Detect the ``split point'' and further split the sequence if the reprojection error is large.
- The above procedure is repeated until the error is less than a threshold.



Segment-based Progressive SfM

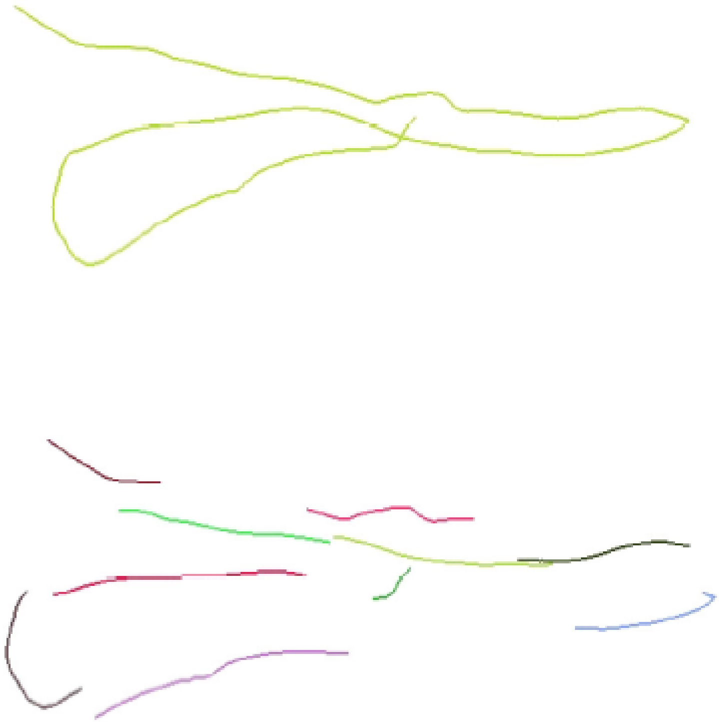
- Split Point Detection
  - Best minimize the reprojection error w.r.t. a, i.e. steepest descent direction

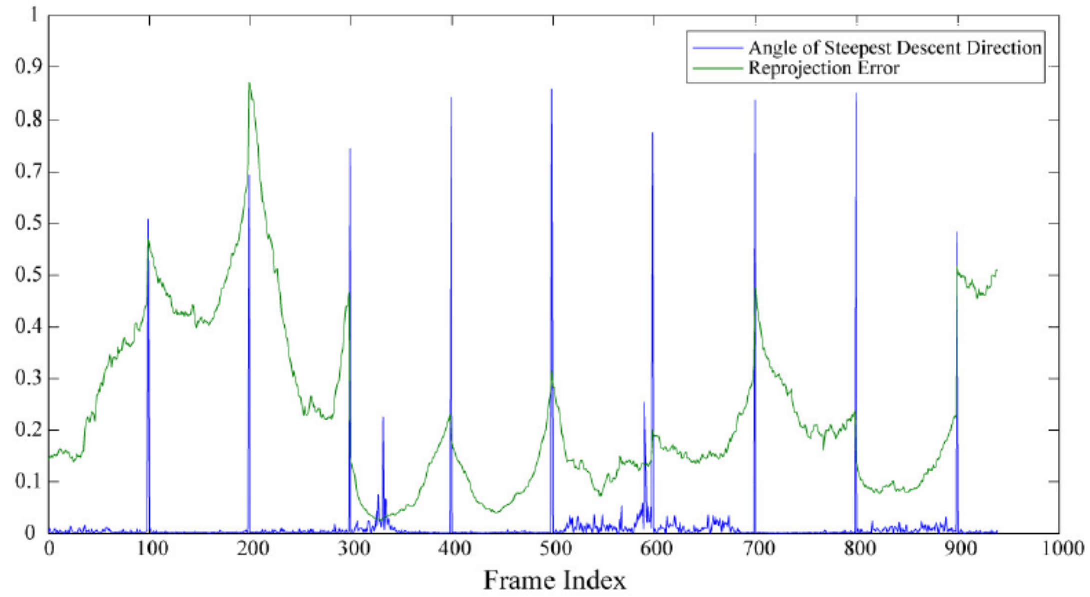
$$g_k = \sum_{i=1\cdots N_k} A_i^T e_i \qquad \begin{aligned} A_i &= \partial\pi(P_k X_i)/\partial a_k \\ e_i &= \mathbf{x}_i - \pi(P_k X_i) \end{aligned}$$

  - The inconsistency between two consecutive frames

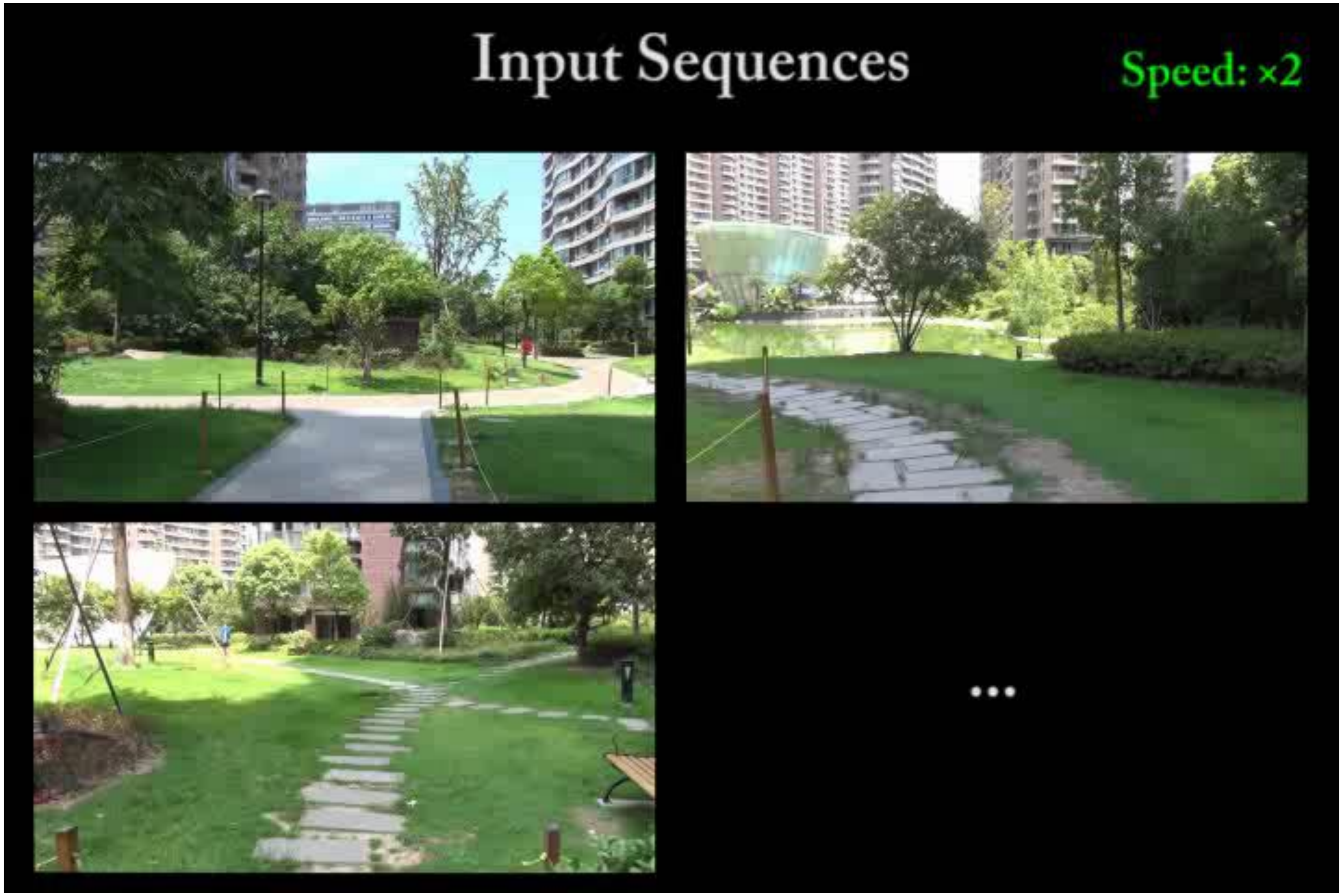
$$C(k, k+1) = \arccos \frac{g_k^T \cdot g_{k+1}}{||g_k|| \cdot ||g_{k+1}||}.$$

Split Point Detection





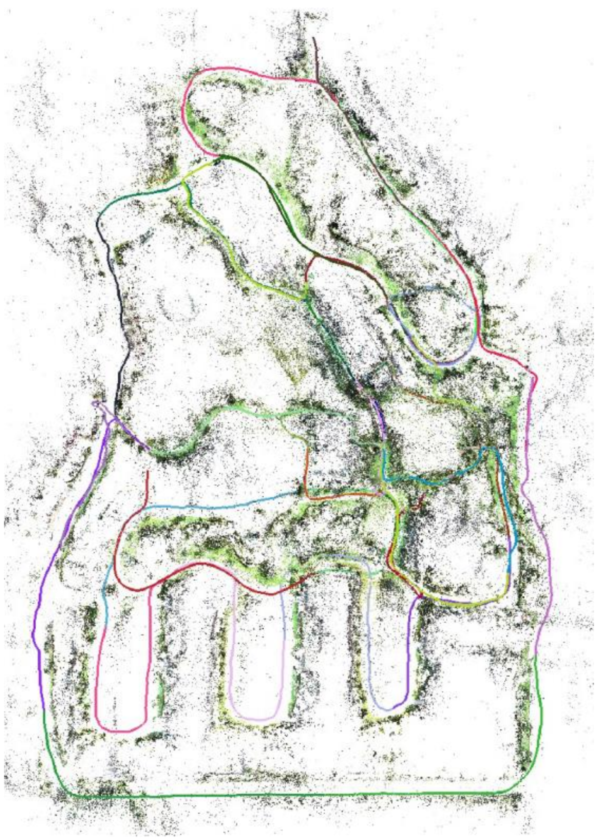
SFM on Garden Dataset



6段长视频序列，将近10万帧，特征匹配74分钟，SfM求解16分钟（单线程），平均17.7fps

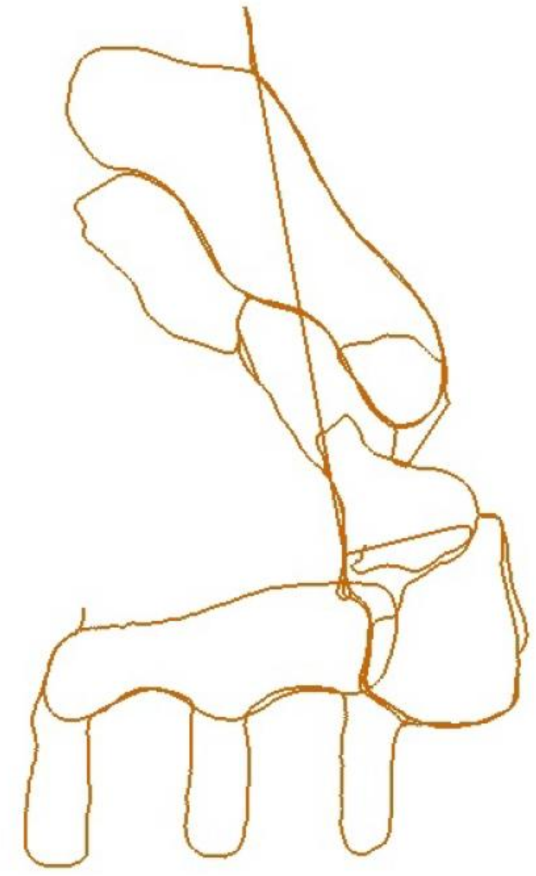
VisualSFM：SfM求解 57 分钟 （GPU加速）

Comparison on Garden Dataset



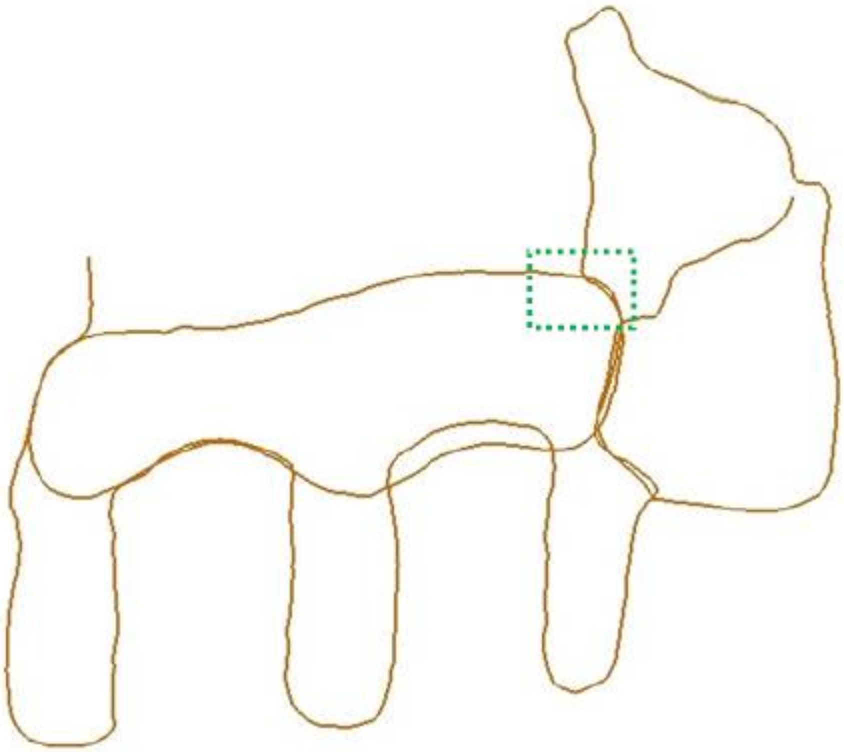
ENFT-SFM

VisualSFM



ORB-SLAM

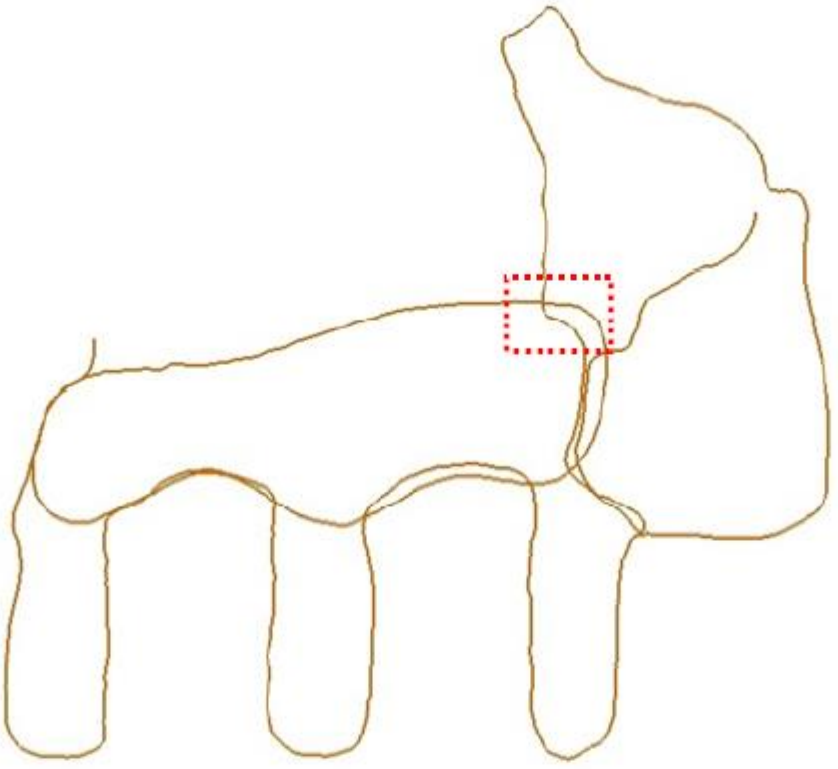
Comparison with ORB-SLAM in Garden 01 Sequence



ENFT-SLAM

Non-consecutive Track Matching

Segment-based BA



ORB-SLAM

Bag-of-words Place Recognition

Pose Graph Optimization + Traditional BA

Incremental BA in iSAM2 Based on Bayes Tree

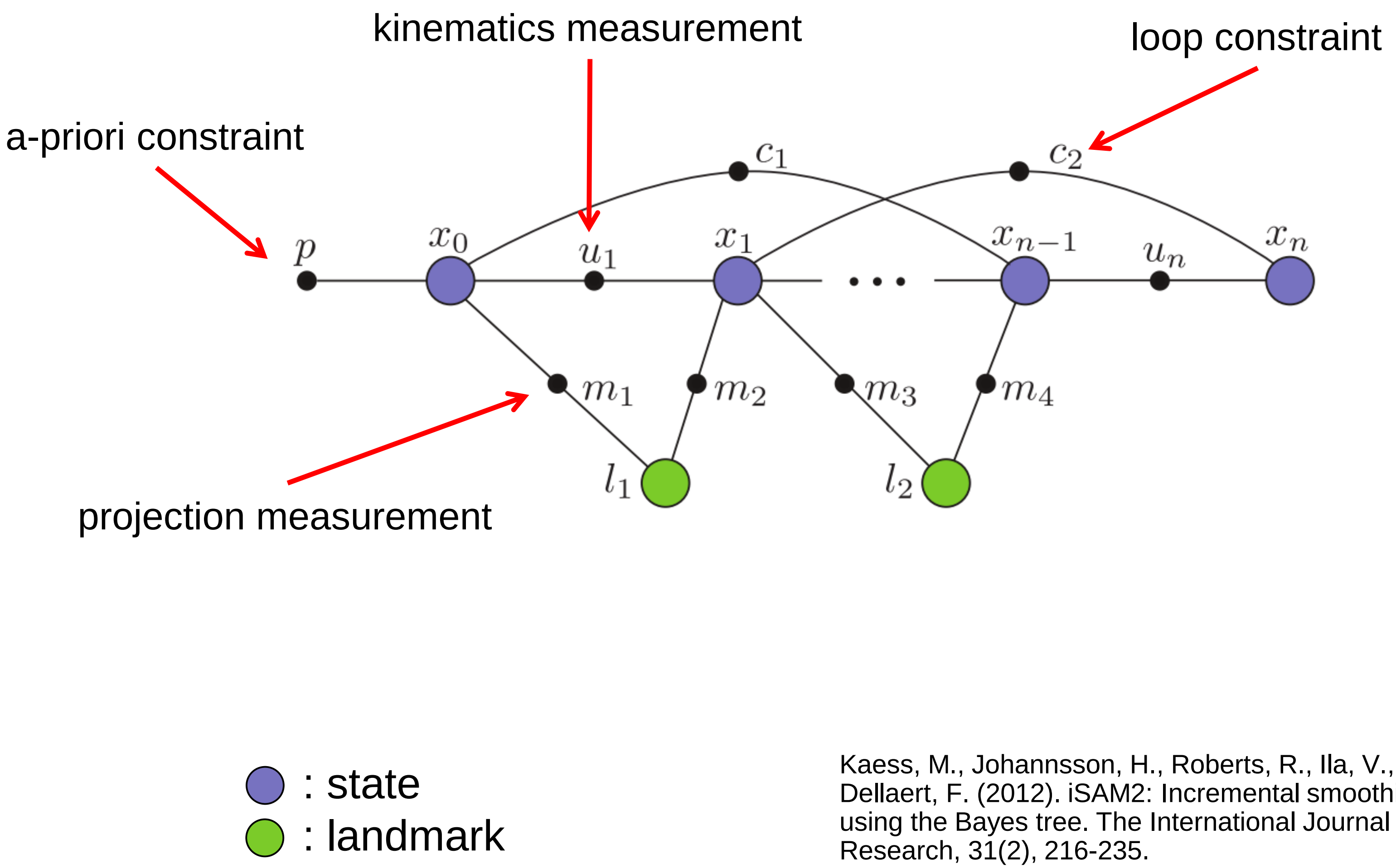
Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J. J., & Dellaert, F. (2012). iSAM2: Incremental smoothing and mapping using the Bayes tree. The International Journal of Robotics Research, 31(2), 216-235.

Incremental Bundle Adjustment

In order to benefit from increased accuracy offered by relinearization in batch optimization:

- Fixed-lag / Sliding-window Approaches
- Keyframe-based Approaches
- Incremental Approaches (iSAM, iSAM2, our EIBA)

Gaussian Factor Graph



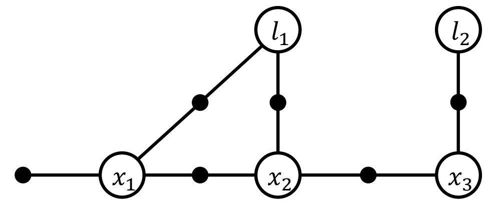
Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J. J., & Dellaert, F. (2012). iSAM2: Incremental smoothing and mapping using the Bayes tree. The International Journal of Robotics Research, 31(2), 216-235.

Main Ideas of iSAM2

- **Reduce fill-in:** Use heuristics algorithms CCOLAMD to provide a suboptimal ordering for factorization (finding the optimal is NP-hard).
- **Encode with the Bayes tree:** Introduce Bayes tree (a.k.a. directed clique tree) to encode the square root information matrix.
- **Fluid relinearization:** Perform fluid relinearization when adding new factors or updating the linearization points to avoid batch optimization.
- **Partial state updates:** Perform partial state updates when solving the Bayes in order to update a state variable only when neccesary.

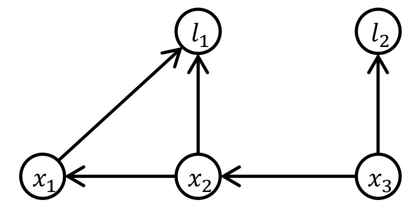
One step: linearization

factor graph



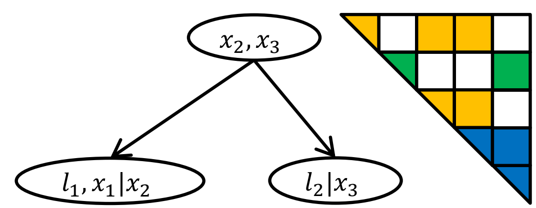
eliminating the factor graph using the CCOLAMD ordering (e.g.l_1, l_2, x_1, x_2, x_3)

chordal Bayes net

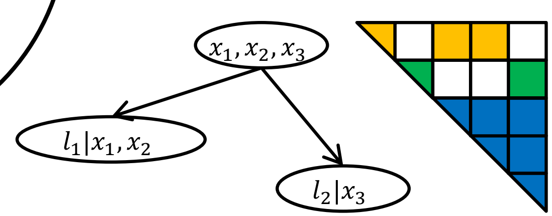


creating Bayes tree in reverse elimination order (e.g.x_3, x_2, x_1, l_2, l_1)

Bayes tree



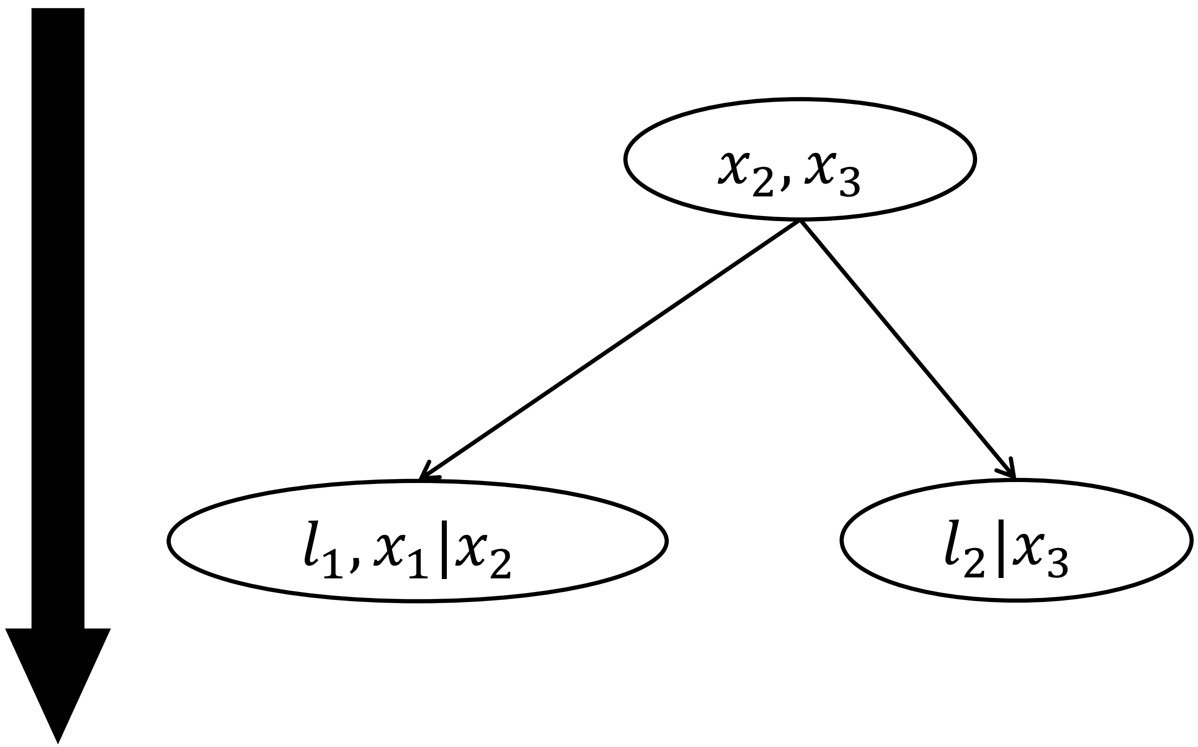
adding new factors/states and applying the fluid relinearization (e.g. $f(x_1, x_3)$)



One step: partial update

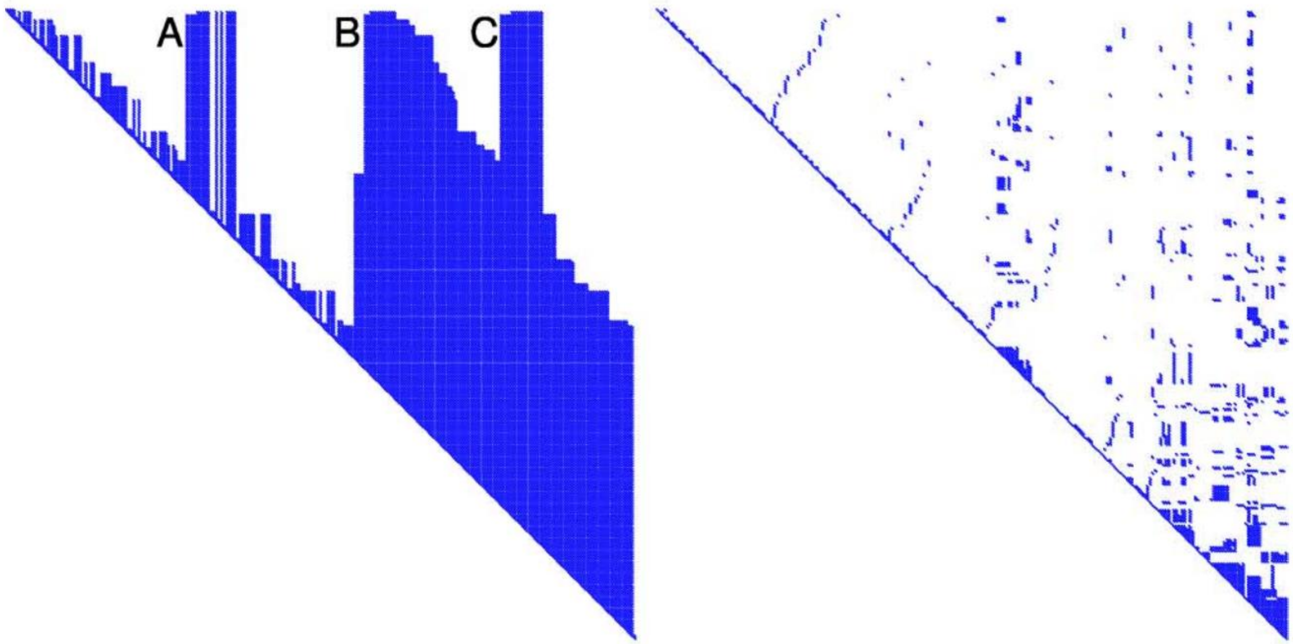
starting from the root clique

updating all variables that change by more than a threshold



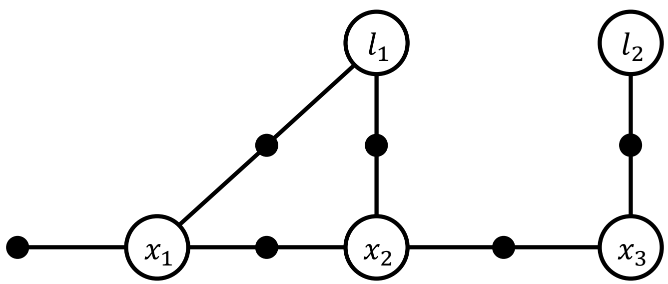
Reduce Fill-in

Reordering with CCOLAMD / CHOLMOD



Kaess, M., Ranganathan, A., & Dellaert, F. (2008). iSAM: Incremental smoothing and mapping. IEEE Transactions on Robotics, 24(6), 1365-1378.

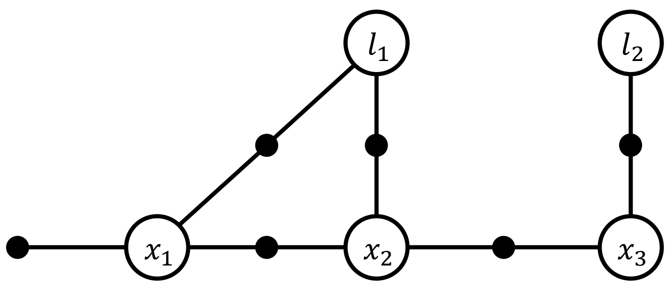
In Gaussian factor graphs, elimination is equivalent to sparse QR factorization of the measurement Jacobian.



$$
\begin{array}{cc}
 & \begin{array}{ccccc} l_1 & l_2 & x_1 & x_2 & x_3 \end{array} \\
J = & \begin{bmatrix}
\times & & \times & & \\
\times & & & \times & \\
 & \times & & & \times \\
 & & \times & & \\
 & & \times & \times & \\
 & & & \times & \times
\end{bmatrix}
\end{array}
$$

sparse pattern of the measurement Jacobian

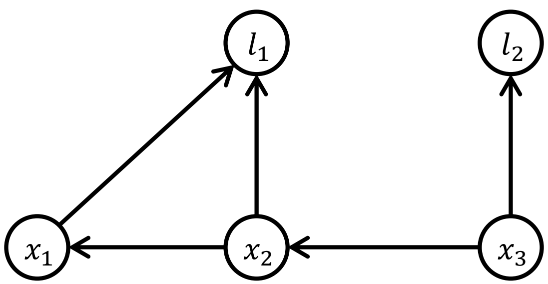
In Gaussian factor graphs, elimination is equivalent to sparse QR factorization of the measurement Jacobian.



$$
\begin{array}{cc}
 & \begin{array}{ccccc} l_1 & l_2 & x_1 & x_2 & x_3 \end{array} \\
H = & \begin{bmatrix}
\times & & \times & \times & \\
 & \times & & & \times \\
\times & & \times & \times & \\
\times & & \times & \times & \times \\
 & \times & & \times & \times
\end{bmatrix}
\end{array}
$$

sparse pattern of the information matrix

In Gaussian factor graphs, elimination is equivalent to sparse QR factorization of the measurement Jacobian.



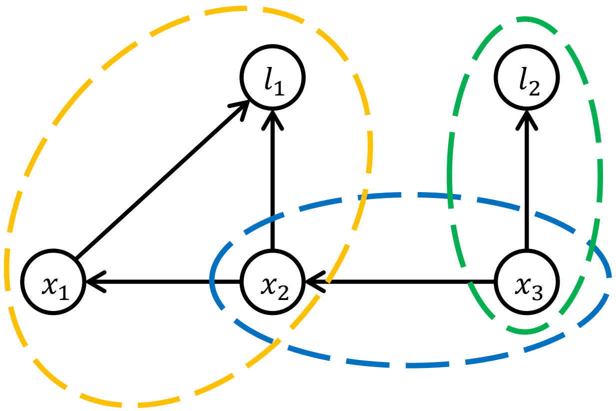
$$R = \begin{array}{c} \begin{array}{ccccc} l_1 & l_2 & x_1 & x_2 & x_3 \end{array} \\ \begin{bmatrix} \times & & \times & \times & \\ & \times & & & \times \\ & & \times & \times & \\ & & & \times & \times \\ & & & & \times \end{bmatrix} \end{array}$$

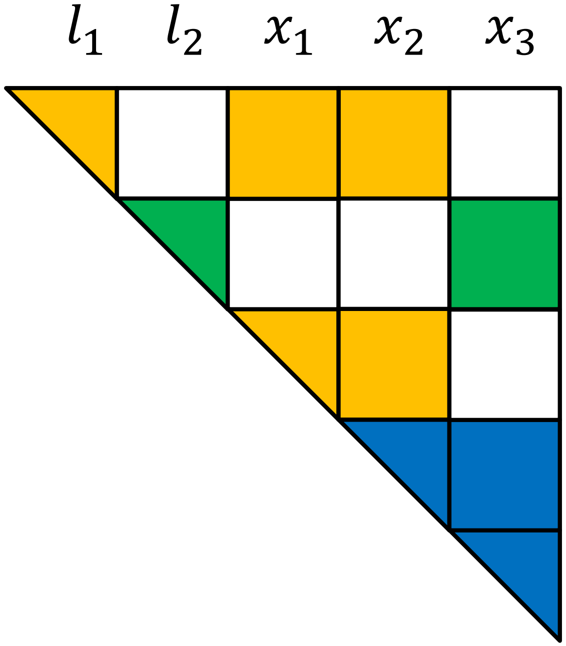
No fill-in if we eliminate the factor graph using the elimination ordering l_1, l_2, x_1, x_2, x_3.

The resulting directed graph is called the **chordal** Bayes net.

sparse pattern of the square root information matrix

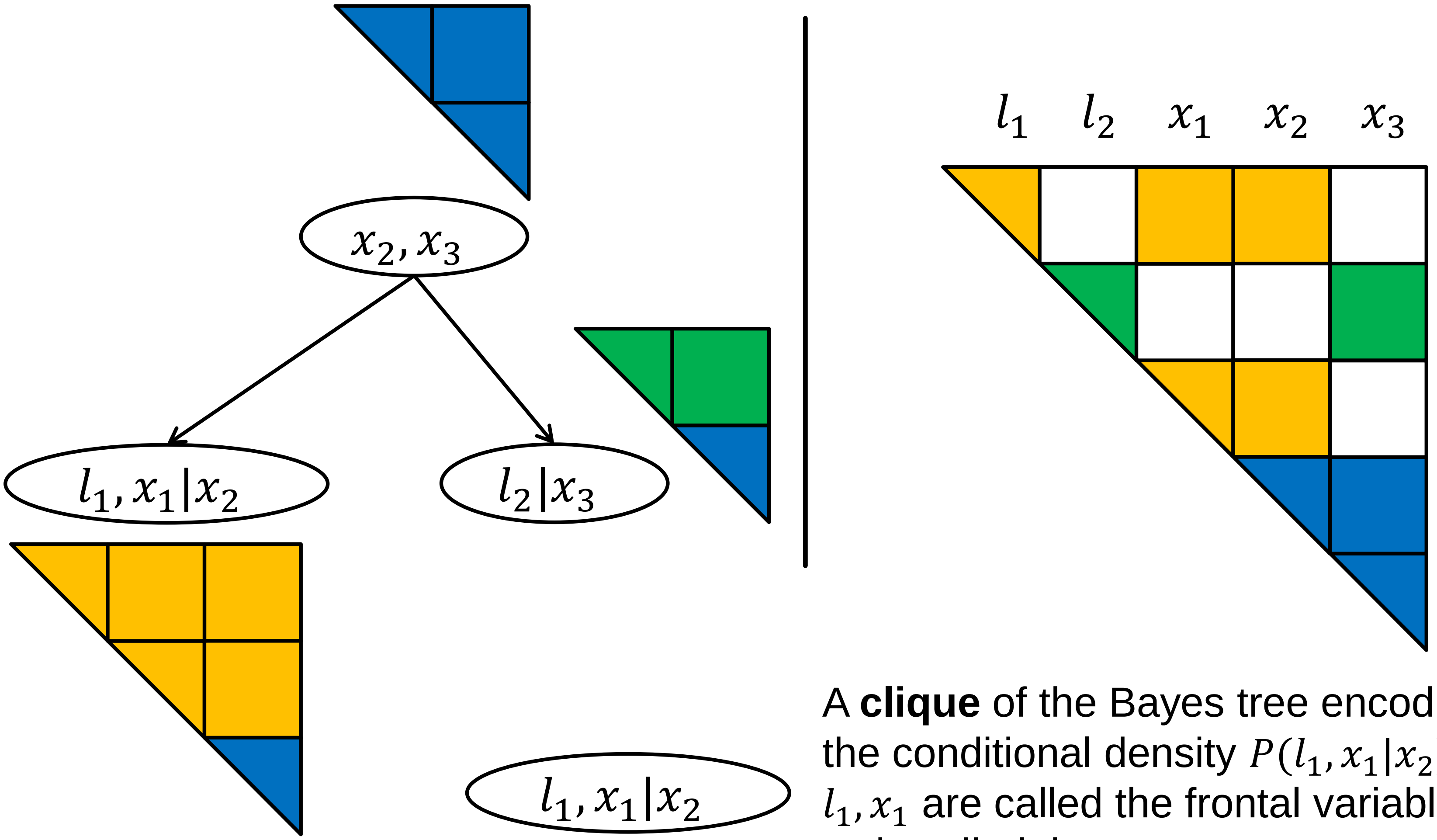
Encode with the Bayes Tree





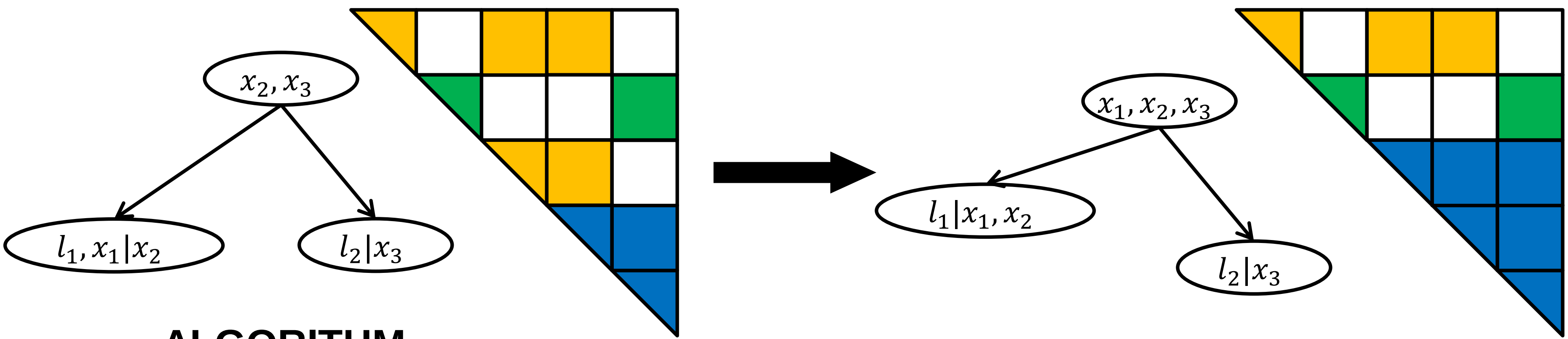
For each conditional density $P(\theta_i|S_i)$ of the Bayes net, in reverse elimination order (i.e. x_3, x_2, x_1, l_2, l_1), we create a Bayes tree.

Encode with the Bayes Tree



A **clique** of the Bayes tree encoding the conditional density $P(l_1, x_1 | x_2)$
l_1, x_1 are called the frontal variables
x_2 is called the separator

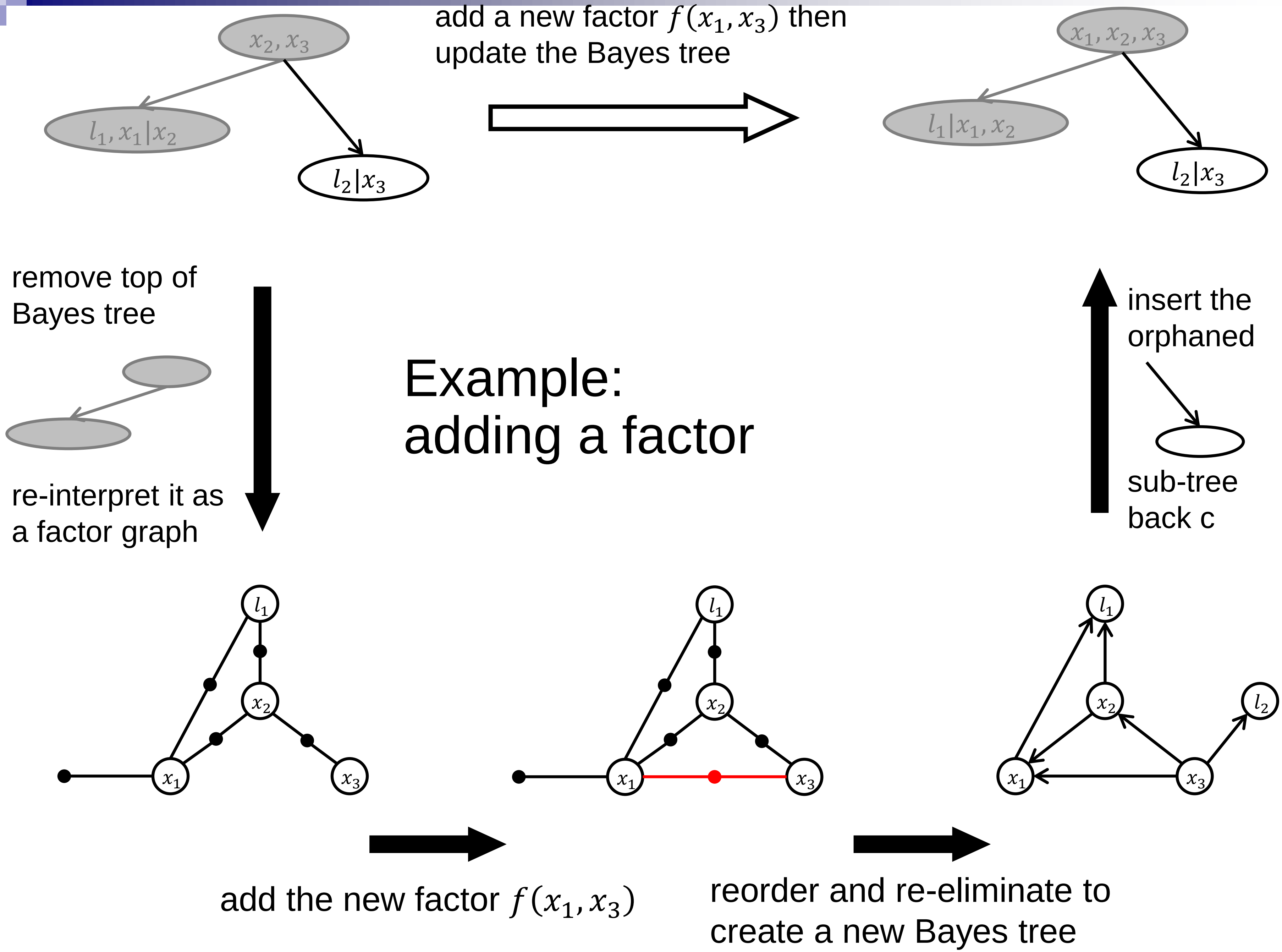
Adding New Factors



ALGORITHM

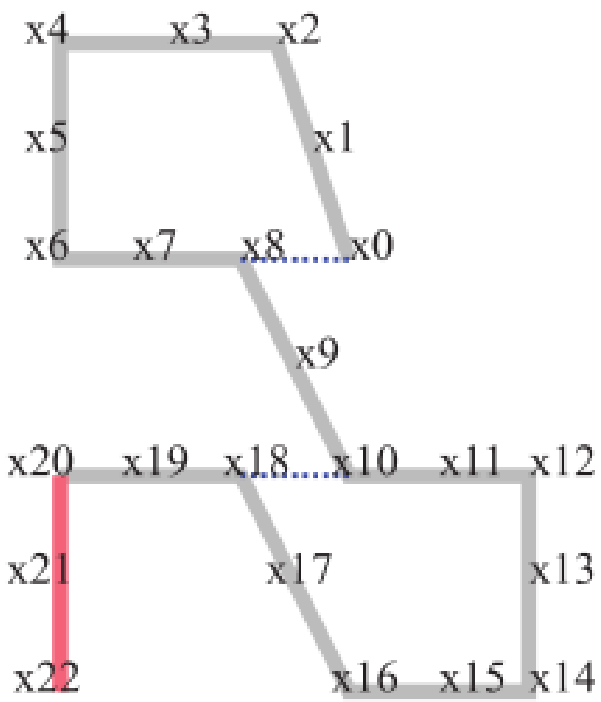
Fluid relinearization when adding new factors.

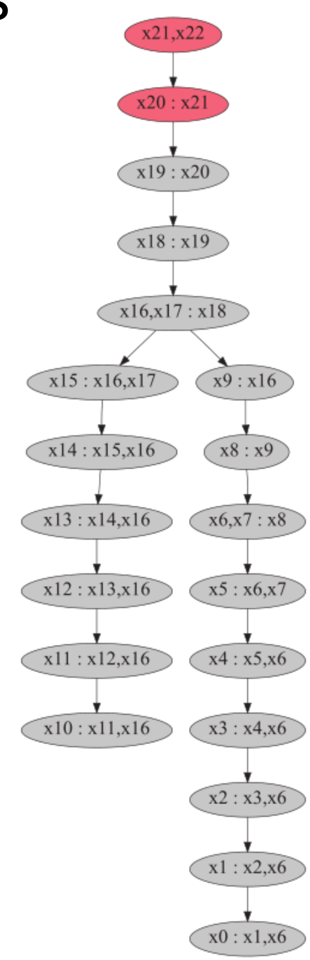
- For each variable affected by new factors, remove the corresponding clique and all parents up to the root
- Re-interpret the removed part as a factor graph
- Add the new factors into the resulting factor graph.
- Re-order variables and eliminate the factor graph to recreate a top Bayes tree.
- Insert the orphaned sub-trees back into the new Bayes tree.


add a new factor $f(x_1, x_3)$ then
update the Bayes tree
x_2, x_3
$l_1, x_1 | x_2$
$l_2 | x_3$
x_1, x_2, x_3
$l_1 | x_1, x_2$
$l_2 | x_3$
remove top of Bayes tree
Example: adding a factor
insert the orphaned sub-tree back c
re-interpret it as a factor graph
l_1
x_2
x_1
x_3
l_1
x_2
x_1
x_3
l_1
x_2
l_2
x_1
x_3
add the new factor $f(x_1, x_3)$
reorder and re-eliminate to create a new Bayes tree

Example of adding new states and factors

Information only propagates upwards.

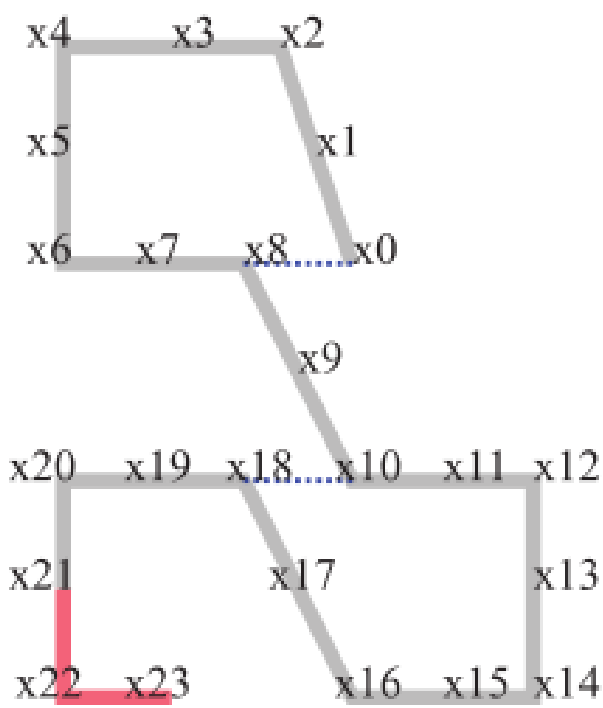


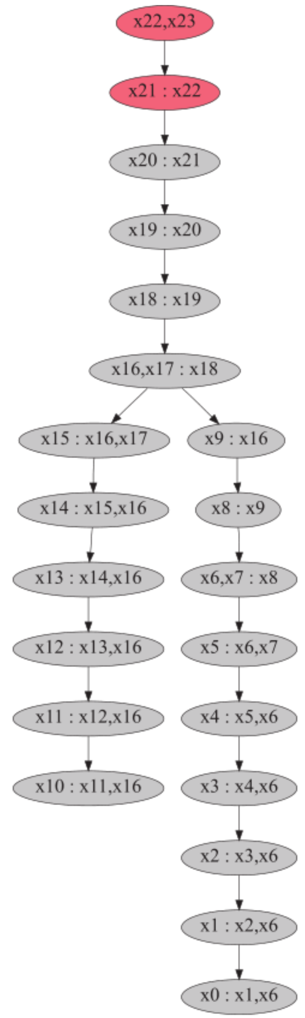


Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J. J., & Dellaert, F. (2012). iSAM2: Incremental smoothing and mapping using the Bayes tree. The International Journal of Robotics Research, 31(2), 216-235.

Example of adding new states and factors

Information only propagates upwards.

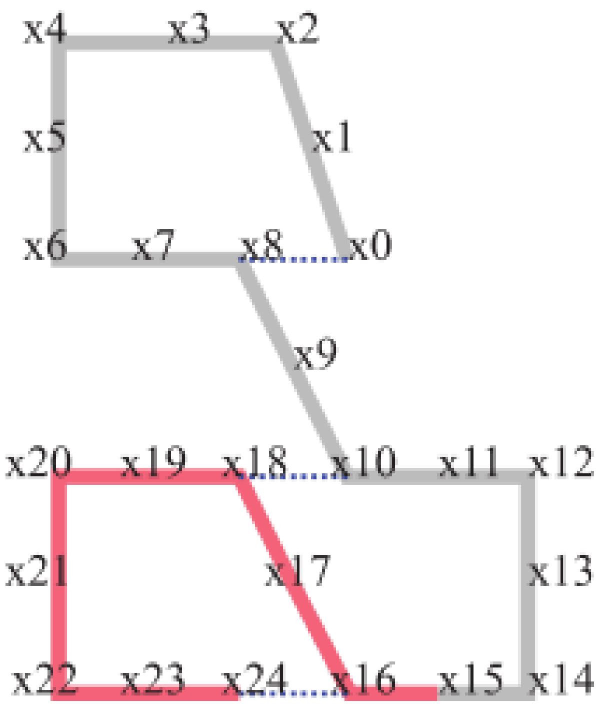


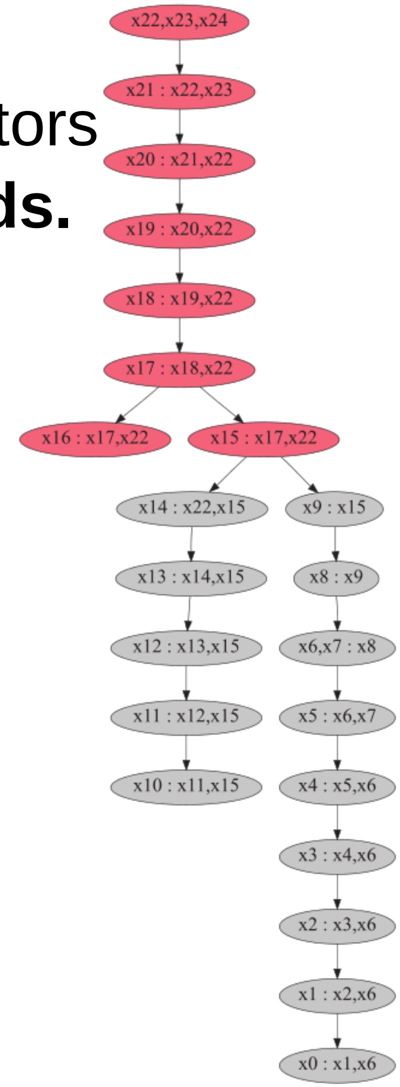


Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J. J., & Dellaert, F. (2012). iSAM2: Incremental smoothing and mapping using the Bayes tree. The International Journal of Robotics Research, 31(2), 216-235.

Example of adding new states and factors

Information only propagates upwards.





Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J. J., & Dellaert, F. (2012). iSAM2: Incremental smoothing and mapping using the Bayes tree. The International Journal of Robotics Research, 31(2), 216-235.

Constrained COLAMD

While adding new states (always along with adding new factors), information only propagates upwards.

1. Force the most recently accessed variables to the end and still provide a good overall ordering.
2. Subsequent updates will then only affect a small part of the tree (the top of the Bayes tree).
3. Efficient in most cases, except for large loop closures.

Fluid Relinearization

ALGORITHM

Fluid relinearization when linearization points change (together with adding new factors).

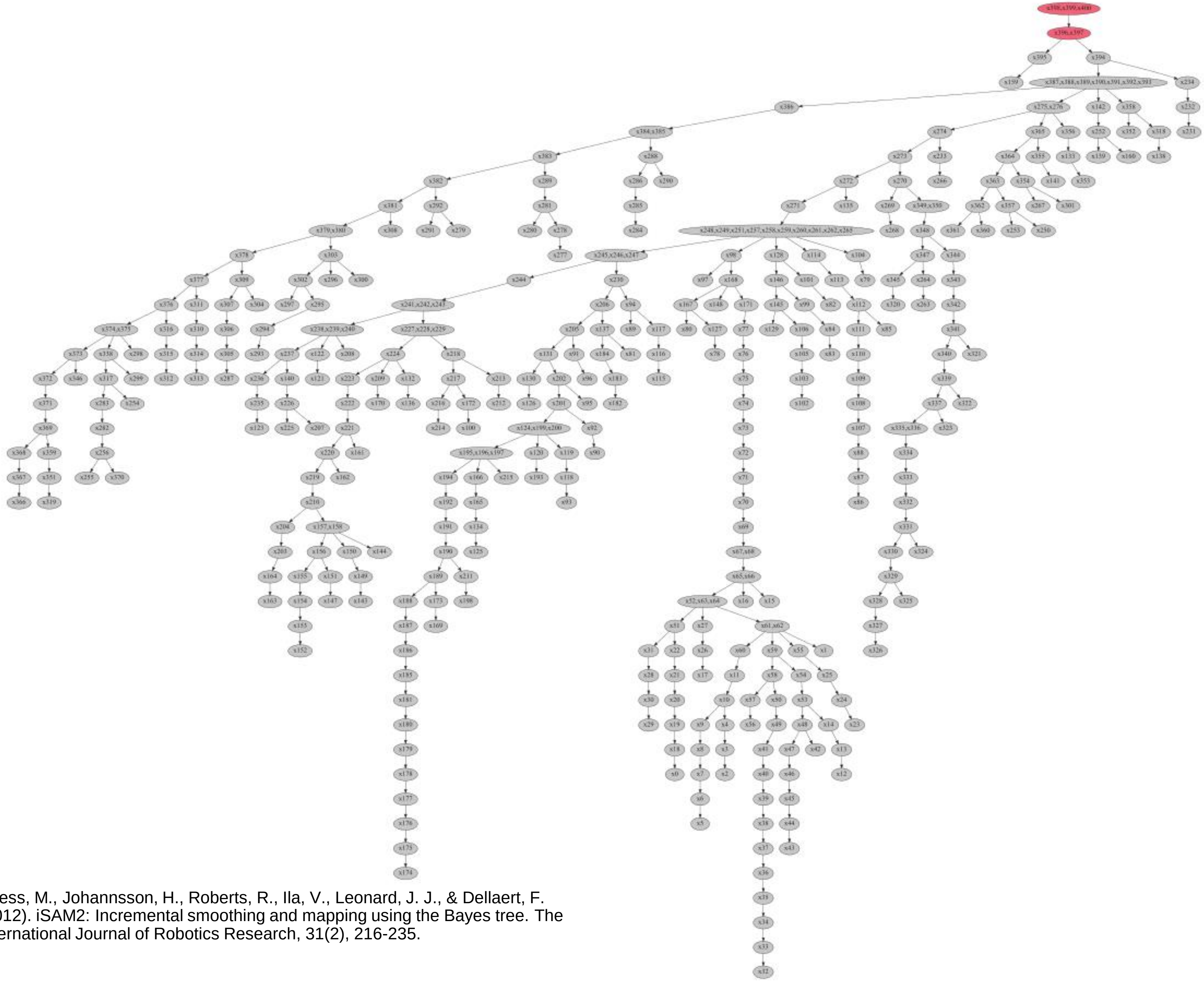
1. For each affected variable remove the corresponding clique and all parents up to the root.
2. Relinearize all factors required to recreate top.
3. *<u>Add cached linear factors from orphans.</u>*
4. Re-order variables and eliminate the factor graph to create a new top Bayes tree.
5. Insert the orphaned sub-trees back into the new Bayes tree.

Partial State Updates

ALGORITHM

Starting from the root clique:

1. For current clique:
   compute update of frontal variables from the local conditional density.
2. For all variables that change by more than a threshold:
   recursively process each descendant containing such a variable.



Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J. J., & Dellaert, F. (2012). iSAM2: Incremental smoothing and mapping using the Bayes tree. The International Journal of Robotics Research, 31(2), 216-235.

Efficient Incremental BA

Liu H, Li C, Chen G, et al. Robust Keyframe-based Dense SLAM with an RGB-D Camera[J]. arXiv preprint arXiv:1711.05166, 2017.

Revisit Standard BA

- A regular BA function

$$\sum_j \sum_{i \in \mathcal{V}_j} \left(\left\| \frac{\boldsymbol{\pi}(\mathbf{K}(\mathbf{C}_i \mathbf{X}_j)) - \mathbf{x}_{ji}}{\sigma_{\mathbf{x}}} \right\|_\delta + \left\| \frac{z^{-1}(\mathbf{C}_i \mathbf{X}_j) - z_{ji}^{-1}}{\sigma_z} \right\|_\delta \right)$$

Reprojection error　　Inverse depth error

$\mathcal{V}_j$ is the set of cameras in which point j is visible.

- Convert Huber norm by re-weighting scheme

$$f = \sum_j \sum_{i \in \mathcal{V}_j} \|\mathbf{f}_{ij}(\mathbf{C}_i, \mathbf{X}_j)\|_2^2,$$

- Linearization

$$\mathbf{f}_{ij}(\mathbf{C}_i, \mathbf{X}_j) \approx \mathbf{J}_{\mathbf{C}_{ij}} \delta_{\mathbf{C}_i} + \mathbf{J}_{\mathbf{X}_{ij}} \delta_{\mathbf{X}_j} - \mathbf{e}_{ij} \qquad f \approx \|\mathbf{J}\boldsymbol{\delta} - \mathbf{e}\|_2^2$$

$\mathbf{J}$ is $3n_x \times (6n_c + 3n_p)$
Jacobian matrix

- Solving normal equation $\mathbf{J}^\top \mathbf{J} \boldsymbol{\delta} = \mathbf{J}^\top \mathbf{e}$

Revisit Standard BA

- Step 1: Construct normal equation
  - Compute and store the small non-zero block matrices $\mathbf{U}_{ii}$, $\mathbf{V}_{jj}$, $\mathbf{W}_{ij}$
  - Do not need to reconstruct $\mathbf{J}^\top\mathbf{J}$ from scratch.
  - Only need to add new block matrices.

$$\mathbf{J}^\top\mathbf{J}\delta = \mathbf{J}^\top\mathbf{e}$$

$$\Downarrow$$

$$\begin{bmatrix} \mathbf{U} & \mathbf{W} \\ \mathbf{W}^\top & \mathbf{V} \end{bmatrix} \begin{bmatrix} \delta_\mathbf{C} \\ \delta_\mathbf{X} \end{bmatrix} = \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix}$$

$\mathbf{U}$: $n_c \times n_c$
$\mathbf{V}$: $n_p \times n_p$
$\mathbf{W}$: $n_c \times n_p$
$\mathbf{W}_{ij}$ no zero only if point j is visible in camera i

Revisit Standard BA

- Step 2: Marginalize points to construct Schur Complement
  - $\boldsymbol{S}$ is also sparse, with non-zero block matrix $\mathbf{S}_{i_1 i_2}$ if and only if camera i_1 and i_2 share common points.

$$\mathbf{S}\delta_{\mathbf{C}} = \mathbf{g},$$

$$\mathbf{S} = (\mathbf{U} - \mathbf{W}\mathbf{V}^{-1}\mathbf{W}^{\top}),$$

$$\mathbf{g} = \mathbf{u} - \mathbf{W}\mathbf{V}^{-1}\mathbf{v}.$$

Revisit Standard BA

- Step 3: Update cameras
  - Use preconditioned conjugate gradient (PCG) to solve for $\delta_{\mathbf{C}}$
    - PCG naturally leverages the sparseness of $\boldsymbol{S}$
  - $\mathbf{C}_i = \exp(\delta_{\mathbf{C}_i})\mathbf{C}_i$
- Step 4: Update points
  - Back substitution

$$\delta_{\mathbf{X}_j} = \mathbf{V}_{jj}^{-1}\left(\mathbf{v}_j - \sum_{i \in \mathcal{V}_j} \mathbf{W}_{ij}^{\top}\delta_{\mathbf{C}_i}\right) \qquad \mathbf{X}_j += \delta_{\mathbf{X}_j}$$

Revisit Standard BA

- Num. of observations in each keyframe much larger than Num. of cameras
  - Computation :

    Step 1, 2 $\gg$ Step 3
  - Construction of normal equation and Schur complement takes much more time than PCG iterations
- most variables nearly unchanged (incremental reconstruction)
  - Most computation in steps 1, 2, 4 are unnecessary
  - Contribution of most $\mathbf{f}_{ij}$s to normal equation nearly remains the same

Efficient Incremental BA (EIBA)

- Local BA vs. Global B
  - local BA : suboptimal, especially when the local map contains large error.
  - global BA : accurate but slow, high latency, lots of unnecessary computation.
- Incremental BA
  - Makes maximum use of intermediate computation for efficiency
  - Adaptively updating affected keyframes for map refinement

One iteration in EIBA

- Step 1 : Update normal equations and Schur complement from the last iteration
  - Store the effect of $\mathbf{f}_{ij}$ in $\mathbf{A}_{ij}^{\mathbf{U}}$, $\mathbf{A}_{ij}^{\mathbf{V}}$, $\mathbf{b}_{ij}^{\mathbf{u}}$ and $\mathbf{b}_{ij}^{\mathbf{v}}$, initialize to 0 at first, only re-computed when linearization point of $\mathbf{f}_{ij}$ is changed.
  - Remove contribution from the last iteration, refresh them, update for current iteration.
  - Update from $\mathbf{A}_{ij}^{\mathbf{U}}$, $\mathbf{A}_{ij}^{\mathbf{V}}$, $\mathbf{b}_{ij}^{\mathbf{u}}$ and $\mathbf{b}_{ij}^{\mathbf{v}}$

One iteration in EIBA

- Step 1 : Update normal equations and Schur complement from the last iteration

for each point j and each camera $i \in \mathcal{V}_j$ that $\mathbf{C}_i$ or $\mathbf{X}_j$ is changed **do**

Construct linearized equation

$\mathbf{S}_{ii} -= \mathbf{A}_{ij}^{\mathbf{U}}$; $\mathbf{A}_{ij}^{\mathbf{U}} = \mathbf{J}_{\mathbf{C}_{ij}}^{\top} \mathbf{J}_{\mathbf{C}_{ij}}$; $\mathbf{S}_{ii} += \mathbf{A}_{ij}^{\mathbf{U}}$

$\mathbf{V}_{jj} -= \mathbf{A}_{ij}^{\mathbf{V}}$; $\mathbf{A}_{ij}^{\mathbf{V}} = \mathbf{J}_{\mathbf{X}_{ij}}^{\top} \mathbf{J}_{\mathbf{X}_{ij}}$; $\mathbf{V}_{jj} += \mathbf{A}_{ij}^{\mathbf{V}}$

$\mathbf{g}_i -= \mathbf{b}_{ij}^{\mathbf{u}}$; $\mathbf{b}_{ij}^{\mathbf{u}} = \mathbf{J}_{\mathbf{C}_{ij}}^{\top} \mathbf{e}_{ij}$; $\mathbf{g}_i += \mathbf{b}_{ij}^{\mathbf{u}}$

$\mathbf{v}_j -= \mathbf{b}_{ij}^{\mathbf{v}}$; $\mathbf{b}_{ij}^{\mathbf{v}} = \mathbf{J}_{\mathbf{X}_{ij}}^{\top} \mathbf{e}_{ij}$; $\mathbf{v}_j += \mathbf{b}_{ij}^{\mathbf{v}}$

$\mathbf{W}_{ij} = \mathbf{J}_{\mathbf{C}_{ij}}^{\top} \mathbf{J}_{\mathbf{X}_{ij}}$

Mark $\mathbf{V}_{jj}$ updated

end for

One iteration in EIBA

- Step 2 : Update point marginalization and Schur complement from last iteration

for each point j that $\mathbf{V}_{jj}$ is updated and each camera pair $(i_1, i_2) \in \mathcal{V}_j \times \mathcal{V}_j$ **do**

$$\mathbf{S}_{i_1 i_2} += \mathbf{A}^{\mathbf{S}}_{i_1 i_2 j}$$
$$\mathbf{A}^{\mathbf{S}}_{i_1 i_2 j} = \mathbf{W}_{i_1 j}\mathbf{V}^{-1}_{jj}\mathbf{W}^{\top}_{i_2 j}$$
$$\mathbf{S}_{i_1 i_2} -= \mathbf{A}^{\mathbf{S}}_{i_1 i_2 j}$$

end for

for each point j that $\mathbf{V}_{jj}$ is updated and each camera $i \in \mathcal{V}_j$ **do**

$$\mathbf{g}_i += \mathbf{b}^{\mathbf{g}}_{ij};\ \mathbf{b}^{\mathbf{g}}_{ij} = \mathbf{W}_{ij}\mathbf{V}^{-1}_{jj}\mathbf{v}_j;\ \mathbf{g}_i -= \mathbf{b}^{\mathbf{g}}_{ij}$$

end for

One iteration in EIBA

- Step 3 : Update cameras
  - Solve $\delta_{\mathbf{C}}$ by PCG
  - Change $\mathbf{C}_i$ only if $||\delta_{\mathbf{C}_i}||$ exceeds a threshold ϵ_c

- Step 4 : Update points
  - Back substitution only for visible points in the changed cameras
  - Change $\mathbf{X}_j$ only if $||\delta_{\mathbf{X}_j}||$ exceeds a threshold ϵ_p

EIBA in RKD-SLAM

- Energy function

Reprojection error   Inverse depth error

$$\sum_{j}\sum_{i\in\mathcal{V}_j}\left(\|\frac{\pi(\mathbf{K}(\mathbf{C}_i\mathbf{X}_j))-\mathbf{x}_{ji}}{\sigma_{\mathbf{x}}}\|_\delta+\|\frac{z^{-1}(\mathbf{C}_i\mathbf{X}_j)-z_{ji}^{-1}}{\sigma_z}\|_\delta\right)$$
$$+\sum_{(i_1,i_2)\in\mathcal{L}}\|\log(\mathbf{C}_{i_1}\circ\mathbf{C}_{i_2}\circ\mathbf{T}_{i_1i_2}^{-1})\|_{\Sigma_{i_1i_2}}^2,$$

Loop constraint

  - Consist of 3D points observation term and loop constraint term

EIBA in RKD-SLAM

- 3D point observation term

$$\sum_{j}\sum_{i\in\mathcal{V}_j}\left(\|\frac{\boldsymbol{\pi}(\mathbf{K}(\mathbf{C}_i\mathbf{X}_j))-\mathbf{x}_{ji}}{\sigma_{\mathbf{x}}}\|_\delta+\|\frac{z^{-1}(\mathbf{C}_i\mathbf{X}_j)-z_{ji}^{-1}}{\sigma_z}\|_\delta\right)$$

  - Use inverse depth parameterize $\mathbf{X}_j$
    - $\mathbf{X}_j=\mathbf{C}_k^{-1}(z_{jk}\mathbf{K}^{-1}\hat{\mathbf{x}}_{jk})$
    - Each re-projection equation $\mathbf{f}_{ij}$ relates two camera poses $\mathbf{C}_i$ and $\mathbf{C}_k$,one 3D point $\mathbf{X}_j$
    - Linearization

$$\mathbf{f}_{ij}(\mathbf{C}_i,\mathbf{C}_k,\mathbf{X}_j)\approx\mathbf{J}_{\mathbf{C}_{ij}}\delta_{\mathbf{C}_i}+\mathbf{J}_{\mathbf{C}_{kj}}\delta_{\mathbf{C}_k}+\mathbf{J}_{\mathbf{X}_{ij}}\delta_{\mathbf{X}_j}-\mathbf{e}_{ij},$$

    - Also need to update $\mathbf{S}_{kk}$, $\mathbf{S}_{ik}$, $\mathbf{W}_{kj}$ and $\mathbf{g}_k$

EIBA in RKD-SLAM

- Loop constraint term

$$\sum_{(i_1,i_2)\in\mathcal{L}} ||\log(\mathbf{C}_{i_1} \circ \mathbf{C}_{i_2} \circ \mathbf{T}_{i_1 i_2}^{-1})||^2_{\Sigma_{i_1 i_2}}$$

  - Represented as relative pose $\mathbf{T}_{i_1 i_2}$
  - Linearization

$$\mathbf{f}(\mathbf{C}_{i_1}, \mathbf{C}_{i_2}) \approx \mathbf{J}_{i_1}\delta_{\mathbf{C}_{i_1}} + \mathbf{J}_{i_2}\delta_{\mathbf{C}_{i_2}} - \mathbf{e}.$$

  - Update

$$\begin{array}{ll} \mathbf{J}_{i_1}^\top \mathbf{J}_{i_1} & \\ & \mathbf{J}_{i_1}^\top \mathbf{e} \\ \mathbf{J}_{i_1}^\top \mathbf{J}_{i_2} & \\ & \mathbf{J}_{i_2}^\top \mathbf{e} \\ \mathbf{J}_{i_2}^\top \mathbf{J}_{i_2} & \end{array} \Rightarrow \begin{array}{ll} \mathbf{S}_{i_1 i_1} & \\ & \mathbf{g}_{i_1} \\ \mathbf{S}_{i_1 i_2} & \\ & \mathbf{g}_{i_2} \\ \mathbf{S}_{i_2 i_2} & \end{array}$$

Performance of EIBA

- Computation time

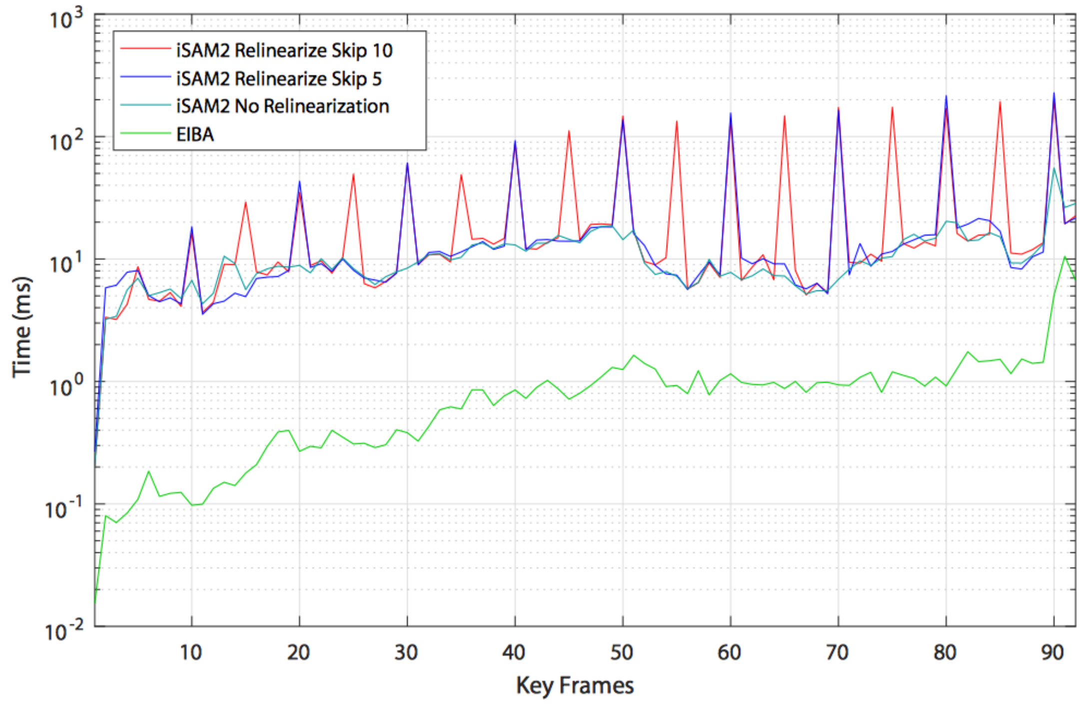


Fig. 4. The computation time of our EIBA and iSAM2 while incrementally adding each new keyframe on "fr3_long_office" sequence.

Performance of EIBA

- Computation time
  - Our EIBA is faster by an order of one magnitude than iSAM2.

Sequence	Num. of Camera / Points	Num. of Observations	EIBA	iSAM2		
				No relinearization	relinearizeSkip = 10	relinearizeSkip = 5
fr3_long_office	92 / 4322	12027	88.9ms	983.9ms	1968.2ms	2670.9ms
fr2_desk	63 / 2780	6897	34.8ms	507.8ms	850.4ms	1152.0ms

Performance of EIBA

- Optimized reprojection error

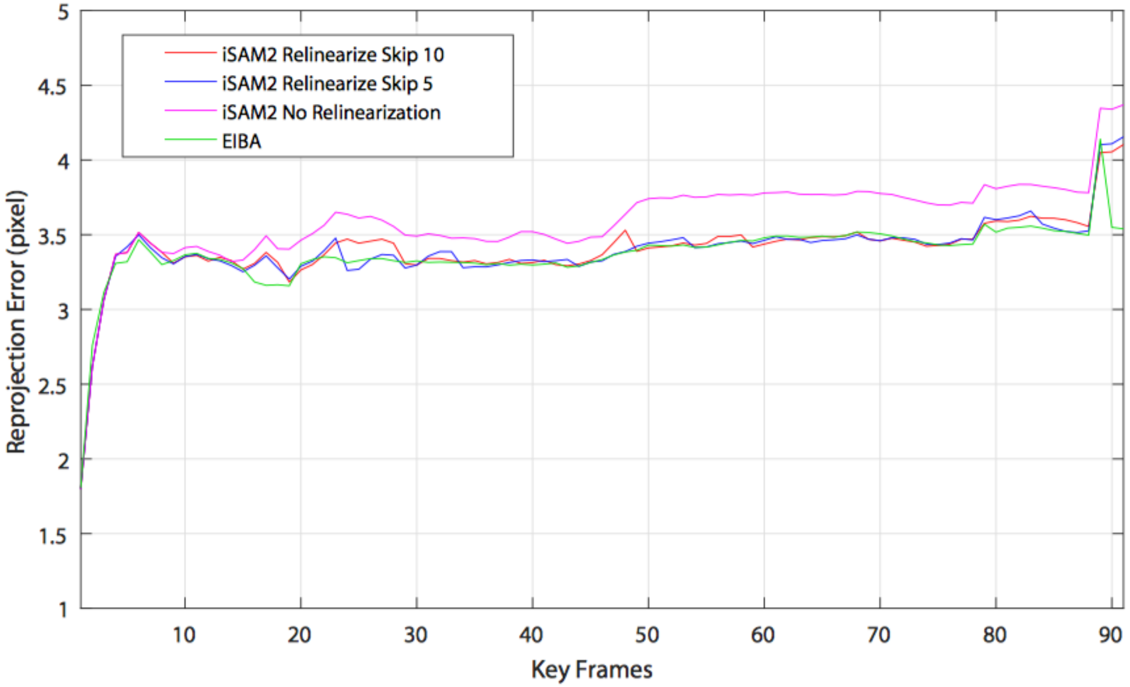


Fig. 5. The optimized reprojection error (RMSE) for our EIBA and iSAM2 while incrementally adding each new keyframe on “fr3_long_office” sequence.

Open-source Solver & BA

- g2o: https://github.com/RainerKuemmerle/g2o
- GTSAM& iSAM: https://bitbucket.org/gtborg/gtsam/
- Ceres Solver: http://ceres-solver.org/
- Bundler: http://www.cs.cornell.edu/~snavely/bundler/
- PBA: https://grail.cs.washington.edu/projects/mcba/
- EIBA: the source code will be released soon.
  http://www.zjucvg.net