

# From Square Root SAM to GTSAM: Factor Graphs in Robotics

Frank Dellaert, Georgia Institute of Technology  
Michael Kaess, Carnegie Mellon University

5 years ago I did a sabbatical at a startup to help them build the most most advanced flying AI on the planet: the Skydio drone



12 Navigation Cameras





Earlier this year the Skydio 2 was released, which innovates in both 360 perception and superior autonomy



# Skydio 2



To deliver value, the autonomy stack has to support superior navigation, tracking, and motion planning at very low power

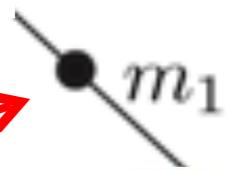


Many of these at their core are optimization problems with locality, which is captured well by *factor graphs*

Robot



Landmark  
measurement

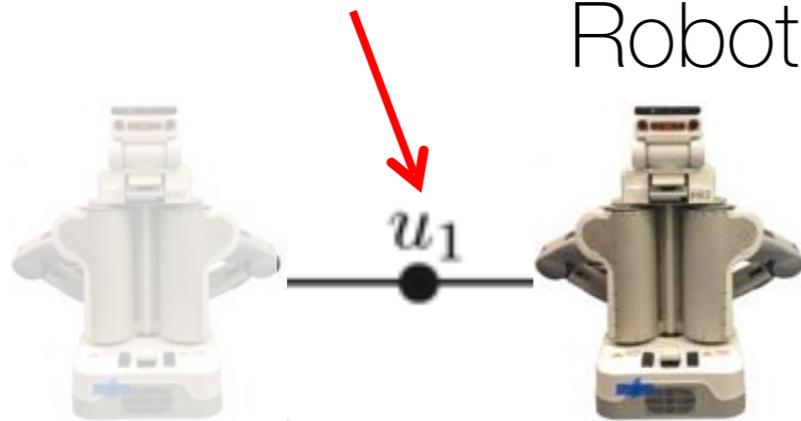


Landmark

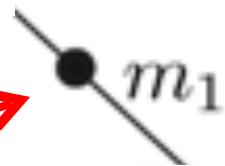
Many of these at their core are optimization problems with locality, which is captured well by *factor graphs*

Odometry measurement

Robot



Landmark measurement



Landmark 1



Landmark 2

Many of these at their core are optimization problems with locality, which is captured well by *factor graphs*

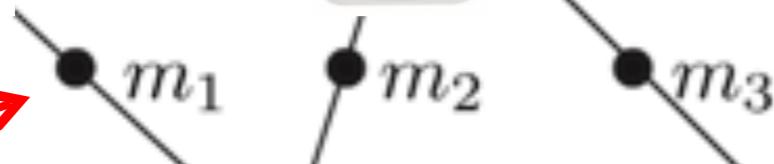
Odometry measurement



Measurements are uncertain

Drift accumulates

Landmark measurement

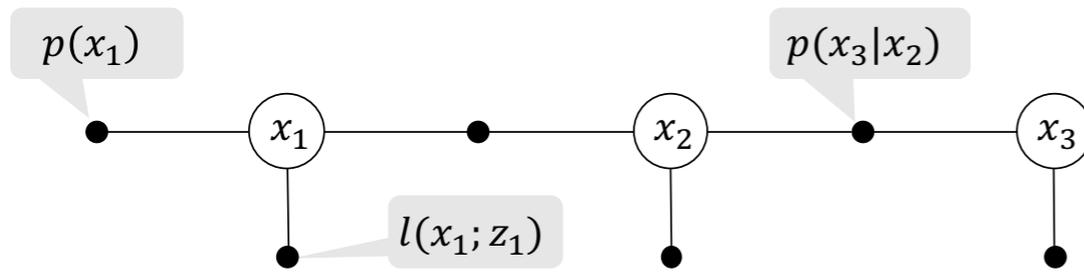


“Loop closure”

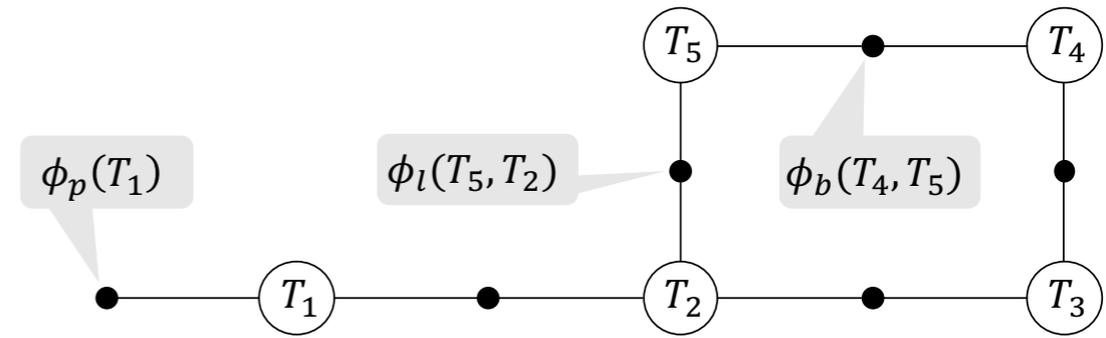
Landmark 1

Landmark 2

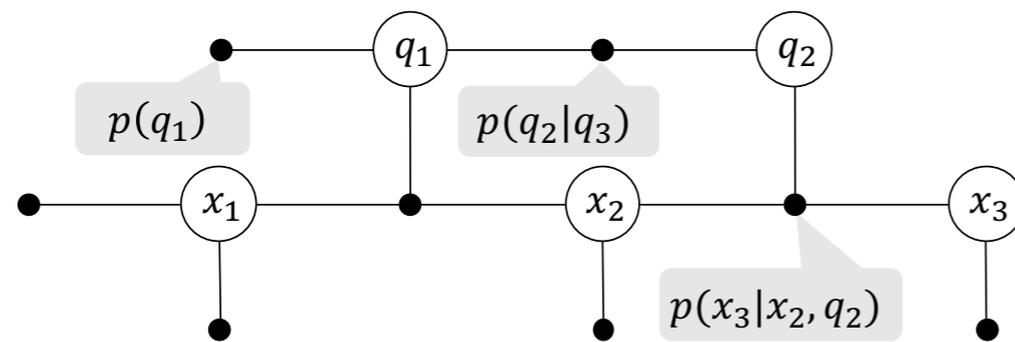
Factor graphs can represent many robotics problems, from tracking to optimal control to sophisticated 3D mapping



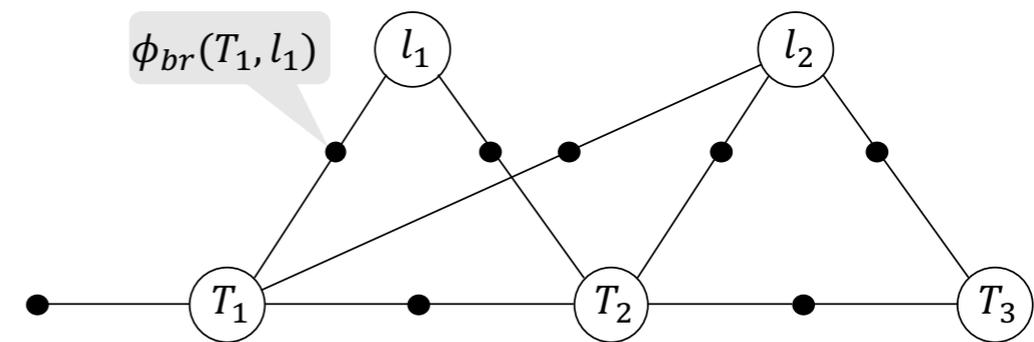
Tracking



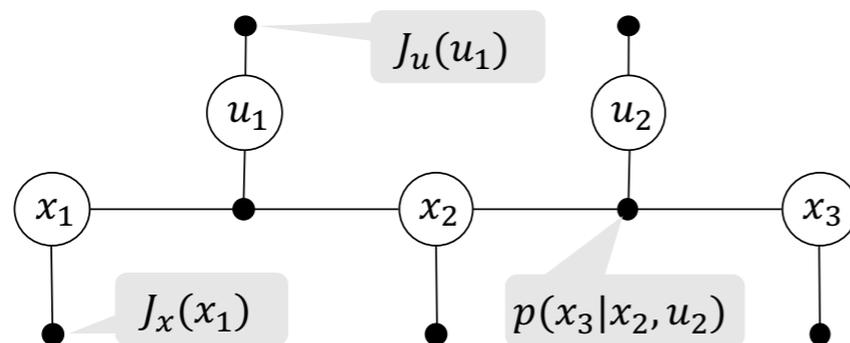
Pose graph



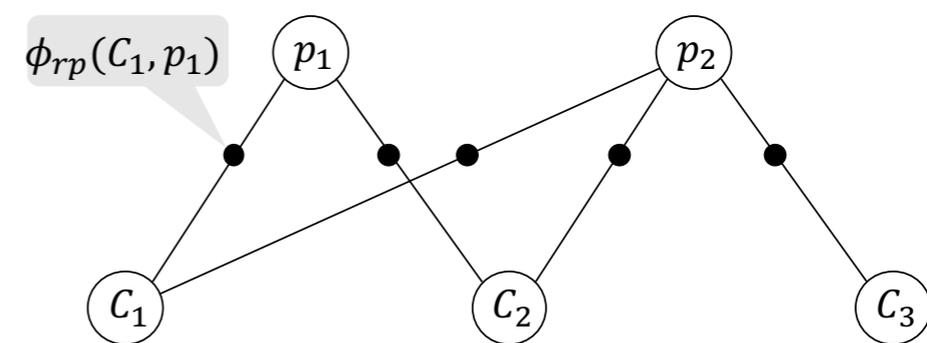
Switching System



SLAM



Optimal Control



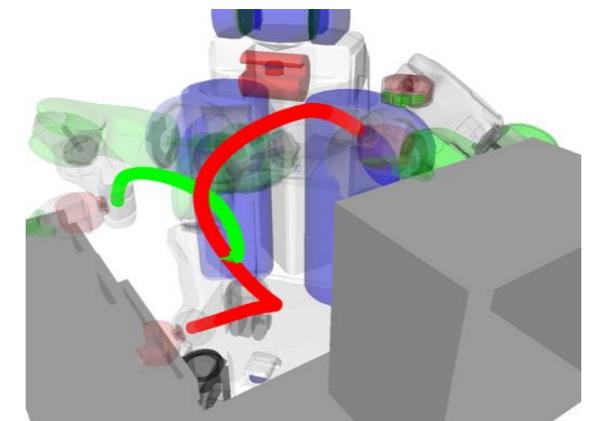
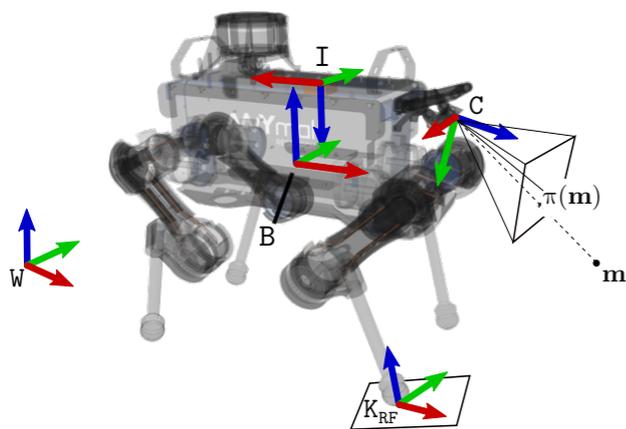
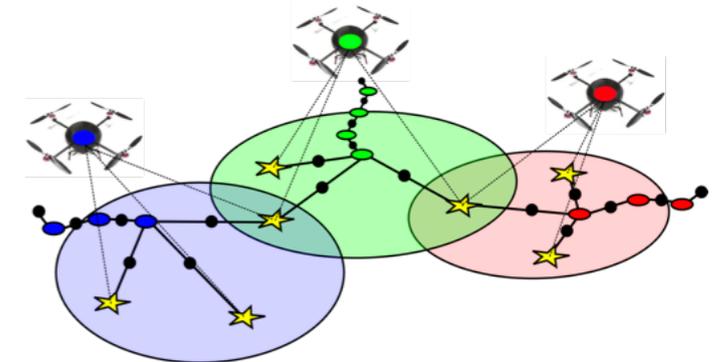
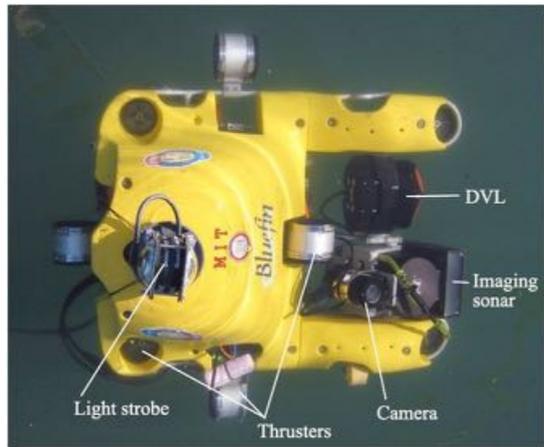
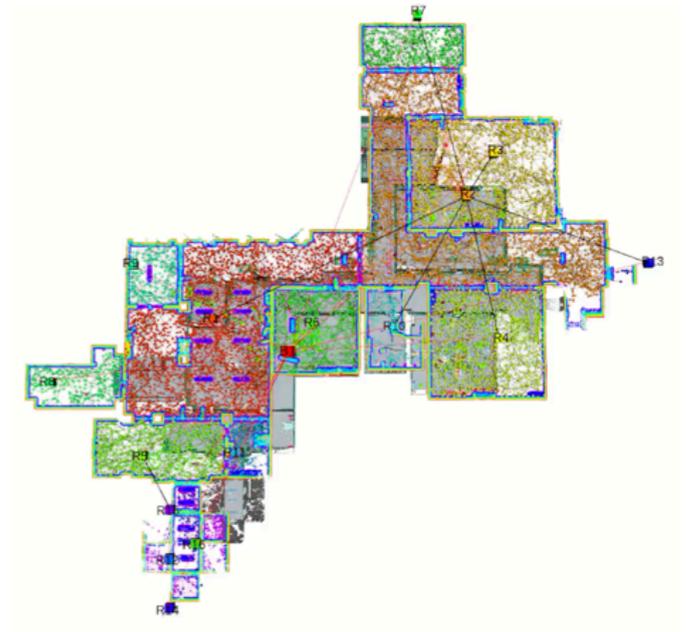
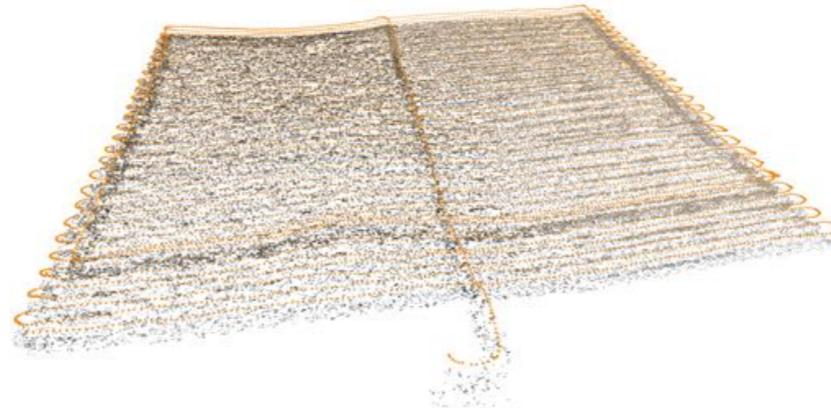
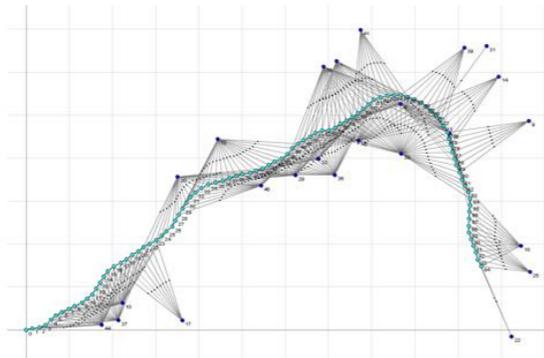
Structure from Motion

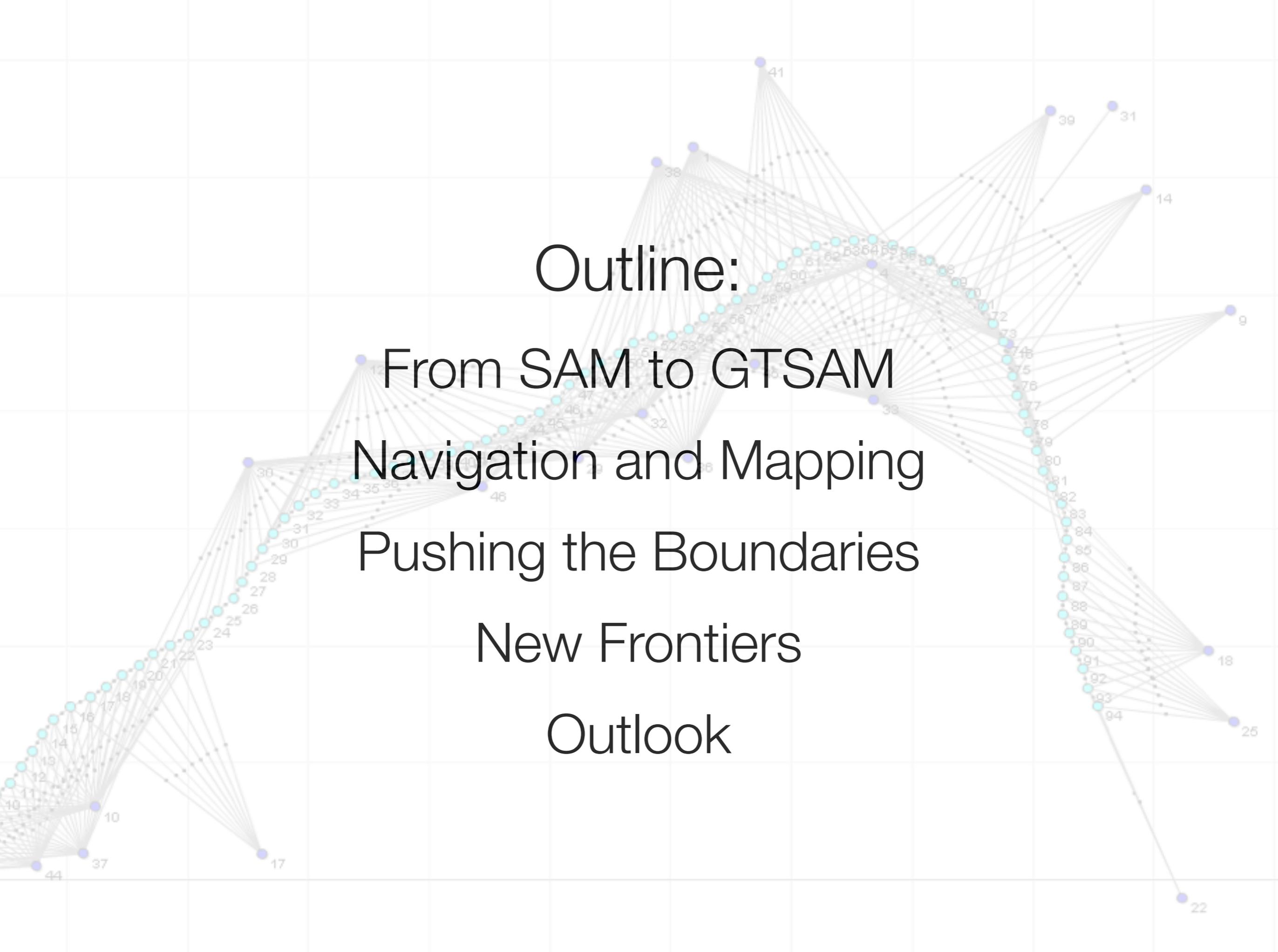
Factor graphs expose opportunities for raw speed because of the deep connection with sparse linear algebra

- Ordering heuristics
- Nested Dissection
- Sparsification
- Pre-integration
- Iterative Solvers
- Incremental Inference and the Bayes tree



Factor graphs are beneficial in designing and thinking about your problem, even aside from performance





Outline:

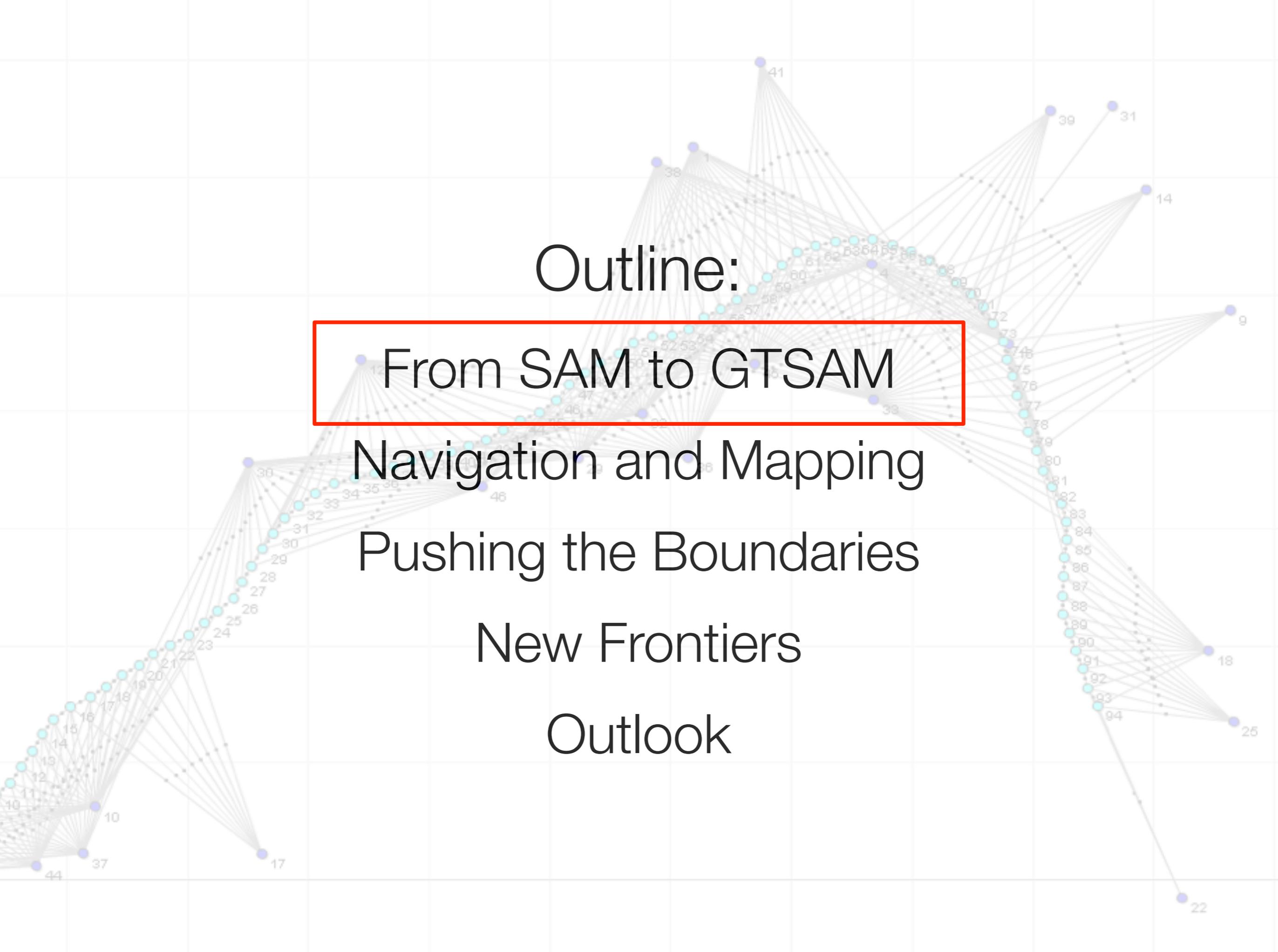
From SAM to GTSAM

Navigation and Mapping

Pushing the Boundaries

New Frontiers

Outlook



# Outline:

From SAM to GTSAM

Navigation and Mapping

Pushing the Boundaries

New Frontiers

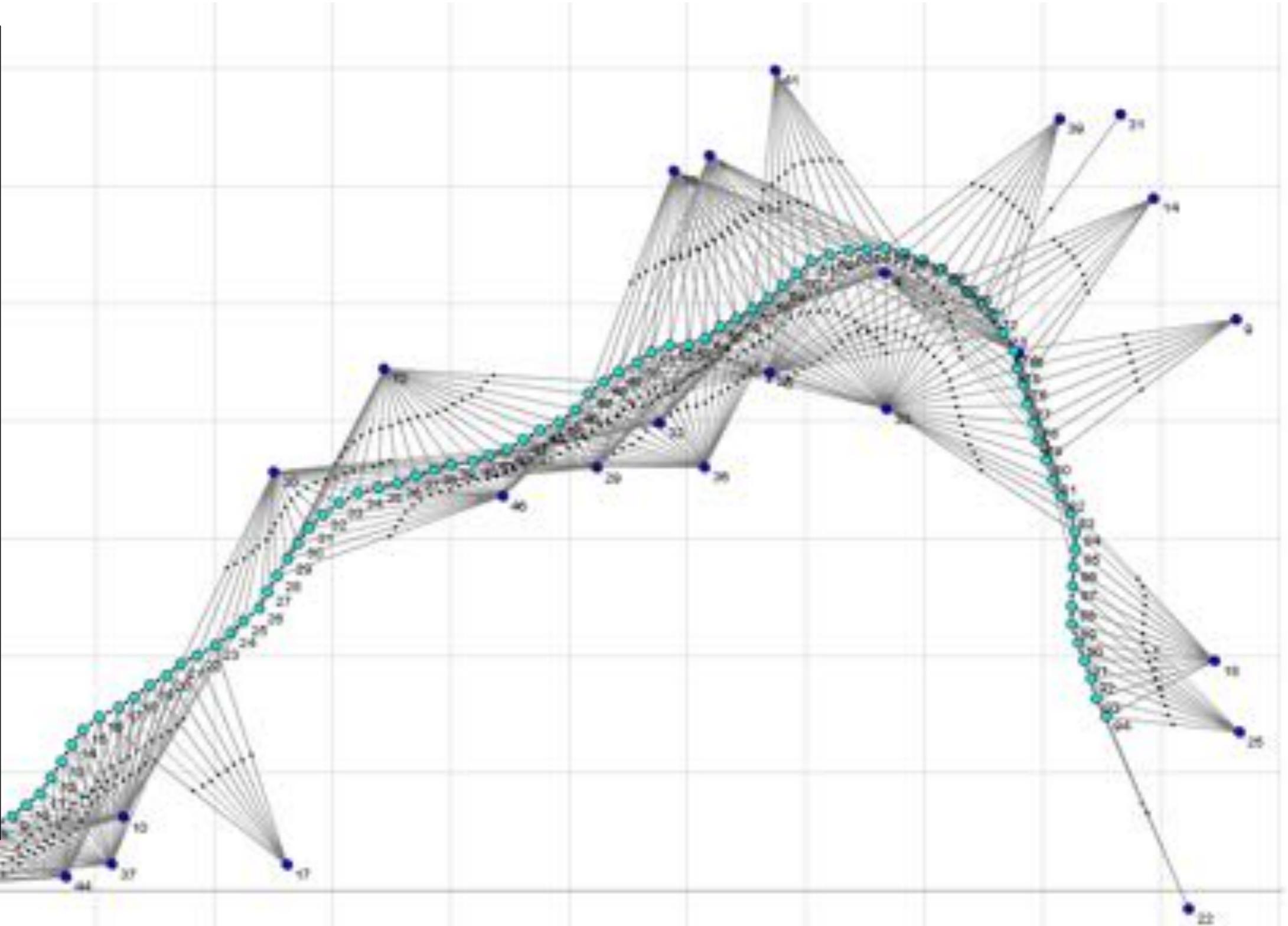
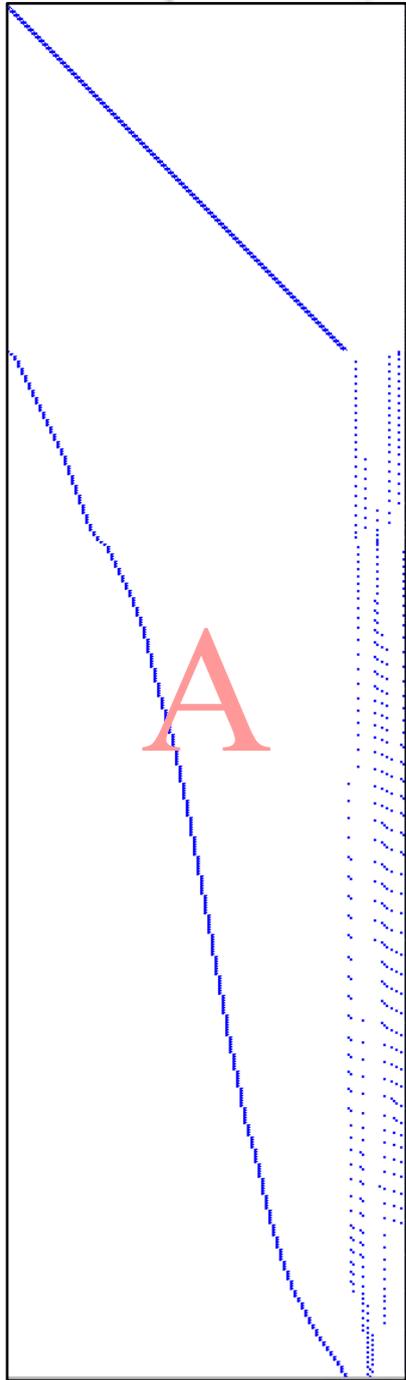
Outlook

In SAM we are interested in inferring the trajectory of the robot and a map of the unknown environment

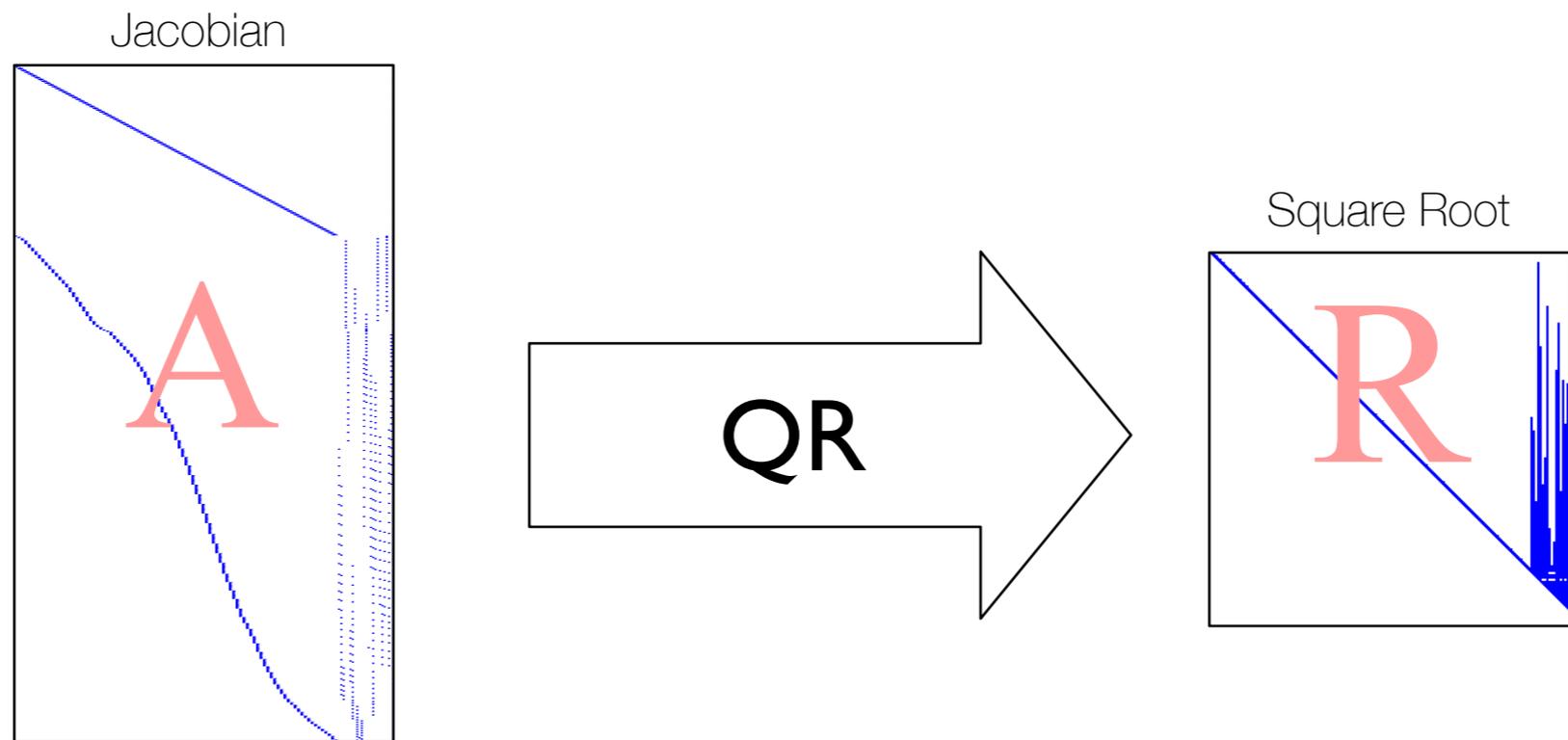


The factor graph associated with a small SAM problem instantaneously shows the structure of the problem

Jacobian



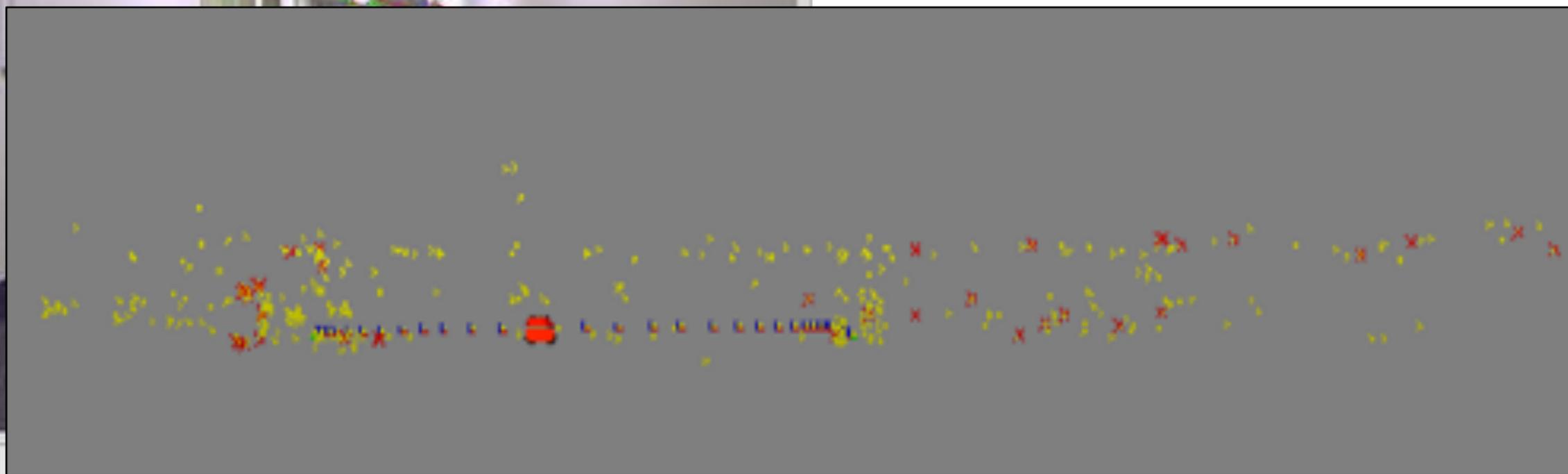
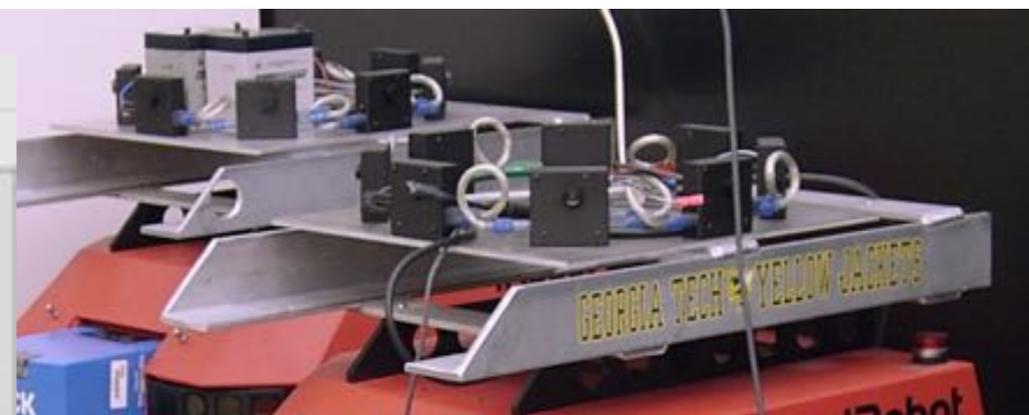
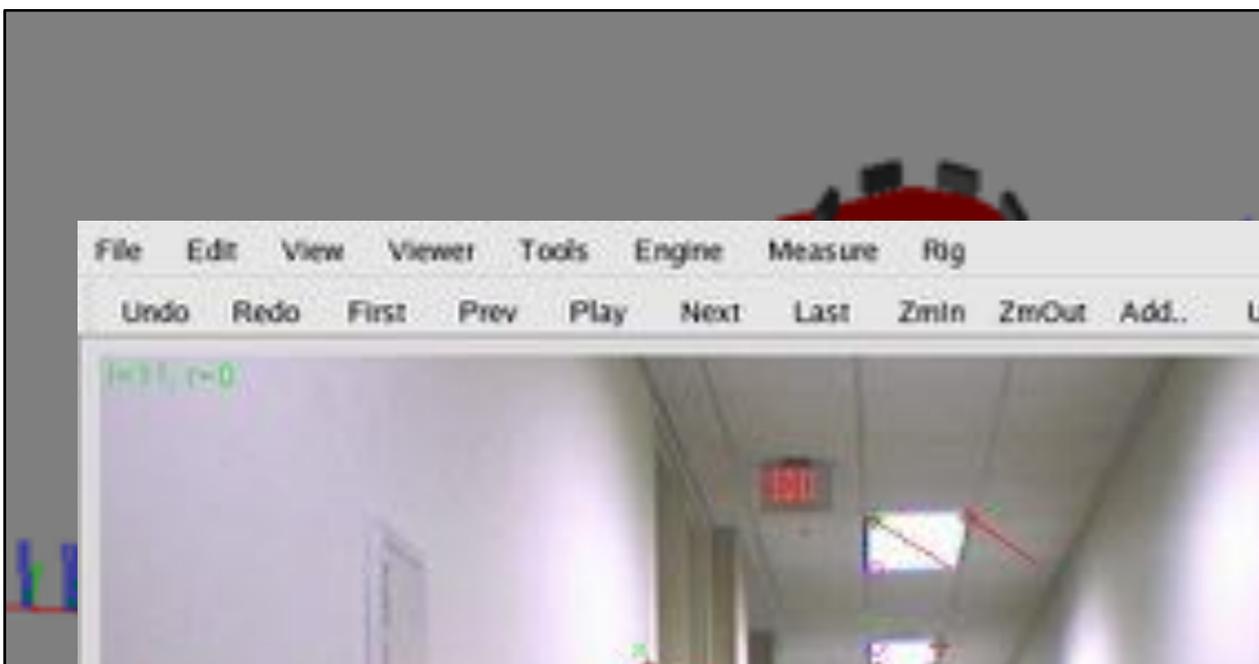
In practice, Square Root SAM is implemented using sparse matrix factorization, which is a computation on a graph



QR Factorization on Factor Graph

Visual SLAM in 2005 might have looked a bit cheesy, but we already did 8-camera visual SLAM back then

We used non-vacuum iRobots



# The key points from the Square Root SAM papers have stood the test of time, but we know so much more now

- Key points:

- Matrices  $\Leftrightarrow$  Graphs

- Factorization  $\Leftrightarrow$  Variable Elimination

- Improving Performance  $\Leftrightarrow$  Variable Ordering

- What we know now:

- Factor graphs can represent many robotics problems

- Factor graphs expose opportunities to improve computational performance

- Factor graphs are beneficial in designing and thinking about your problem, even aside from performance

**Square Root SAM**  
Simultaneous Localization and Mapping  
via Square Root Information Smoothing  
Frank Dellaert and Michael Kaess  
Center for Robotics and Intelligent Machines, College of Computing  
Georgia Institute of Technology, Atlanta, GA 30332-0280  
To appear in the Intl. Journal of Robotics Research

**Abstract**  
Solving the SLAM problem is one way to enable a robot to explore, map, and navigate in a previously unknown environment. We investigate smoothing approaches as a viable alternative to extended Kalman filter-based solutions to the problem. In particular, we look at approaches that factorize either the associated information matrix or the measurement Jacobian into square root form. Such techniques have several significant advantages over the EKF: they are faster yet exact, they can be used in either batch or incremental mode, are better equipped to deal with non-linear processes and measurement models, and yield the entire robot trajectory, at least over a large class of SLAM problems. In addition, as an indirect but dramatic way, column ordering heuristics automatically exploit the locality inherent in the geographic nature of the SLAM problem. In this paper we present the theory underlying these methods, along with an interpretation of factorization in terms of the graphical model associated with the SLAM problem. We present both simulation results and actual SLAM experiments in large-scale environments that underscore the potential of these methods as an alternative to EKF-based approaches.

**1 Introduction**  
The problem of simultaneous localization and mapping (SLAM) [69, 51, 76] has received considerable attention in mobile robotics as it is one way to enable a robot to explore and navigate previously unknown environments. In addition, in many applications the environment itself is the artifact of interest, e.g. in urban reconstruction, search-and-rescue operations, battlefield reconnaissance etc. As such, it is one of the core competencies of autonomous robots [77]. We will primarily be concerned with landmark-based SLAM, for which the earliest and most popular methods are based on the extended Kalman filter (EKF) [70, 62, 61, 2, 71, 53, 51]. The EKF recursively estimates a Gaussian density over the current pose of the robot and the position of all landmarks (the map). However, it is well known that the computational complexity of the EKF becomes untenable fairly quickly, and hence a large number of efforts have focused on

5 A GRAPHICAL MODEL PERSPECTIVE 11

from left to right. For each column  $j$ , all non-zero elements below the diagonal are zeroed out by multiplying  $A$  on the left with a *Householder reflection matrix*  $H_j$ . After  $n$  iterations  $A$  is completely factored:

$$H_n \dots H_1 A = Q^T A = \begin{bmatrix} R \\ 0 \end{bmatrix} \quad (12)$$

The orthogonal matrix  $Q$  is not usually formed: instead, the transformed RHS  $Q^T b$  is computed by appending  $b$  as an extra column to  $A$ . Because the  $Q$  factor is orthogonal, we have:

$$\|A\|_F - \|b\|_2 = \|Q^T A - Q^T b\|_F = \|R - b\|_2 = \|r\|_2$$

Clearly,  $\|r\|_2$  will be the least-squares residual, and the LS solution  $x^*$  can be obtained by solving the square system

$$R\hat{x} = d \quad (13)$$

via back-substitution. The cost of QR is dominated by the cost of the Householder reflections, which is  $2(n^3 - n)/3$ .

Comparing QR with Cholesky factorization, we see that both algorithms require  $O(mn^2)$  operations when  $m \geq n$ , but that QR factorization is a factor of 2 slower. While these numbers are valid for dense matrices only, we have seen that in practice LDL and Cholesky factorization far outperform QR factorization on sparse problems as well, and not just by a constant factor.

5 A Graphical Model Perspective

**5.1 Matrices  $\Leftrightarrow$  Graphs**  
From the exposition above it can now be readily appreciated that the measurement Jacobian  $A$  is the matrix of the factor graph associated with SLAM. We can understand this statement at two levels. First, every block of  $A$  corresponds to one term in the least-squares criterion (8), either a landmark measurement or an odometry constraint, and every block row corresponds to one factor in the factor graph. Within each block-row, the sparsity pattern indicates which unknown poses and/or landmarks are connected to the factor. Hence, the *block-structure* of  $A$  corresponds exactly to the *adjacency matrix* of the factor graph associated with SAM.

Second, at the scalar level, every row  $A_{ij}$  in  $A$  (see Figure 4) corresponds to a scalar term  $\|A_{ij} - h_{ij}\|_2$  in the sparse matrix least-squares criterion (9), as

$$\|A - h\|_2^2 = \sum_i \|A_{i\cdot} - h_{i\cdot}\|_2^2$$

Hence, this defines a *finely structured factor graph*, via

$$P(\theta) \propto \exp\left(-\frac{1}{2}\|A - h\|_2^2\right) = \prod_i \exp\left(-\frac{1}{2}\|A_{i\cdot} - h_{i\cdot}\|_2^2\right)$$

It is important to realize, that in this factor view, the block-structure of the SLAM problem is discarded, and that it is this graph that is examined by general purpose linear algebra methods. By working with the block-structure instead, we will be able to do better.

5 A GRAPHICAL MODEL PERSPECTIVE 12

As noted before in [78] and by others, the information matrix  $Z = A^T A$  is the matrix of the Markov random field associated with the SLAM problem. Again, at the block-level the sparsity pattern of  $A^T A$  is exactly the adjacency matrix of the associated MRF. The objective function in Equation 5 corresponds to a pairwise Markov random field (MRF) [81, 82] through the Hammersley-Clifford theorem [81], and the nodes in the MRF correspond to the robot states and the landmarks. Links represent either odometry or landmark measurements.

In [85, 79] the MRF graph view is taken to expose the correlation structure inherent in the filtering version of SLAM. It is shown there that inevitably, when marginalizing out the past trajectory  $X_{1:t-1}$ , the information matrix becomes completely dense. Hence, the emphasis in these approaches is to selectively remove links to reduce the computational cost of the filter, with great success. In contrast, in this paper we consider the MRF associated with the smoothing information matrix  $Z$ , which does not become dense, as past states are never marginalized out.

5.2 Factorization  $\Leftrightarrow$  Variable Elimination

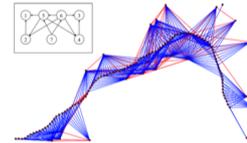


Figure 5: The triangulated graph for a good ordering (colored, as in Figure 4). This is a directed graph, where each edge corresponds to a non-zero in the Cholesky triangle  $R$ . Note that we have dropped the arrows in the simulation example for simplicity.

The one question left is what graph the square root information matrix  $R$  corresponds to? Remember that  $R$  is the result of factoring either  $Z$  or  $A$ , as in Section 4. Cholesky or QR factorization are most often used as “black box” algorithms, but in fact they are actually much more recently developed methods for inference in graphical models [11]. It will be seen below that  $R$  is essentially in correspondence with junction tree, known from inference in graphical models and also recently applied to SLAM [65].

Both factorization methods, QR and Cholesky (LDL), are based on the variable elimination algorithm [4, 11]. The difference between these methods is that QR eliminates variable nodes from the factor graph and obtains  $A = QR$ , while Cholesky or LDL start from the MRF and hence obtain

5 A GRAPHICAL MODEL PERSPECTIVE 15

from the leaves to the root to factorize a sparse matrix, and then from the root to the leaves to perform a backward substitution step. A complete treatment of the relationship between square root information matrices and clique trees is beyond the scope of the current paper, but in other work we have used the clique-tree structure in novel algorithms for distributed inference [19].

5.3 Improving Performance  $\Leftrightarrow$  Reducing Fill-in

The single most important factor to good performance is the order in which variables are eliminated. Different variable orderings can yield dramatically more or less fill-in, defined as the amount of edges added into the graph during factorization. As each edge added corresponds to a non-zero in the Cholesky triangle  $R$ , both the cost of computing  $R$  and back-substitution is heavily dependent on how much fill-in occurs. Unfortunately, finding an optimal ordering is NP-complete. Discovering algorithms that approximate the optimal ordering is an active area of research in sparse linear algebra. A popular method for medium-sized problems is column [1], which works on the columns of  $A$ . Another popular method, based on graph theory and often used to speed up finite element methods, is generalized nested dissection [55, 54].

Given that an optimal ordering is out of reach in general, heuristics or domain knowledge can do much better than general-purpose algorithms. A simple idea is to use a standard method such as column, but have it work on the sparsity pattern of the blocks instead of pointing to the original measurement Jacobian  $A$ . In other words we treat a collection of scalar variables such as the  $x$  and  $y$  position and orientation as a single variable and create a smaller graph which encapsulates the constraints between these blocks rather than the individual variables. Not only is it cheaper to call column on this smaller graph, it also leads to a substantially improved ordering. As we mentioned above, the block-structure is real knowledge about the SLAM problem and is not accessible to column or any other approximate ordering algorithm. While the effect on the performance of column is negligible, we have found that making it work on the SLAM MRF instead of on the sparse matrix  $Z$  directly can yield improvements of 2 to sometimes a 100-fold, with 15 being a good rule of thumb.

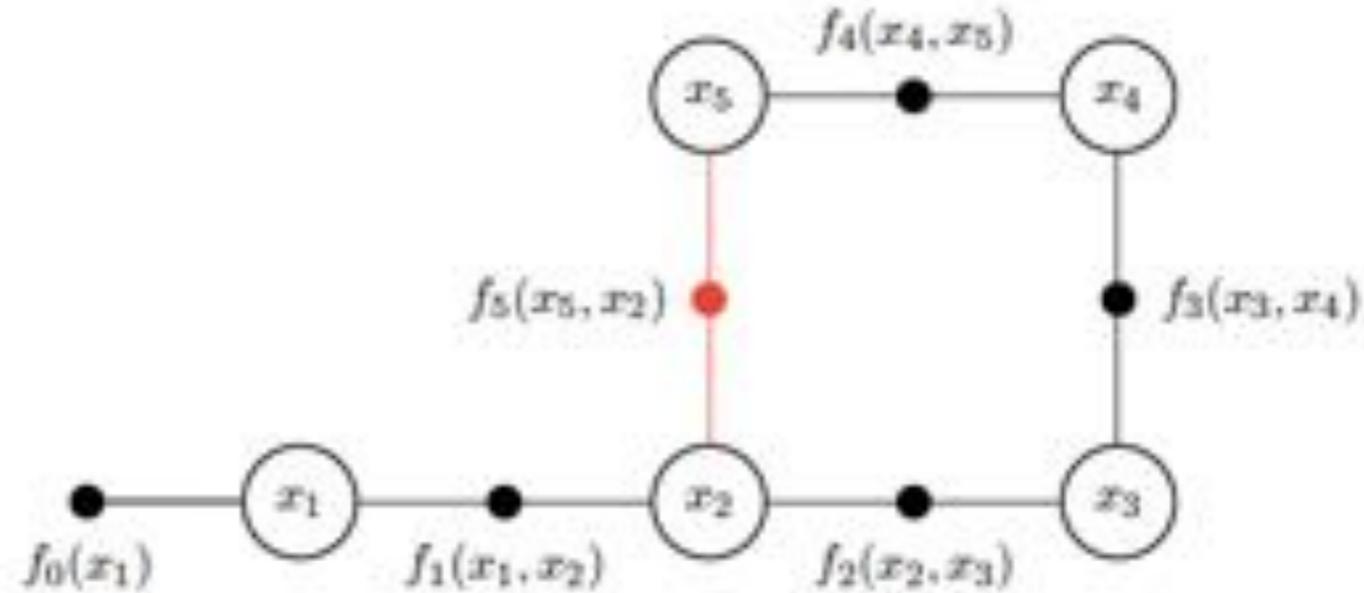
Note that there are cases in which any ordering will result in the same large fill-in. The worst-case scenario is a fully connected bipartite MRF: every landmark is seen from every location. In that case, eliminating any variable will completely connect all variables on the other side, and after that the structure of the clique tree is completely known: if a pose was chosen first, the root will be the entire map, and all poses will be computed once the map is known. Vice versa, if a landmark is chosen, the trajectory will be the clique tree root clique, and computation will proceed via an (expensive) trajectory optimization, followed by (very cheap) computation of landmarks. Interestingly, these two cases form the basis of the standard partitioned inverse, or “Schur complement”, which is well-known in structure from motion applications [79, 41] and also used in GraphSLAM [77].

However, the worst-case scenario outlined above is an exceptional case in robotics: sensors have limited range and are occluded by walls, objects, buildings, etc. This is especially true in large-scale mapping applications, and it essentially means that the MRF will in general be sparsely connected, even though it is one large connected component.

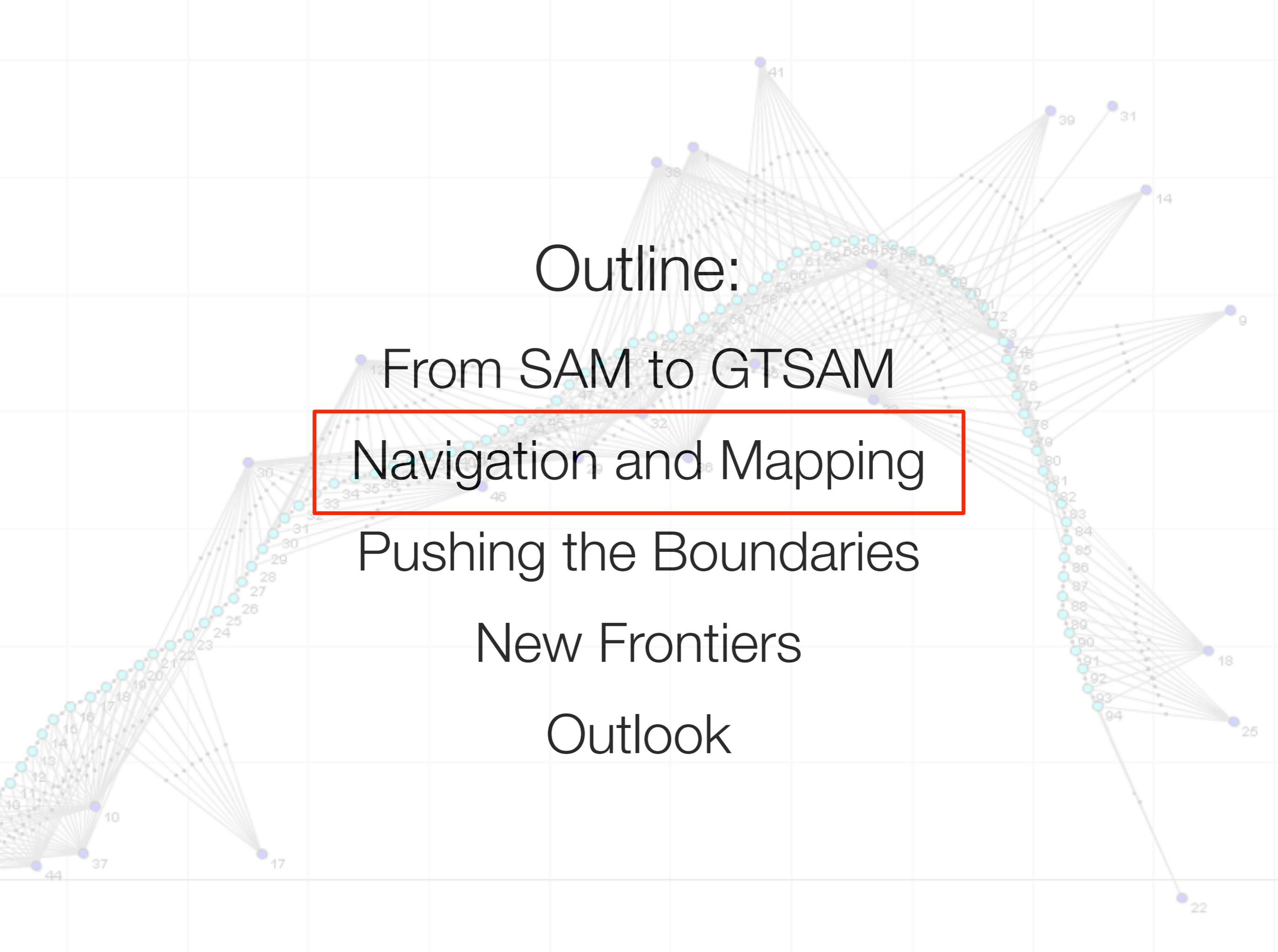
<sup>1</sup>Equivalently, one can use to indicate the partition of variables into poses and landmarks, not the factor graph sense.

# GTSAM embodies many of the ideas we and others have developed around factor graphs since then

- C++ library: [gtsam.org](http://gtsam.org)
- python & Matlab wrappers
- Open-source, BSD-licensed
- Optimization on Manifolds and Lie groups
- Reverse AD Expression Language



```
1 NonlinearFactorGraph graph;
2 noiseModel::Diagonal::shared_ptr priorNoise =
3   noiseModel::Diagonal::Sigmas(Vector_(3, 0.3, 0.3, 0.1));
4 graph.add(PriorFactor<Pose2>(1, Pose2(0, 0, 0), priorNoise));
5
6 // Add odometry factors
7 noiseModel::Diagonal::shared_ptr model =
8   noiseModel::Diagonal::Sigmas(Vector_(3, 0.2, 0.2, 0.1));
9 graph.add(BetweenFactor<Pose2>(1, 2, Pose2(2, 0, 0), model));
10 graph.add(BetweenFactor<Pose2>(2, 3, Pose2(2, 0, M_PI_2), model));
11 graph.add(BetweenFactor<Pose2>(3, 4, Pose2(2, 0, M_PI_2), model));
12 graph.add(BetweenFactor<Pose2>(4, 5, Pose2(2, 0, M_PI_2), model));
13
14 // Add pose constraint
15 graph.add(BetweenFactor<Pose2>(5, 2, Pose2(2, 0, M_PI_2), model));
```



Outline:

From SAM to GTSAM

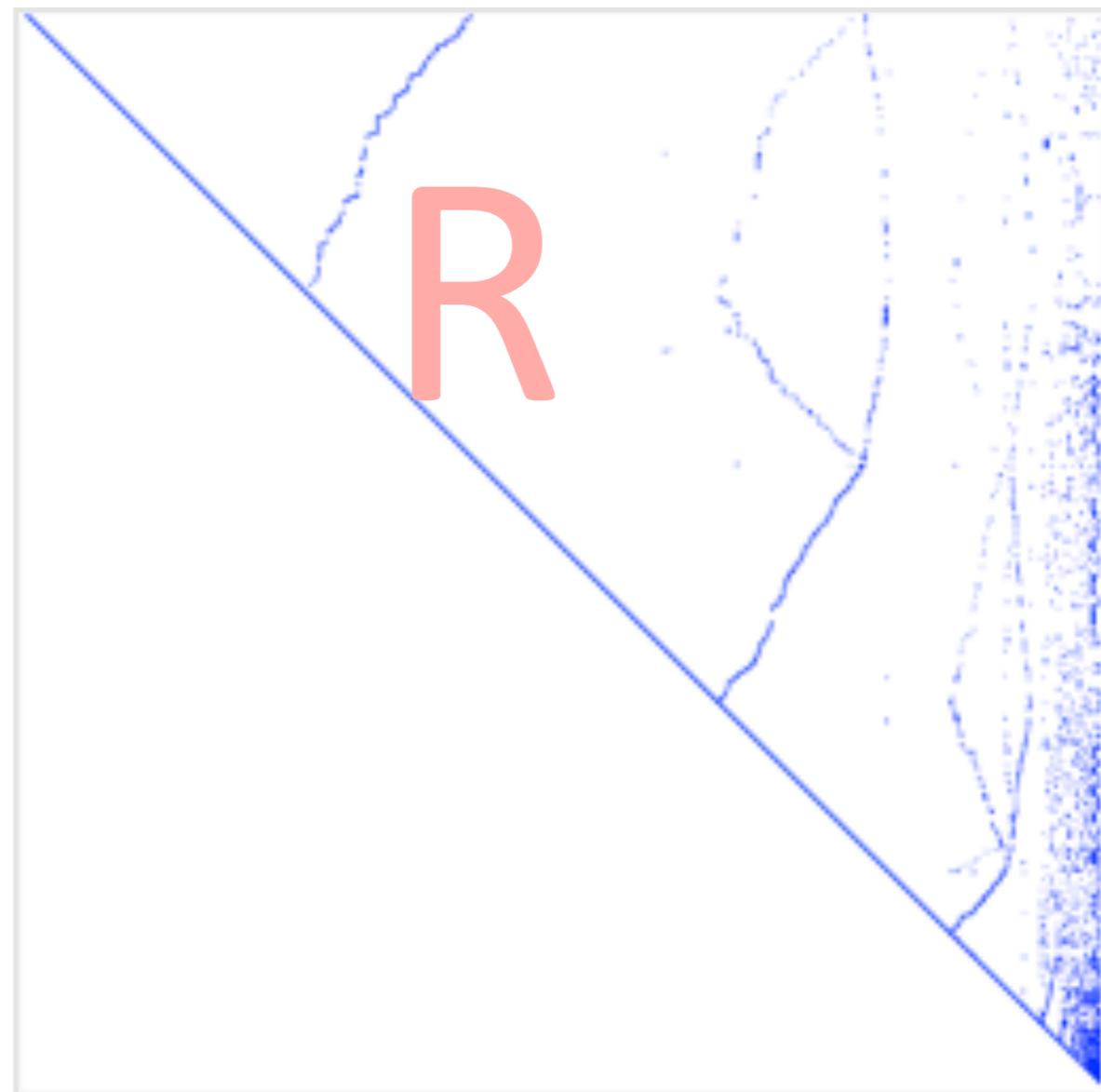
Navigation and Mapping

Pushing the Boundaries

New Frontiers

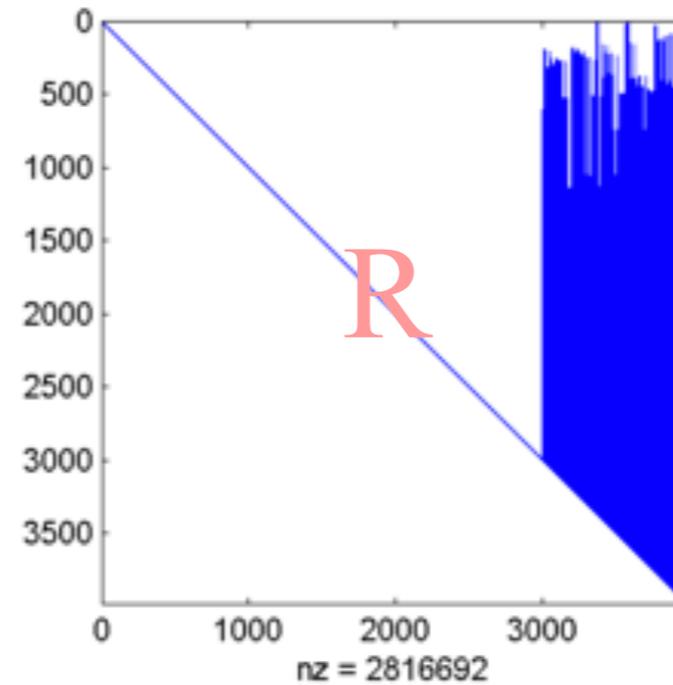
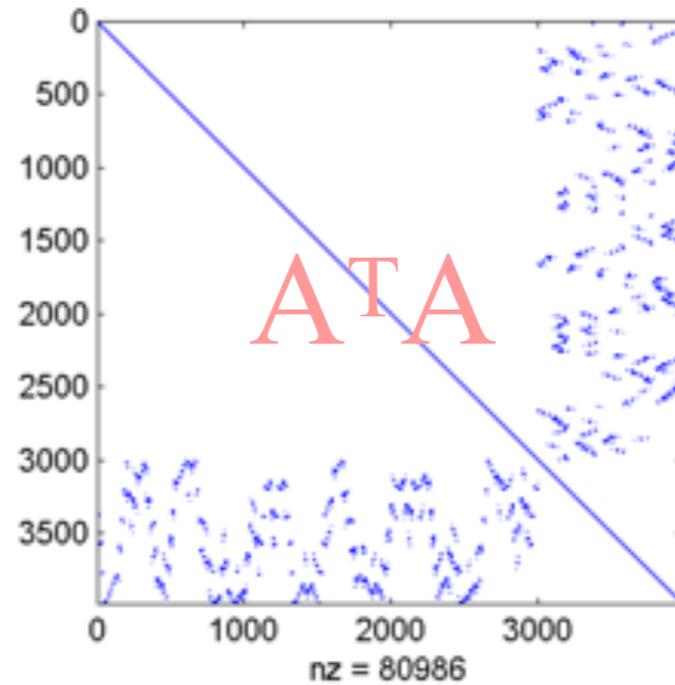
Outlook

Square Root SAM on real sequence, the Sydney Victoria Park dataset, shows how sparsity is key to performance



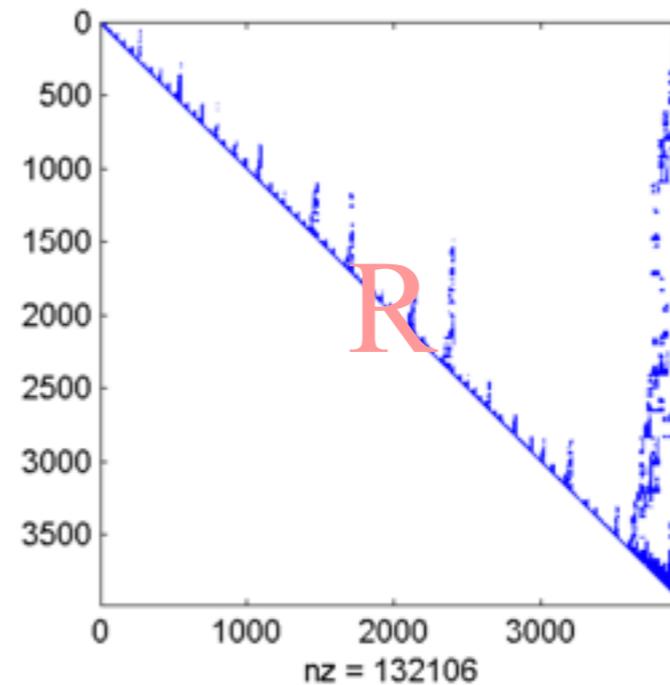
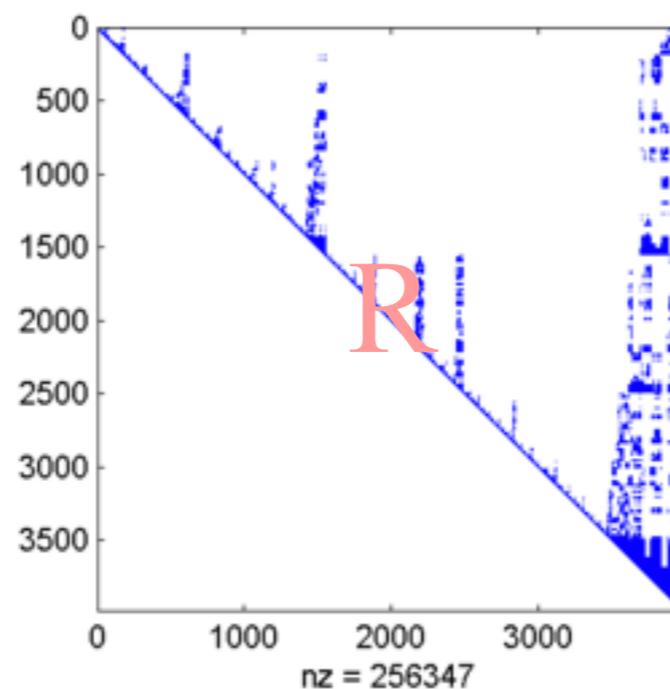
While finding an optimal ordering is NP complete, heuristics coupled with domain knowledge can do wonders

Information matrix



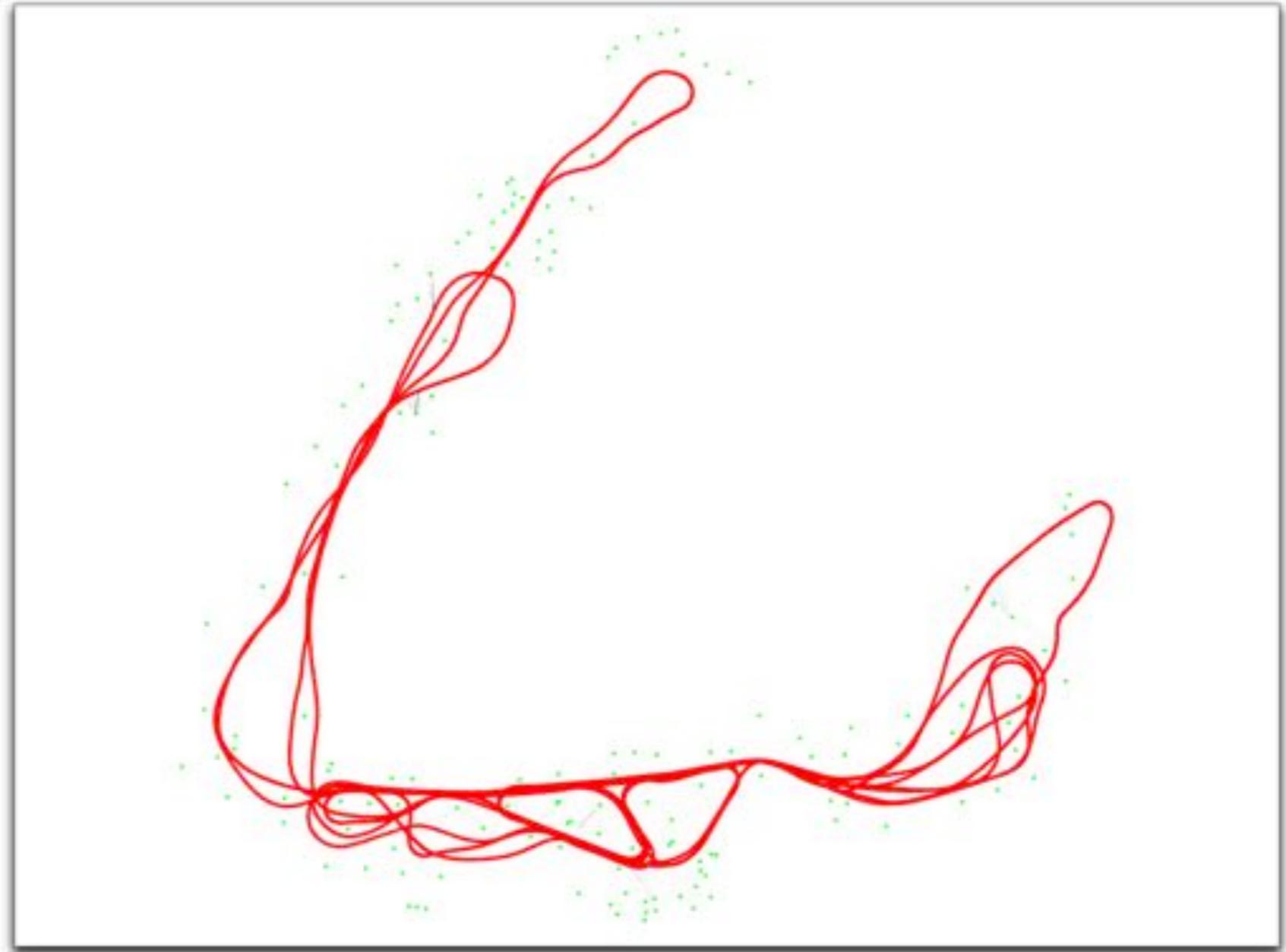
Square Root:  
Naive

Square Root:  
AMD



Square Root:  
AMD on blocks

Domain knowledge often shows how to break up graphs, a generalization of “nested dissection” [Kai Ni et al. IROS '10]



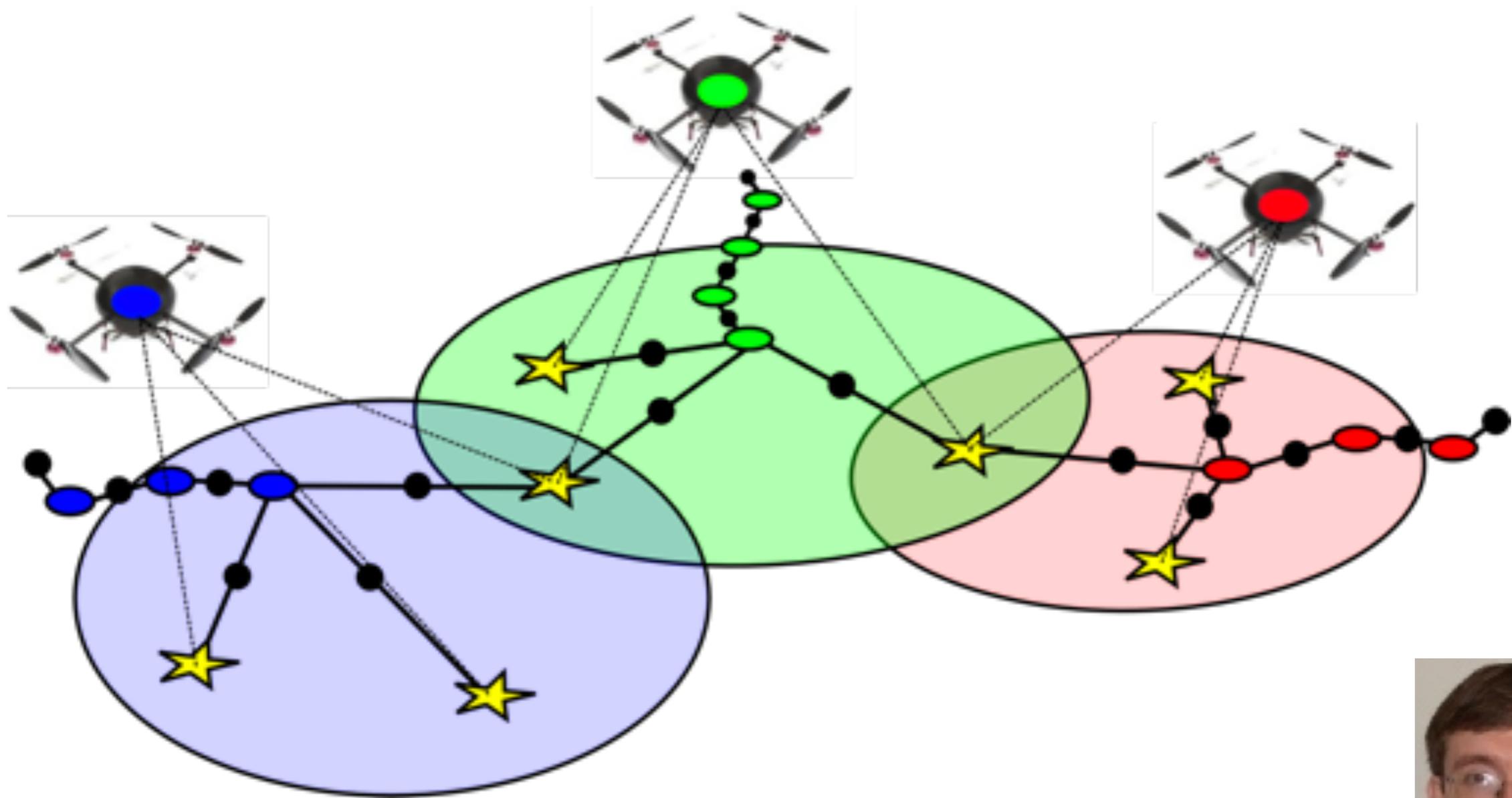
Now CEO of Beijing autonomous driving startup

 **HOLOMATIC** 禾多科技

Hyper-SFM applies hierarchical nested dissection to the structure from motion problem [Ni et al. 3DIMPVT'12]

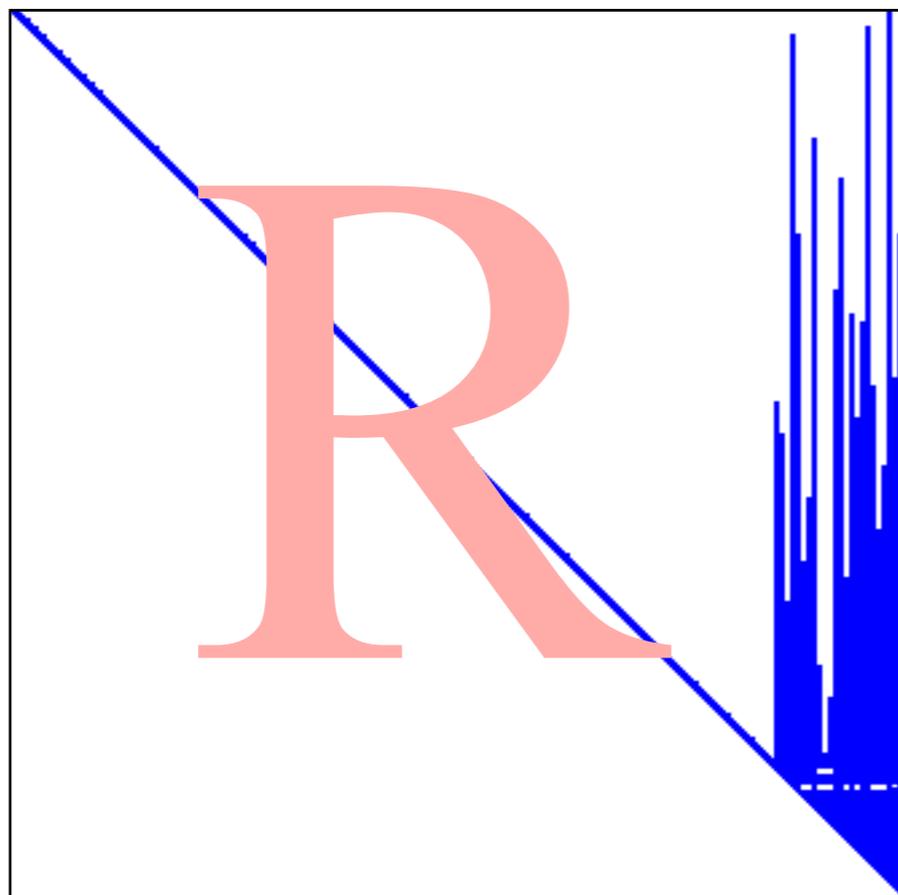


Breaking up graphs can lead to powerful new paradigms for distributed mapping [Alex Cunningham et al, ICRA '13]

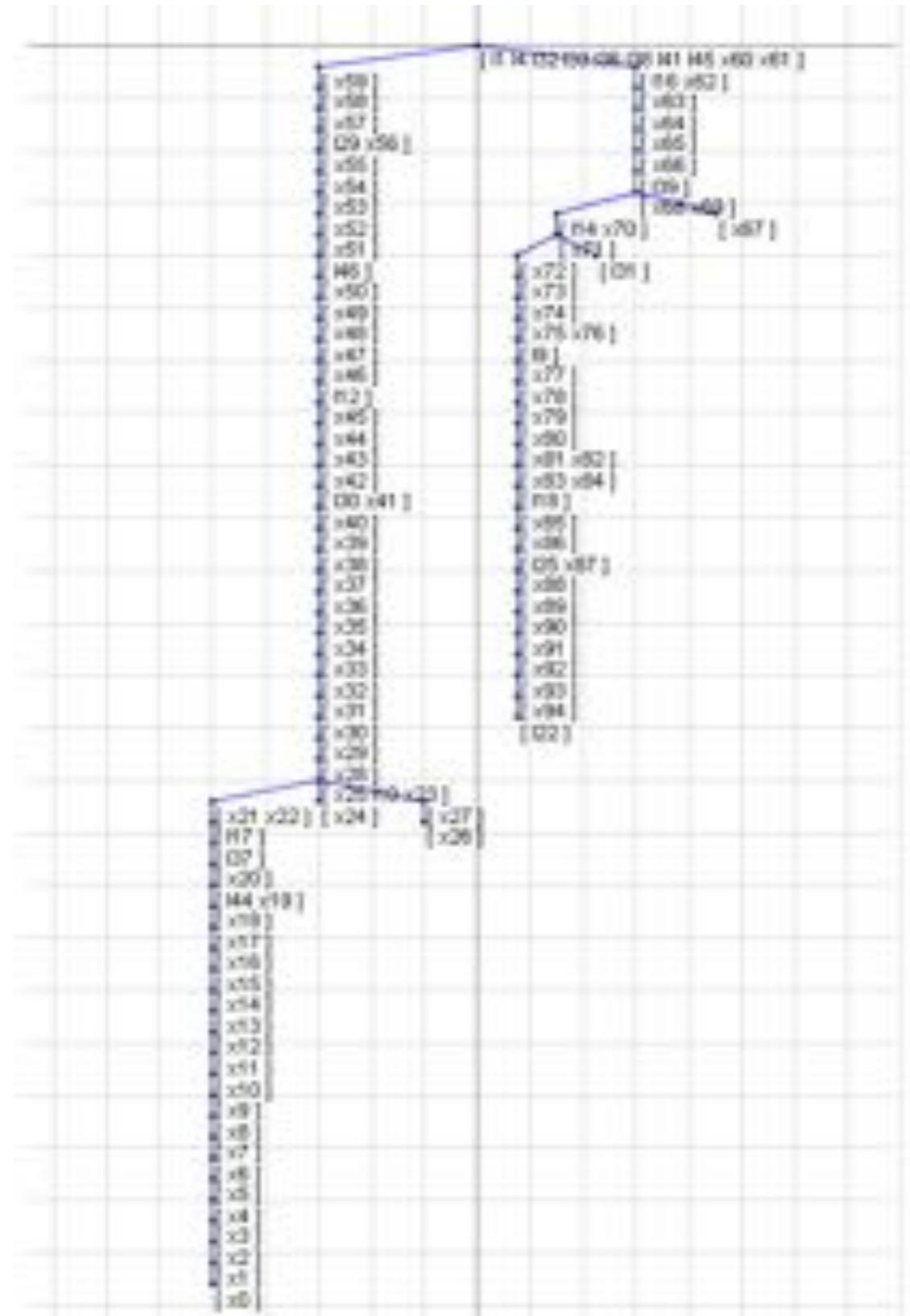


The Bayes tree is a powerful graphical model that enables incremental Smoothing and Mapping (iSAM) [IJRR '12]

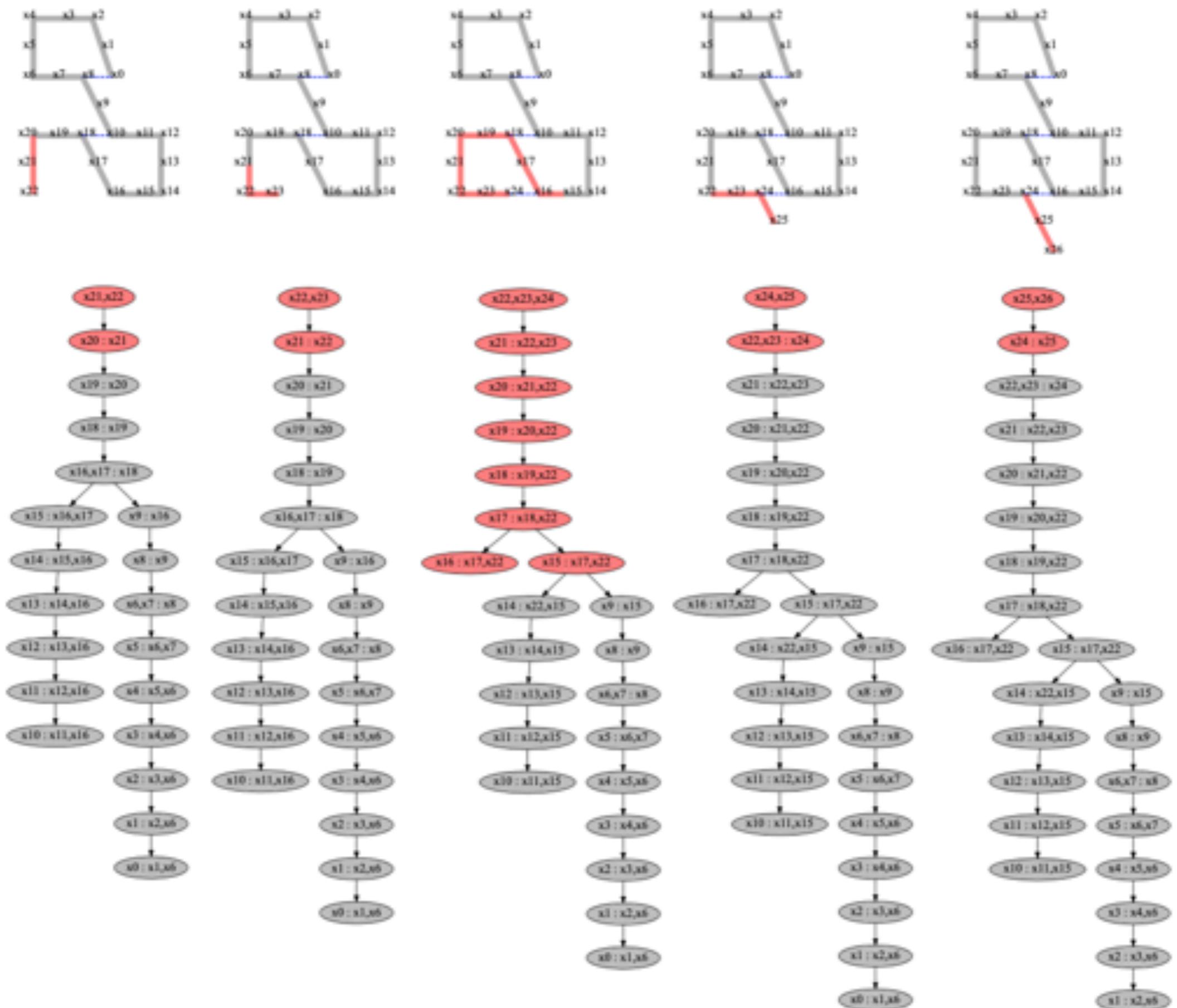
Exploit the fact that the square root information matrix can be understood as a directed junction tree: **the Bayes tree**



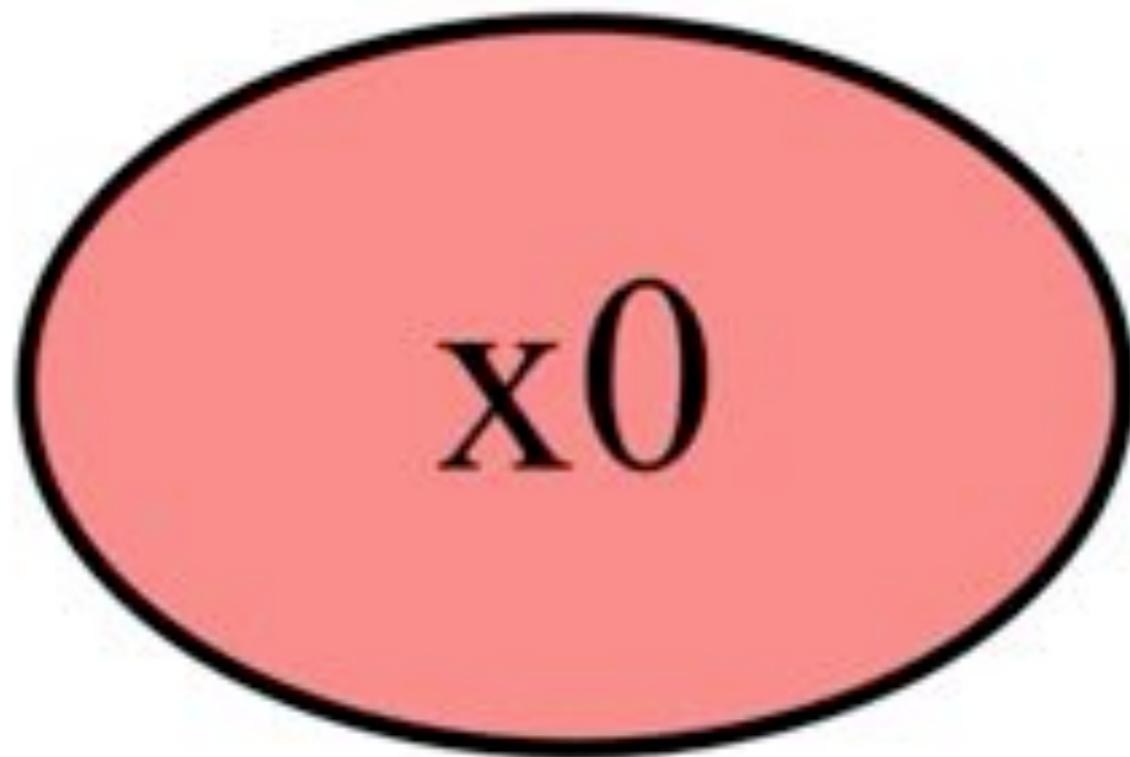
=



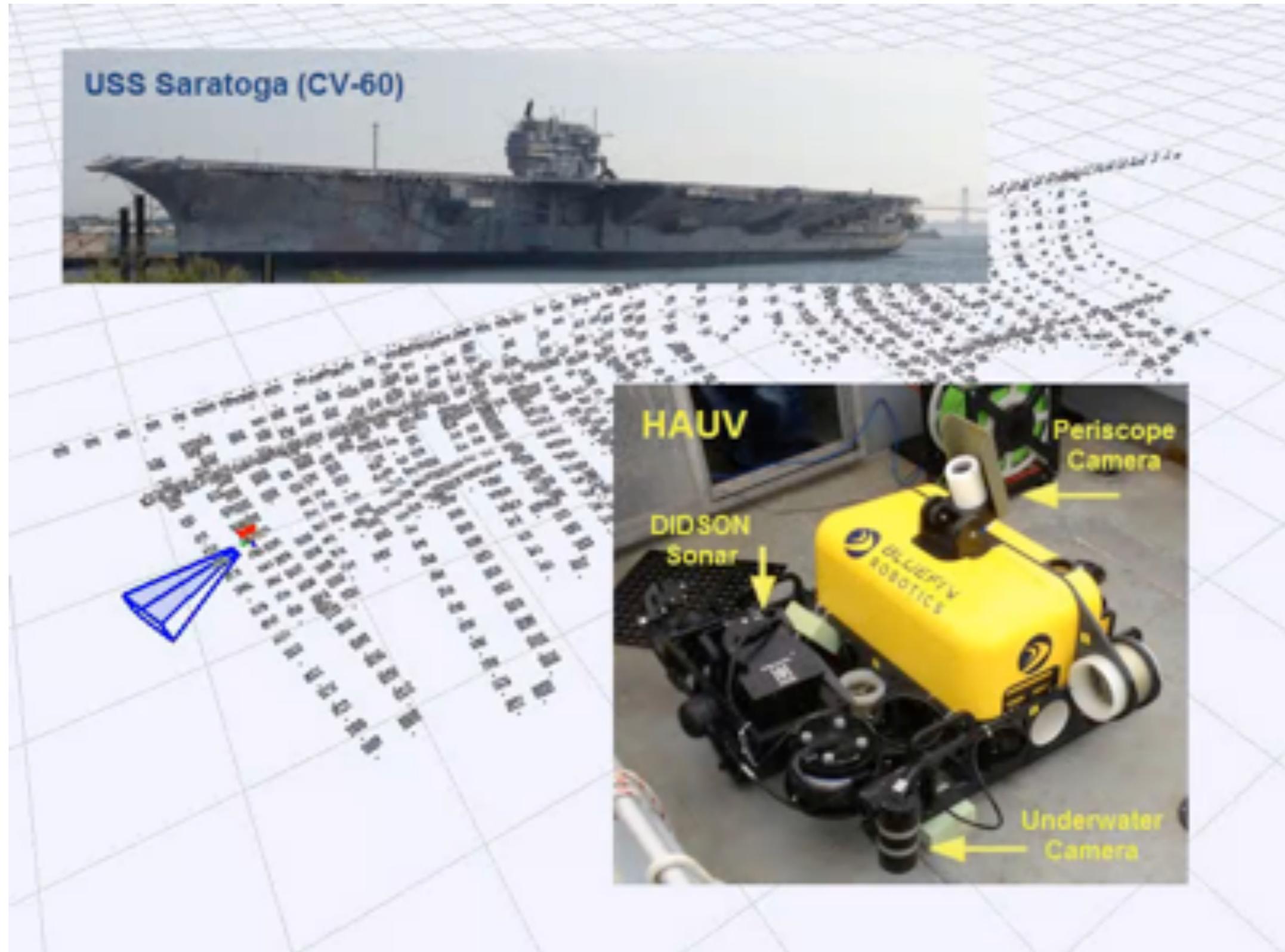
# iSAM edits a Bayes tree as new measurements arrive



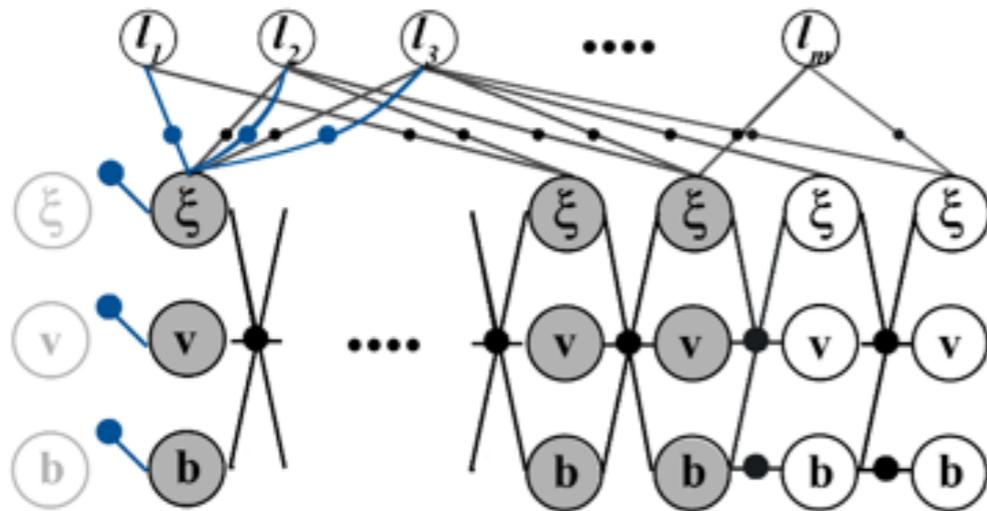
iSAM2 at work on a synthetic sequence really shows off the reduction in amortized costs afforded by the Bayes tree



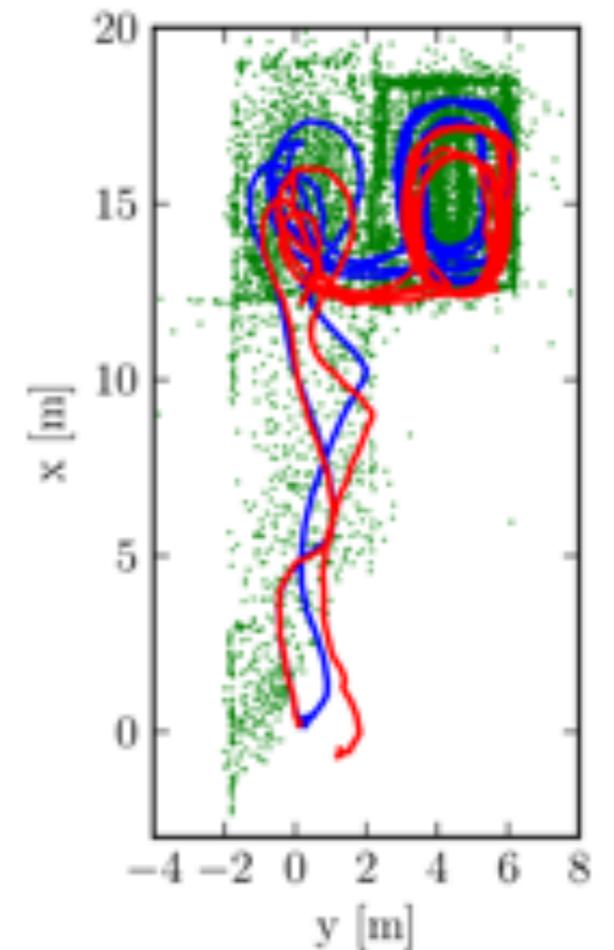
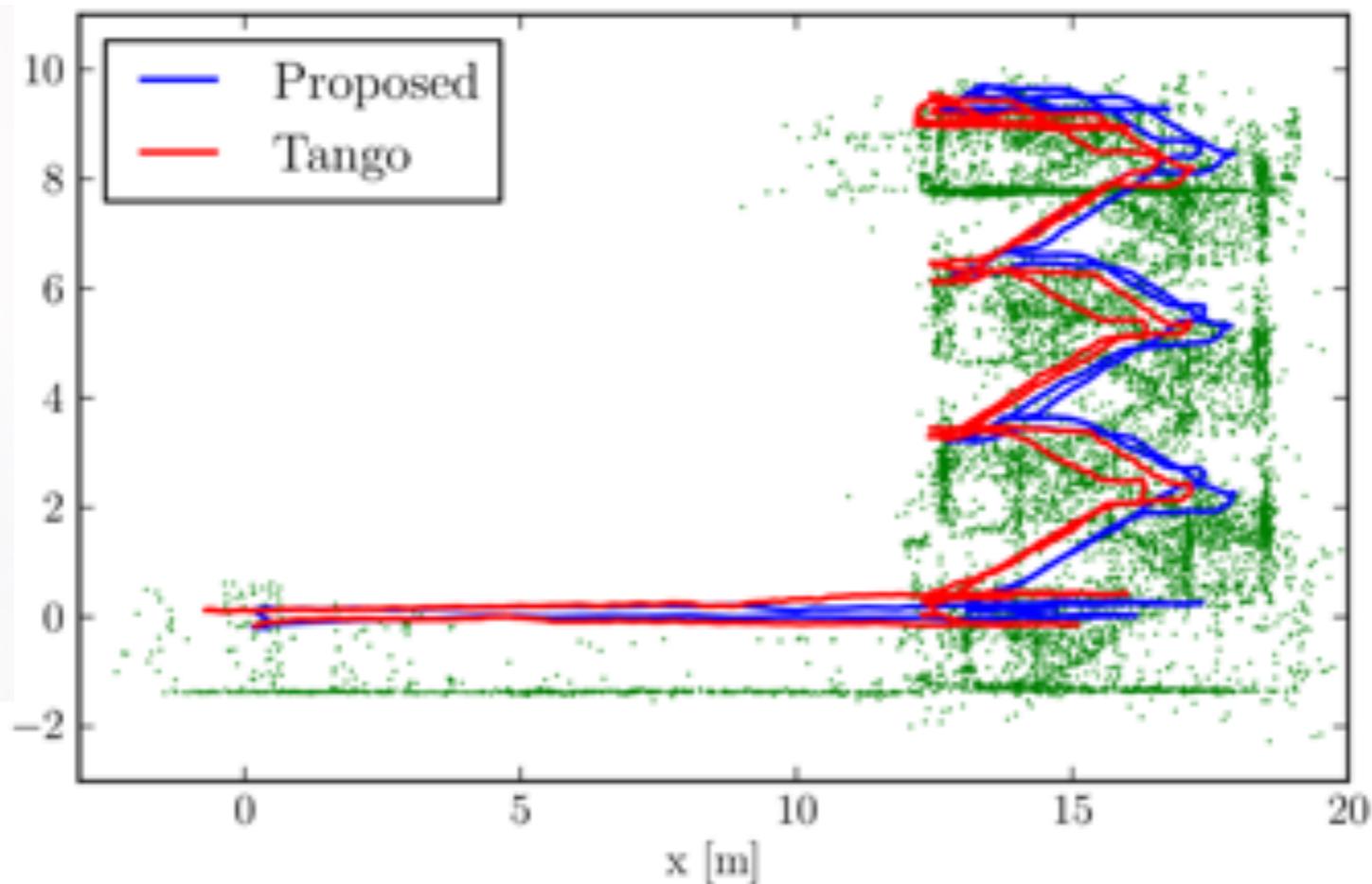
iSAM has been applied in many applications, from mapping aircraft carriers [Kim et al 2013] to experiments on the ISS



# Pre-integrating IMU measurements yields state of the art visual-inertial navigation [Forster et al. TRO'17]

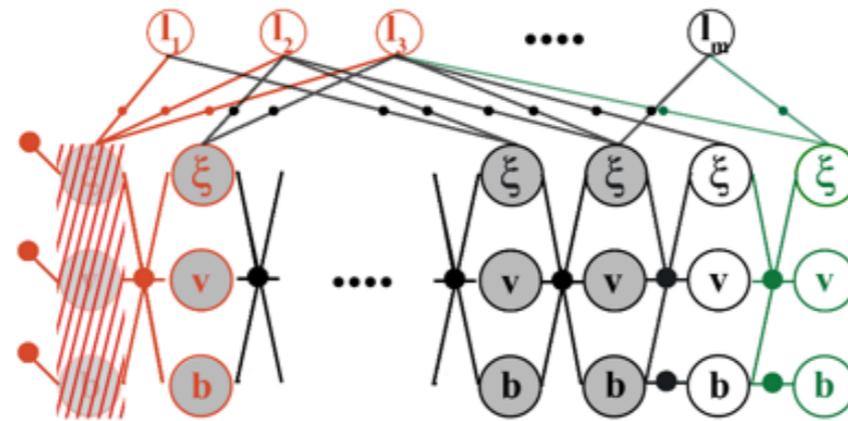


- VIO pre-integrated IMU
- Integrates IMU measurements between poses, subtracting gravity
- Efficient and accurate!

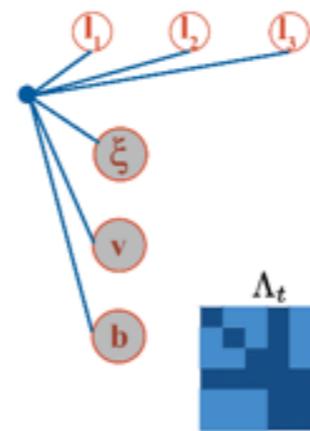
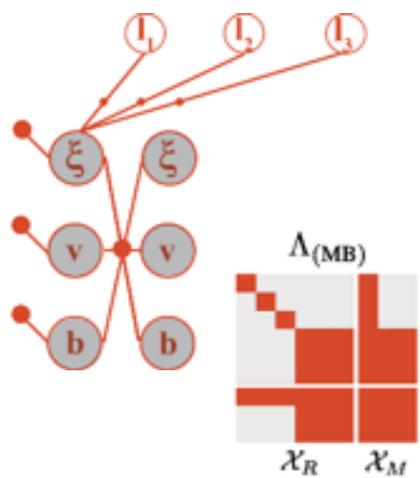


# Sparsification in visual-inertial navigation strikes a perfect balance between efficiency and accuracy

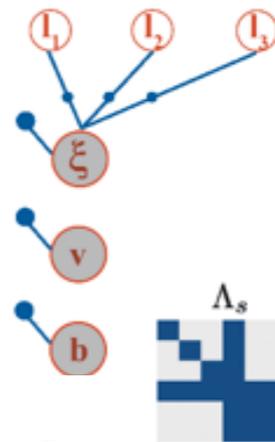
1. Extract Markov Blanket



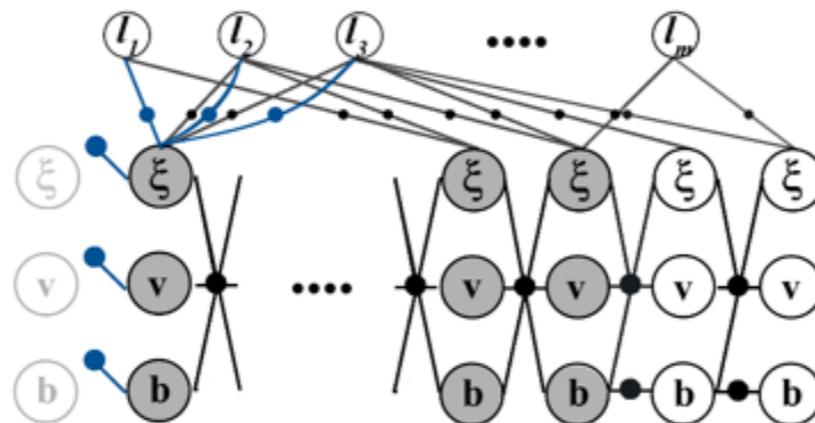
2. Marginalize

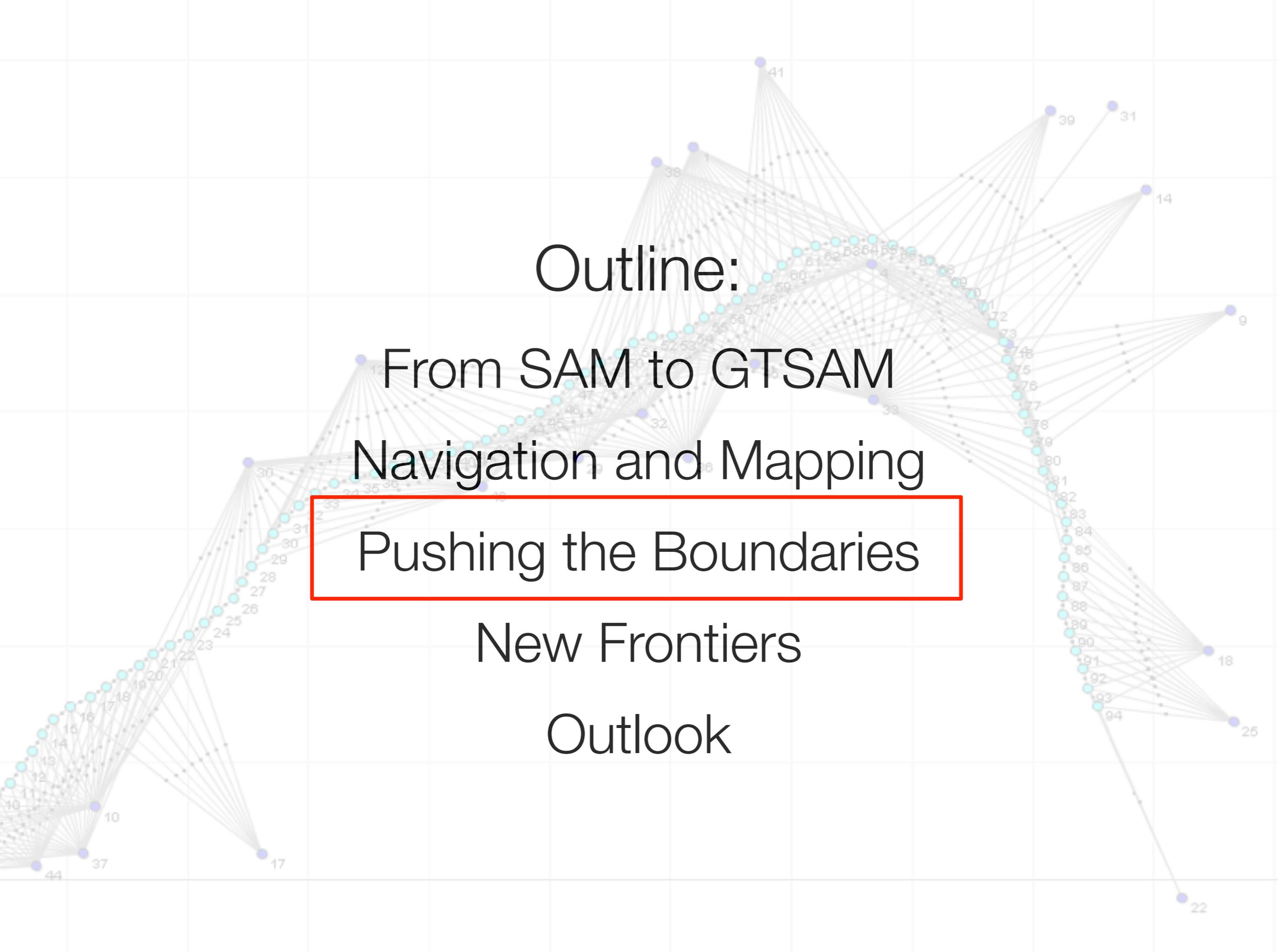


3. Sparsify



4. Reinsert





Outline:

From SAM to GTSAM

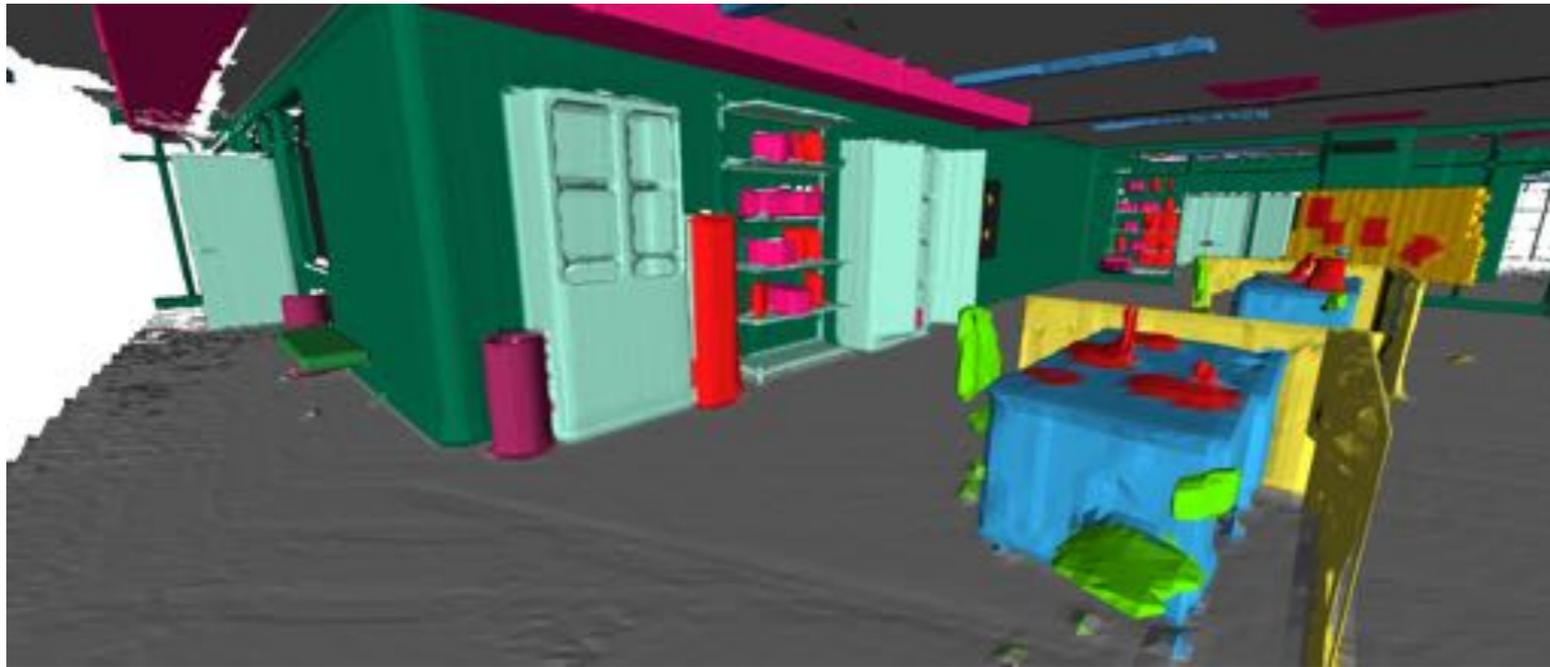
Navigation and Mapping

Pushing the Boundaries

New Frontiers

Outlook

MIT's Kimera is a state of the art metric-semantic SLAM built upon factor graphs and GTSAM [Rosinol et al. ICRA '20]

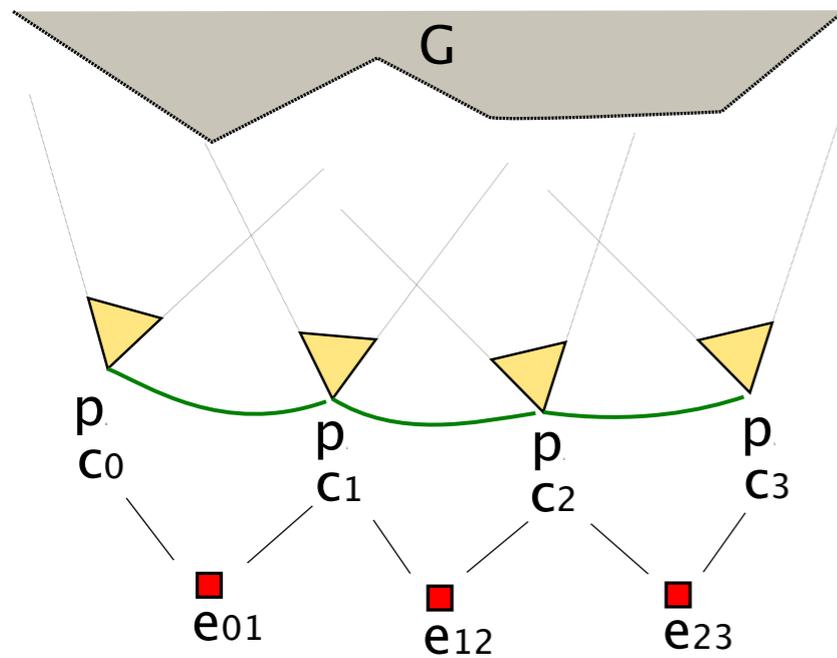


- Four modules:
  - VIO pre-integrated IMU
  - Robust factor-graph-based pose graph
  - Real-time meshing module
  - Semantics module fuses semantic 2D information into the 3D mesh representation.

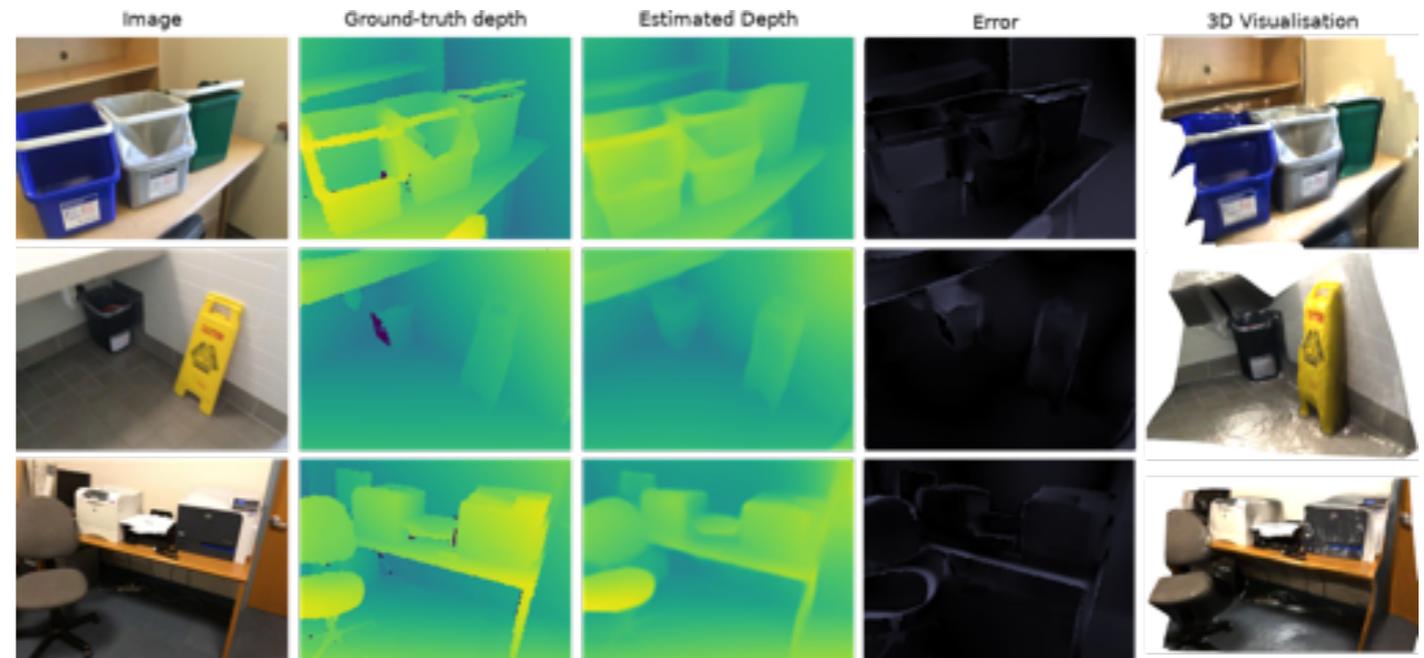
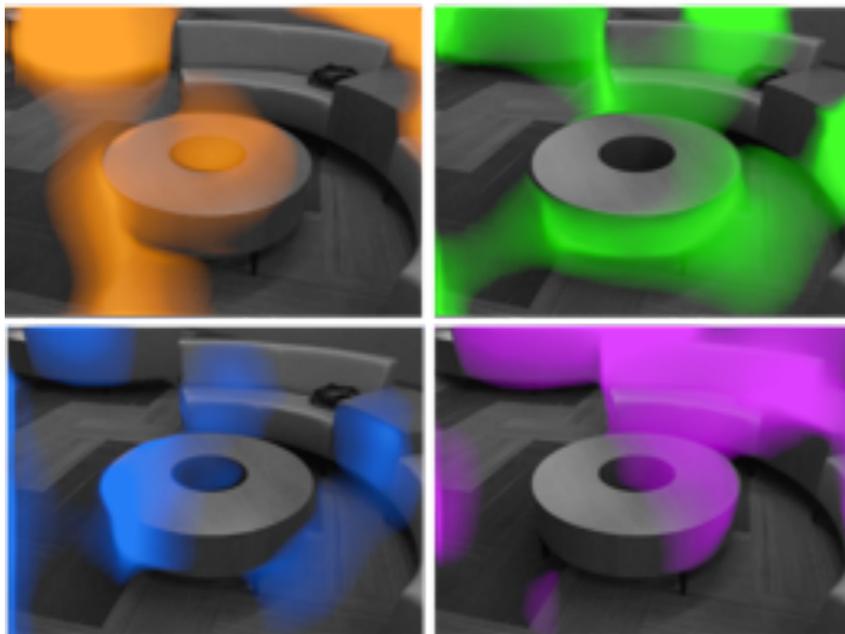
Rosinol et al. will present the impressive Dynamic Scene Graphs here at RSS, which builds upon Kimera



Czarnowski et al. [RAL '19] integrated deep VAEs into factor graphs to build a real-time, dense SLAM system

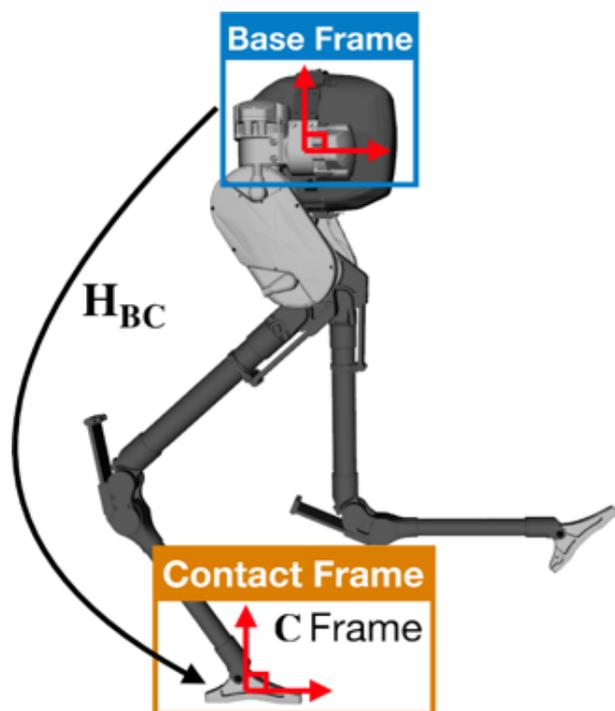
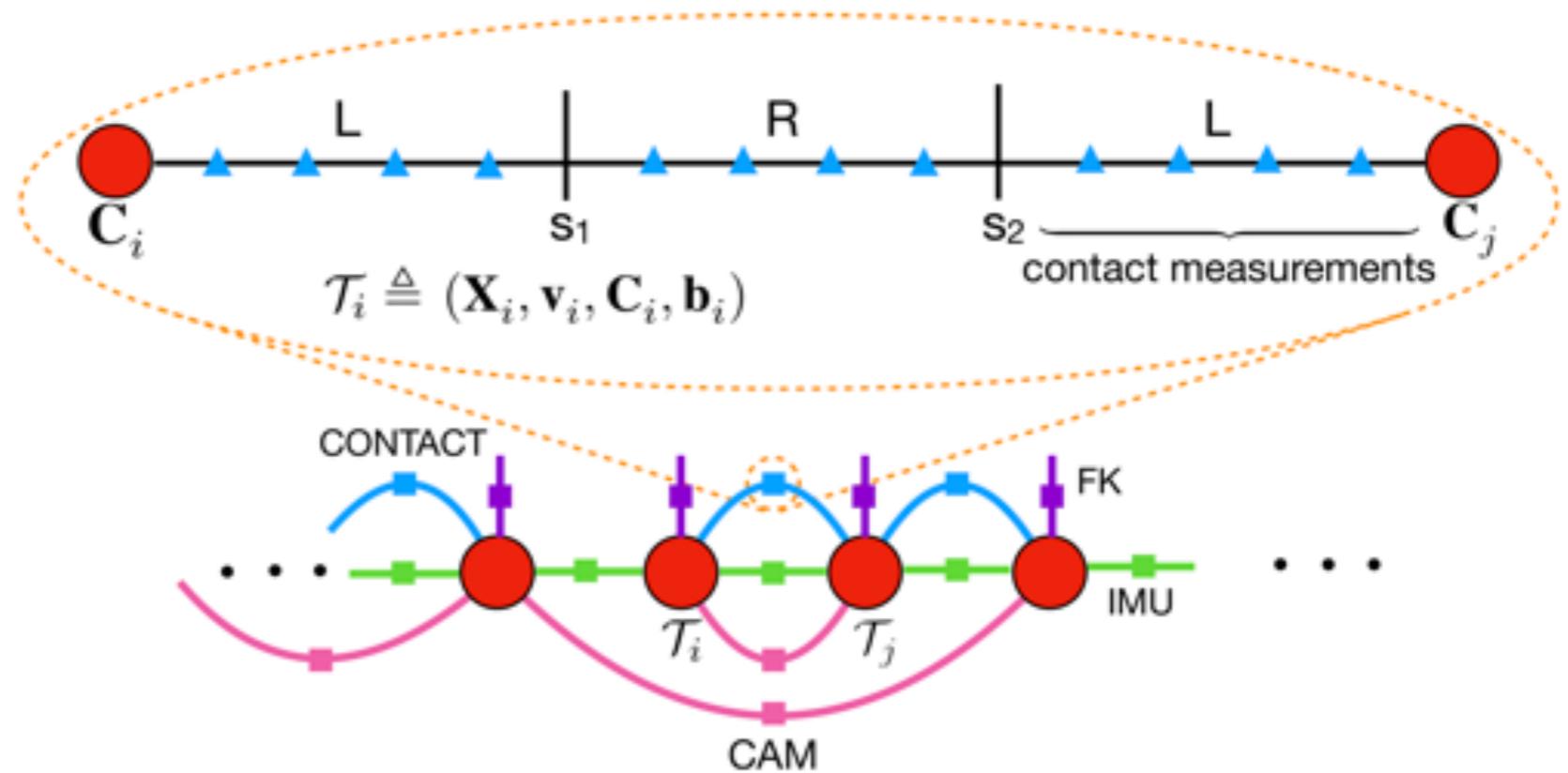


Code element



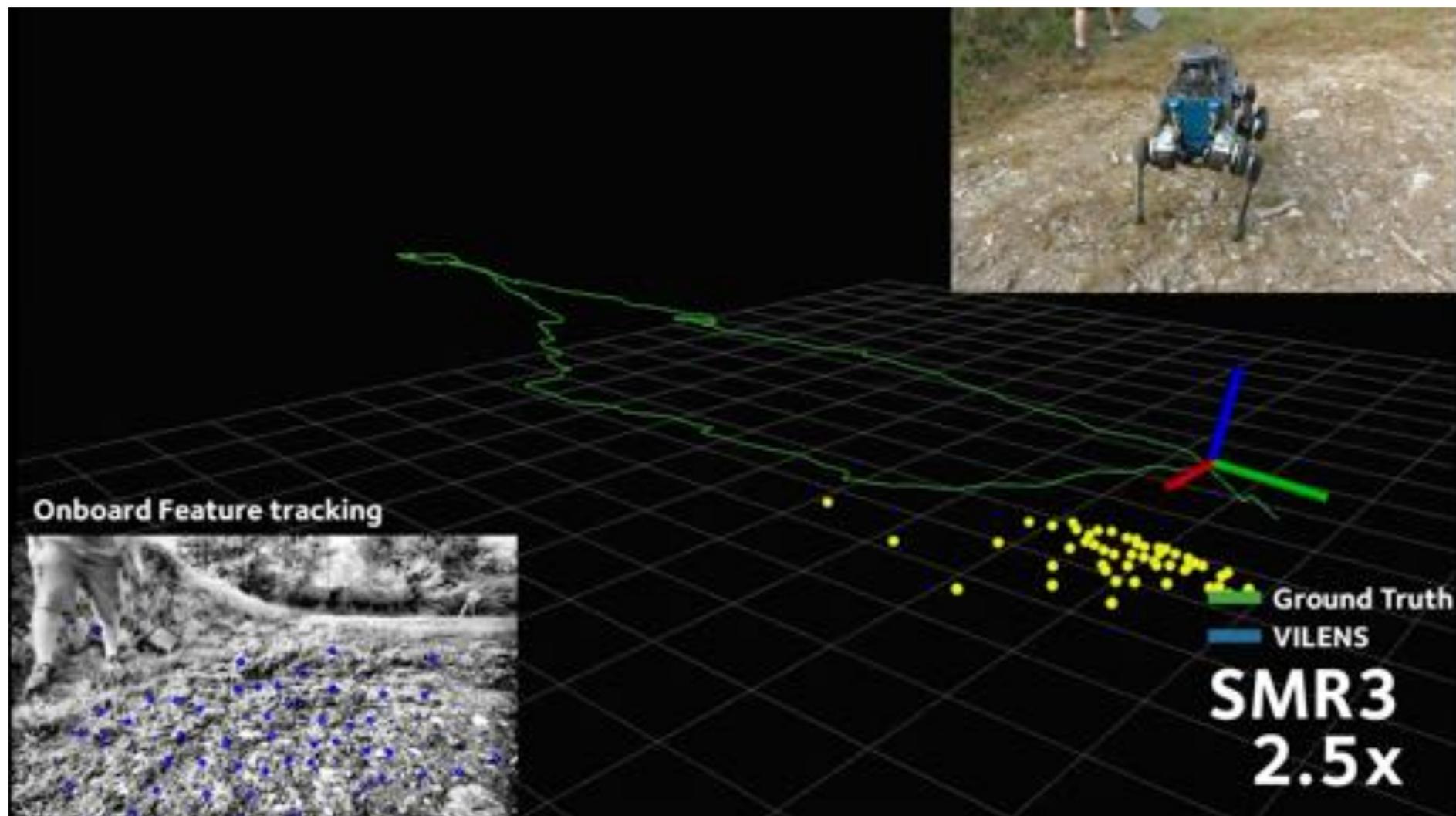
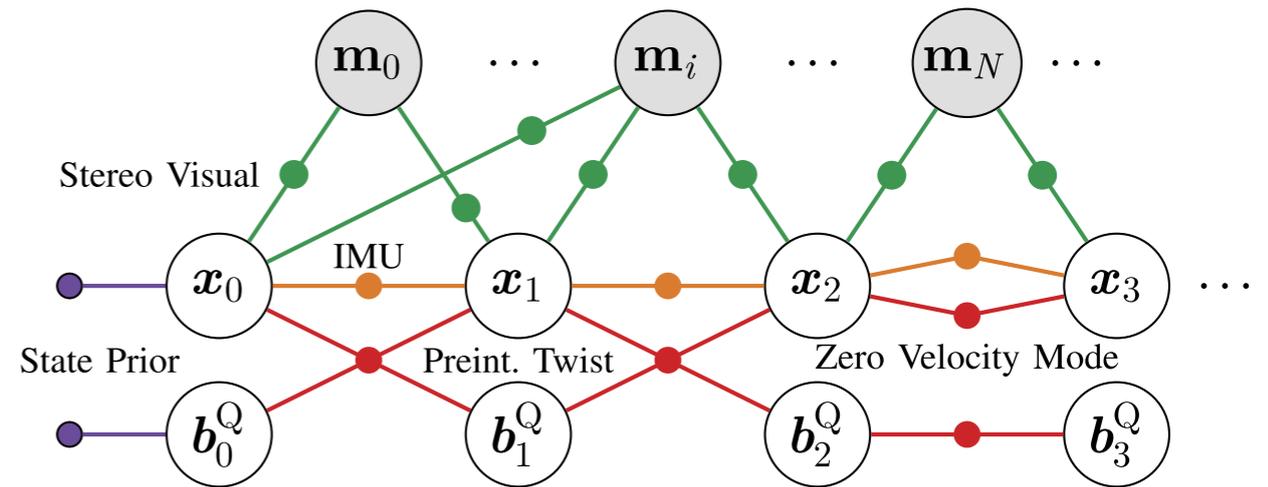
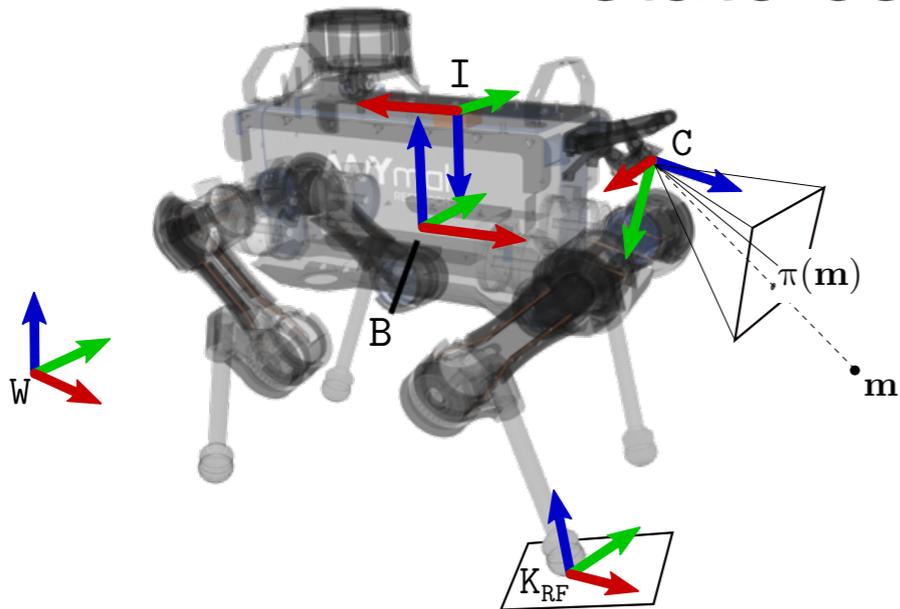
- Real-time dense SLAM system
  - Variational auto-encoder (VAE)
  - Compact "latent" codes
  - Codes are unknowns in a iSAM-based SLAM system

Factor graphs have been used in humanoid state estimation at University of Michigan... [Hartley et al. IROS '18] (1/2)

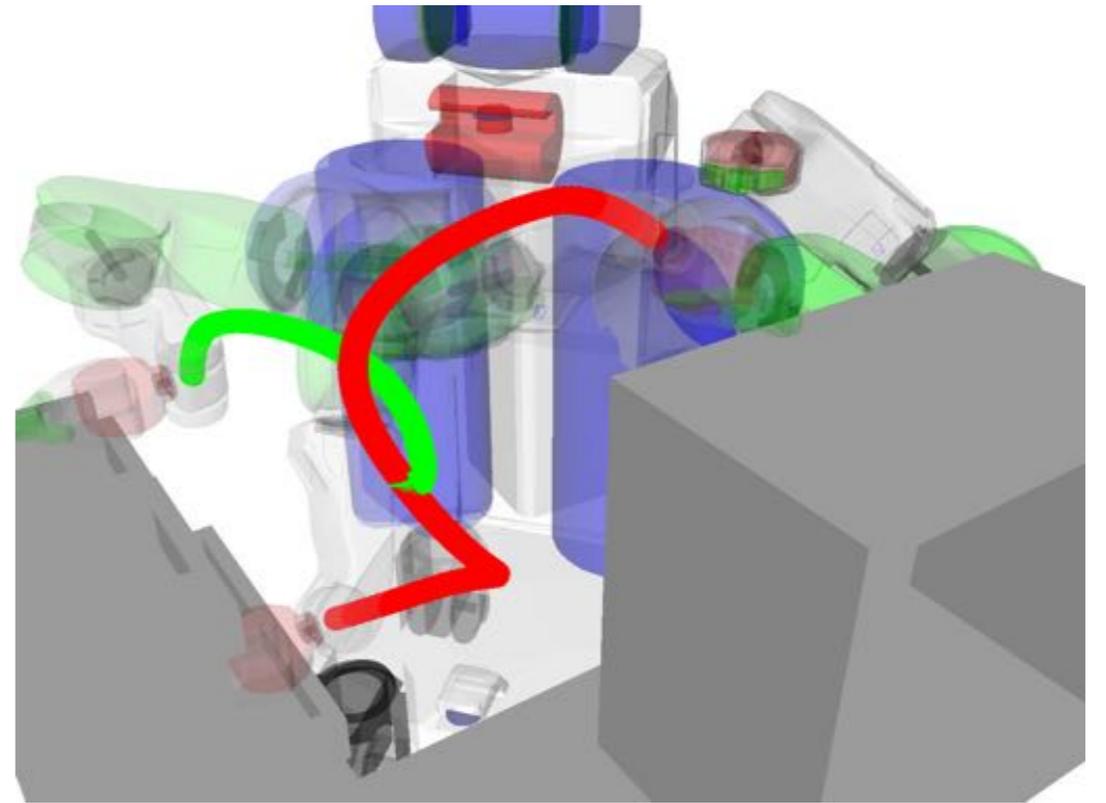
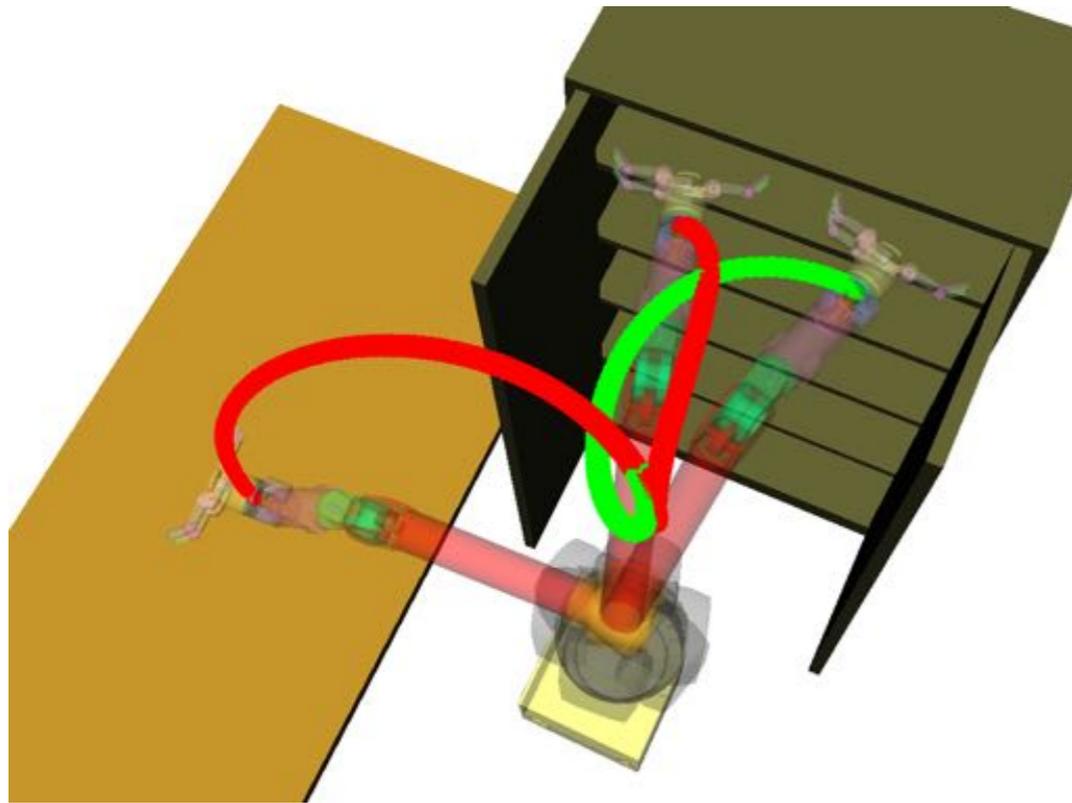


- Fuse inertial with visual and domain specific knowledge about legged robots.
- Forward kinematics (FK).
- Pre-integrated contact factors, which integrate foot contacts.

...and at Oxford for fusing visual odometry and quadruped state estimation [Wisth et al. '19-20]



Factor graphs turn out to be an excellent framework in which to innovate in motion planning [Mukadam et al. IJRR '18]



- Factors for:
  - Overall task-related objective
  - Gaussian Process motion prior factors
  - Obstacle avoidance, joint limits, etc...
- Fast incremental replanning using the Bayes Tree

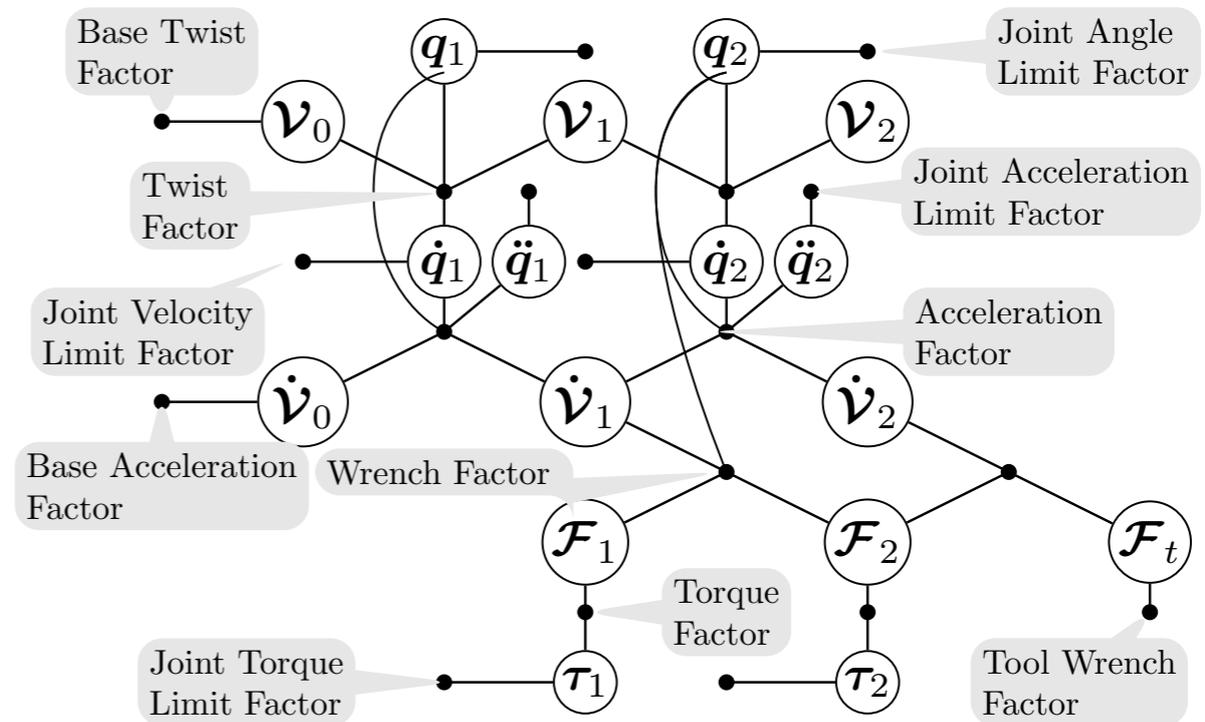
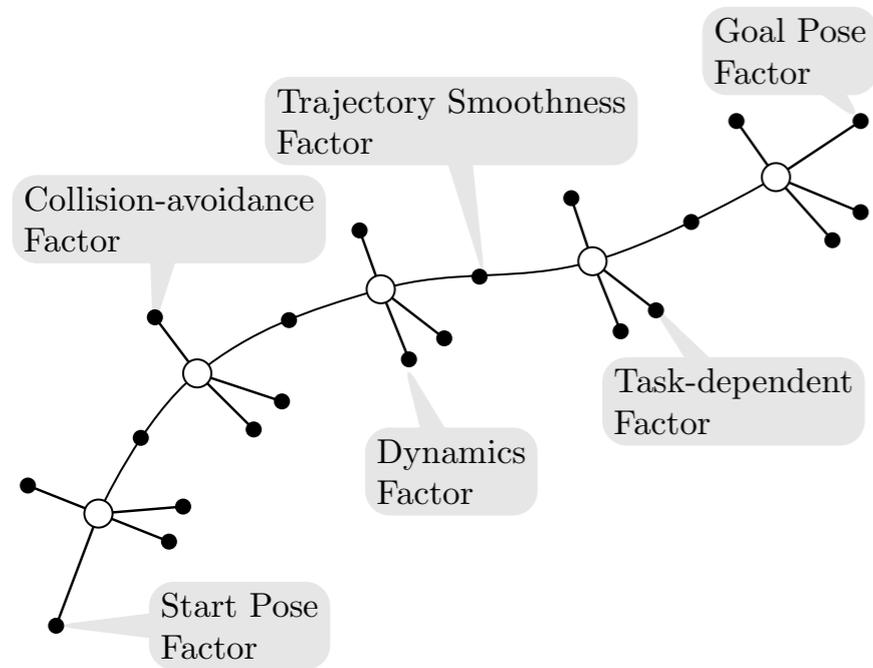
We used factor-graph-based motion planning to plan artistic action such as robot calligraphy [Wang et al. IROS '20]

空  
乱  
思  
我



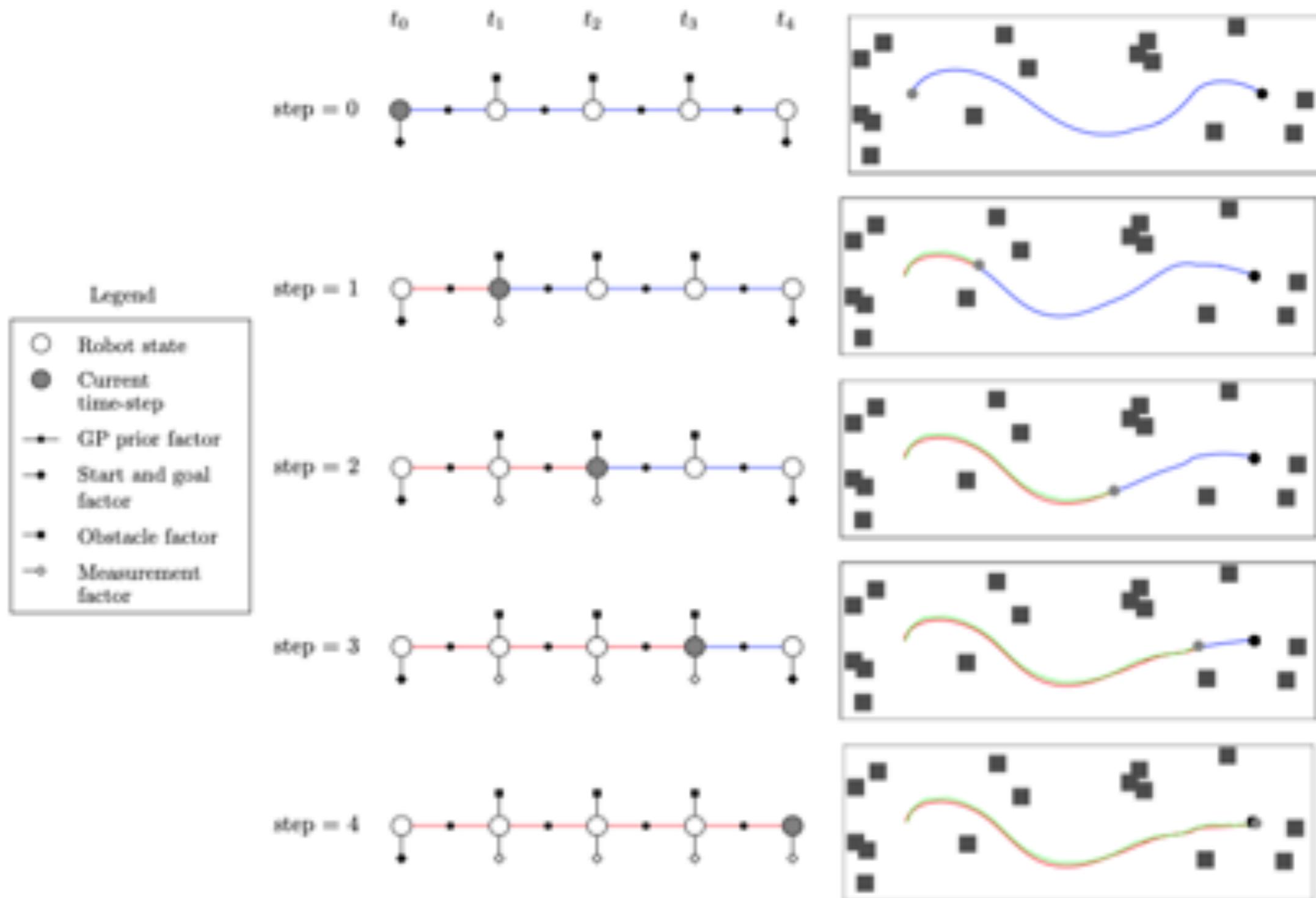
空  
乱  
思  
我

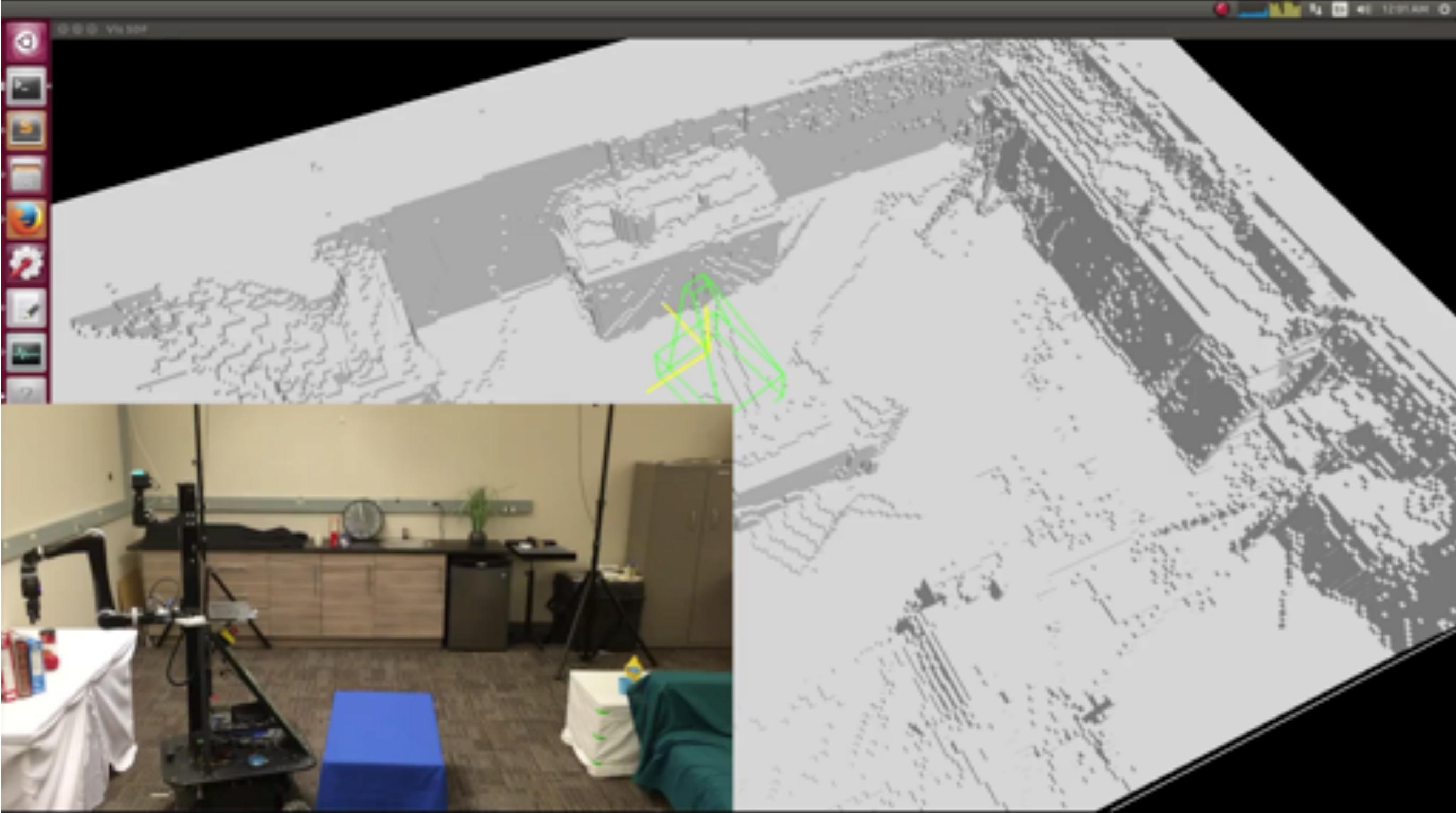
We used factor graphs to encode robot dynamics and applied to kino-dynamic motion planning [Xie et al. '20]



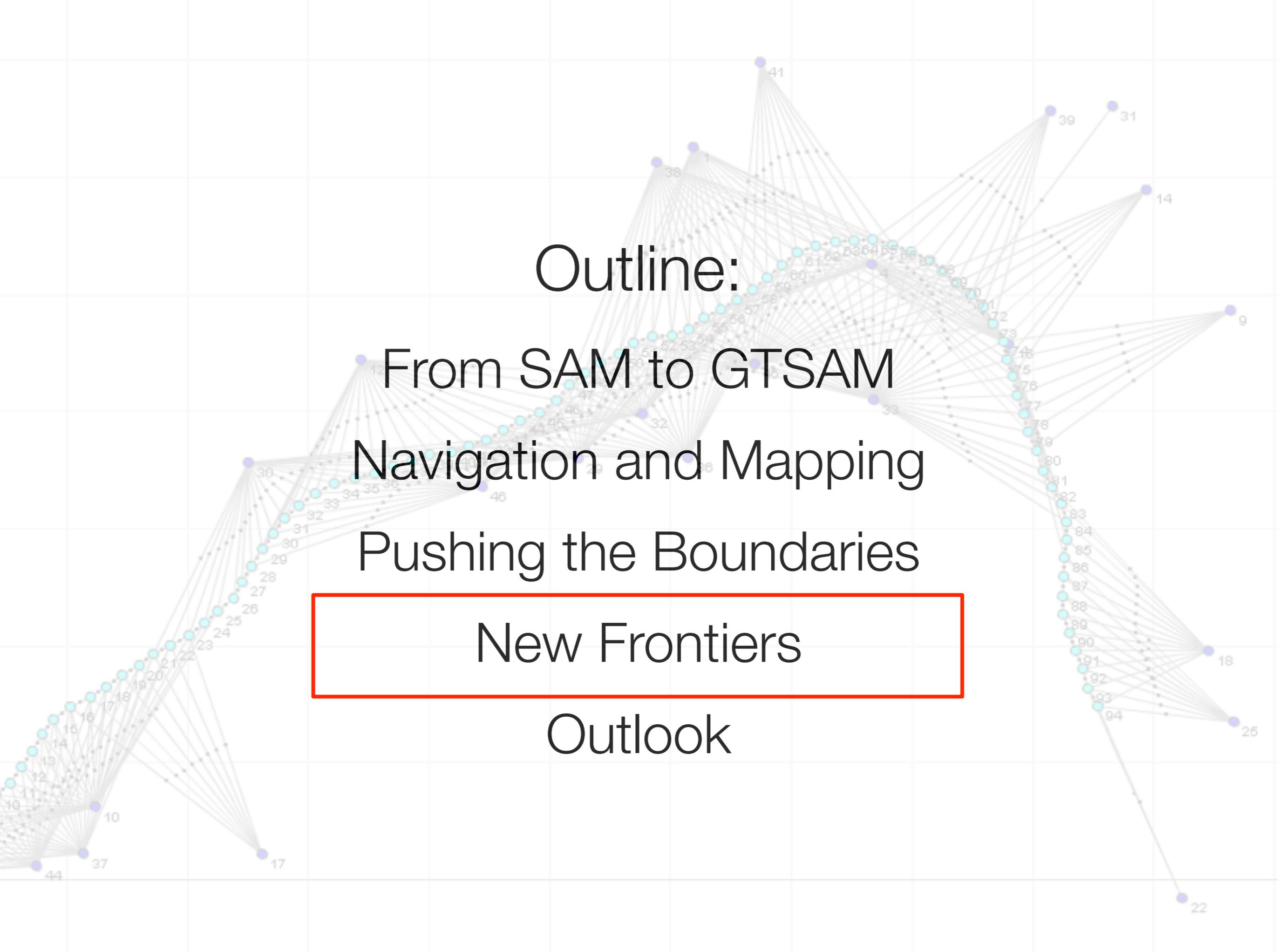
- Recipe:
  - Take Lynch & Park modern dynamics formulation
  - Turn into factor graph
  - Optimize with sparse (incremental) solvers

STEAP does both: simultaneous trajectory estimation & (motion) planning [Mukadam Auro'18]





Mustafa Mukadam, Jing Dong, Frank Dellaert & Byron Boots  
Robotics: Science and Systems, 2017, Autonomous Robotics, 2018



Outline:

From SAM to GTSAM

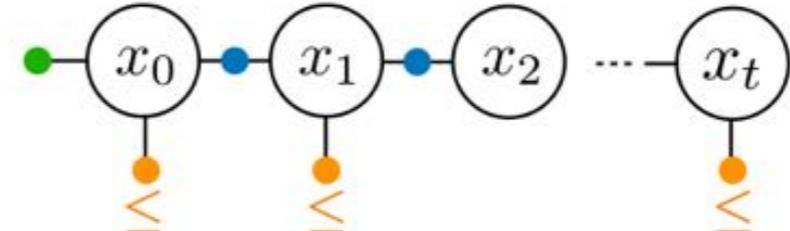
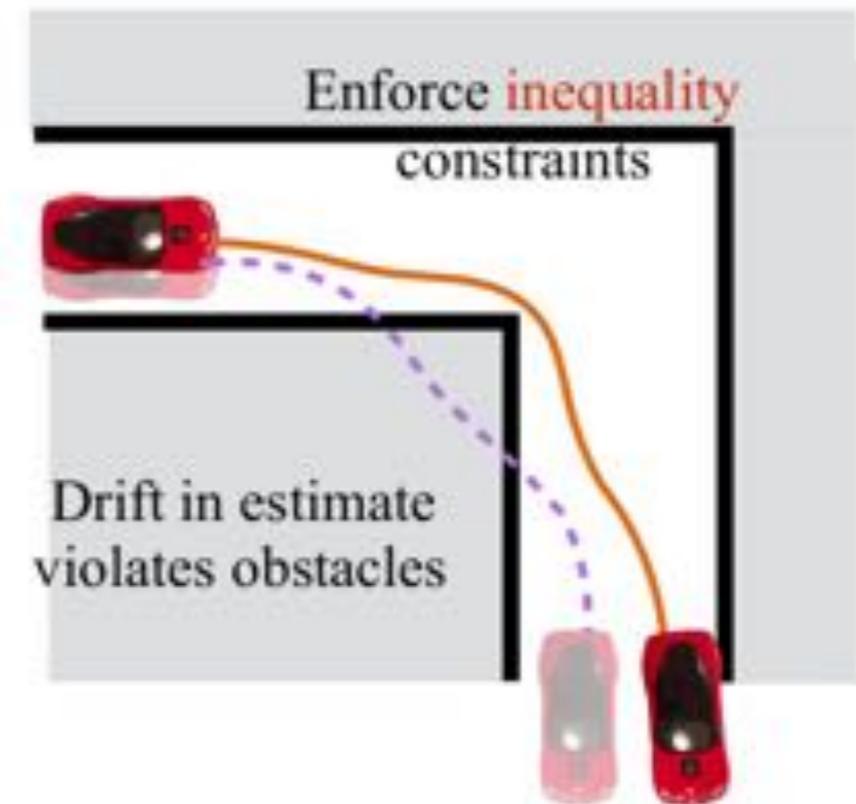
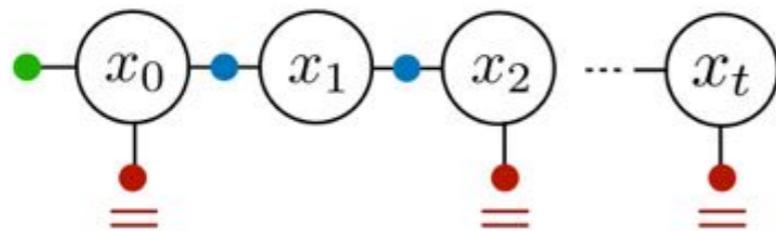
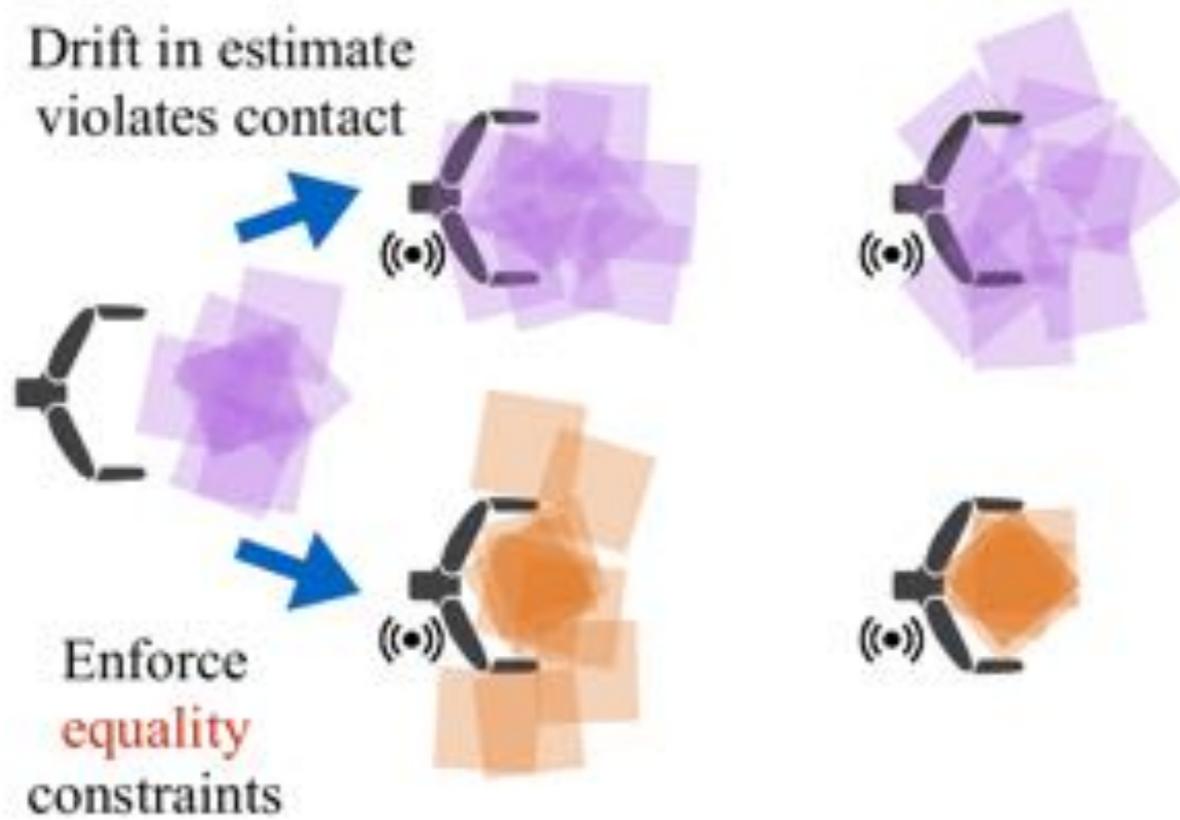
Navigation and Mapping

Pushing the Boundaries

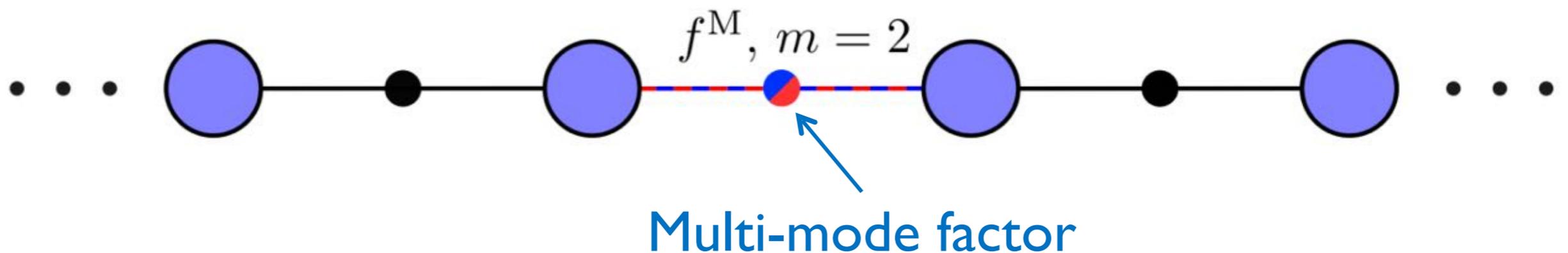
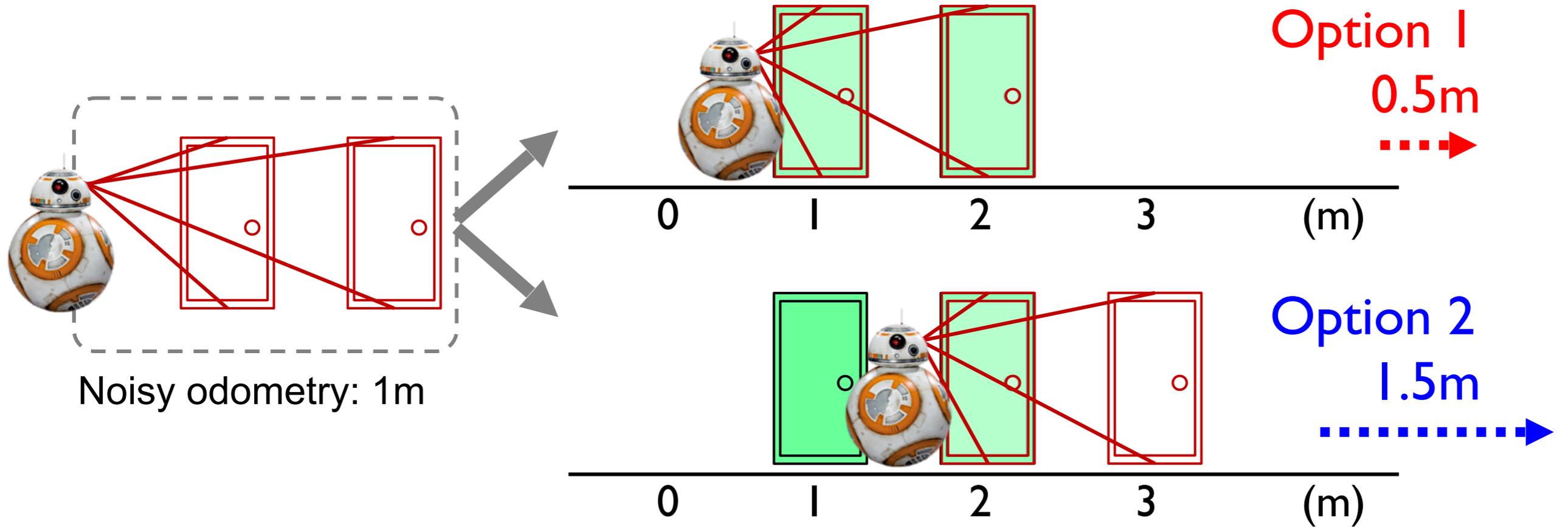
New Frontiers

Outlook

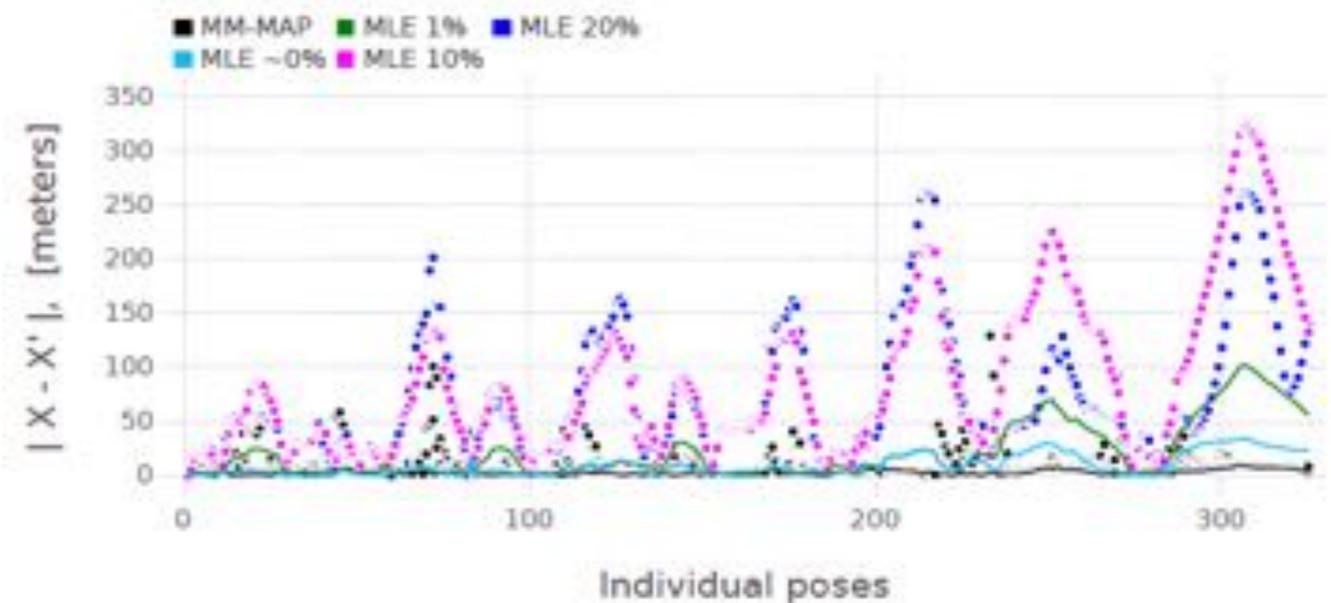
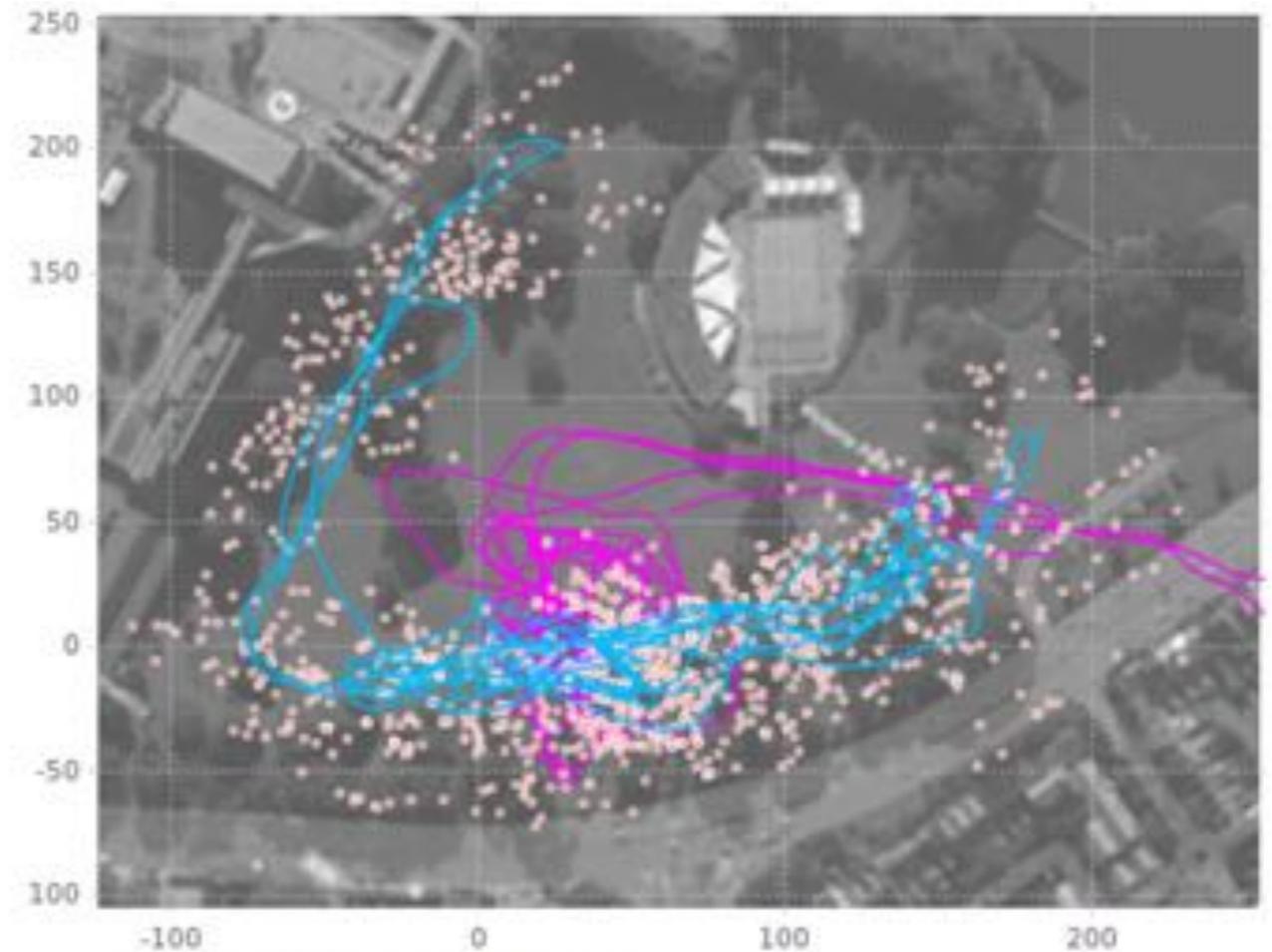
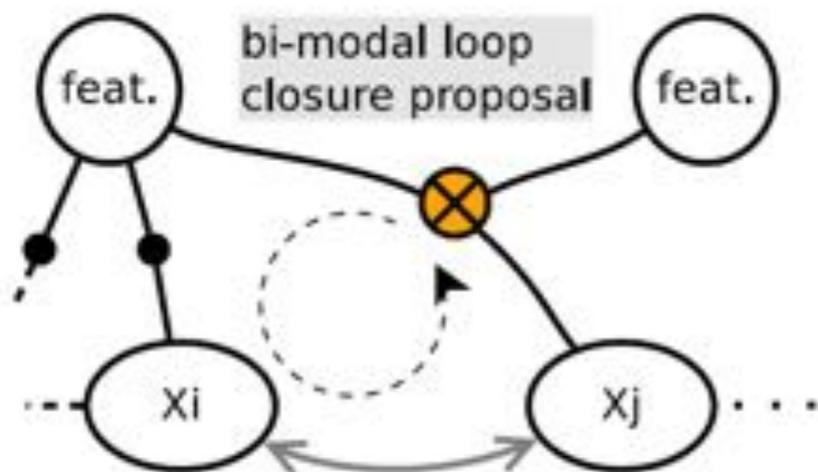
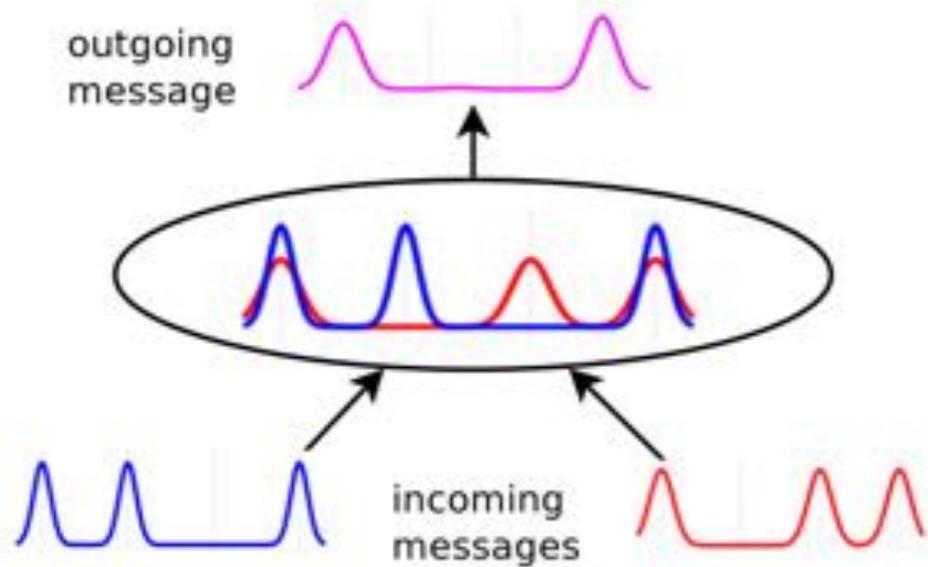
# Hard constraints for Bayes tree significantly expand iSAM2 capabilities [Sodhi et al 2020]



Factor graphs support modeling ambiguous situations,  
but what about inference?

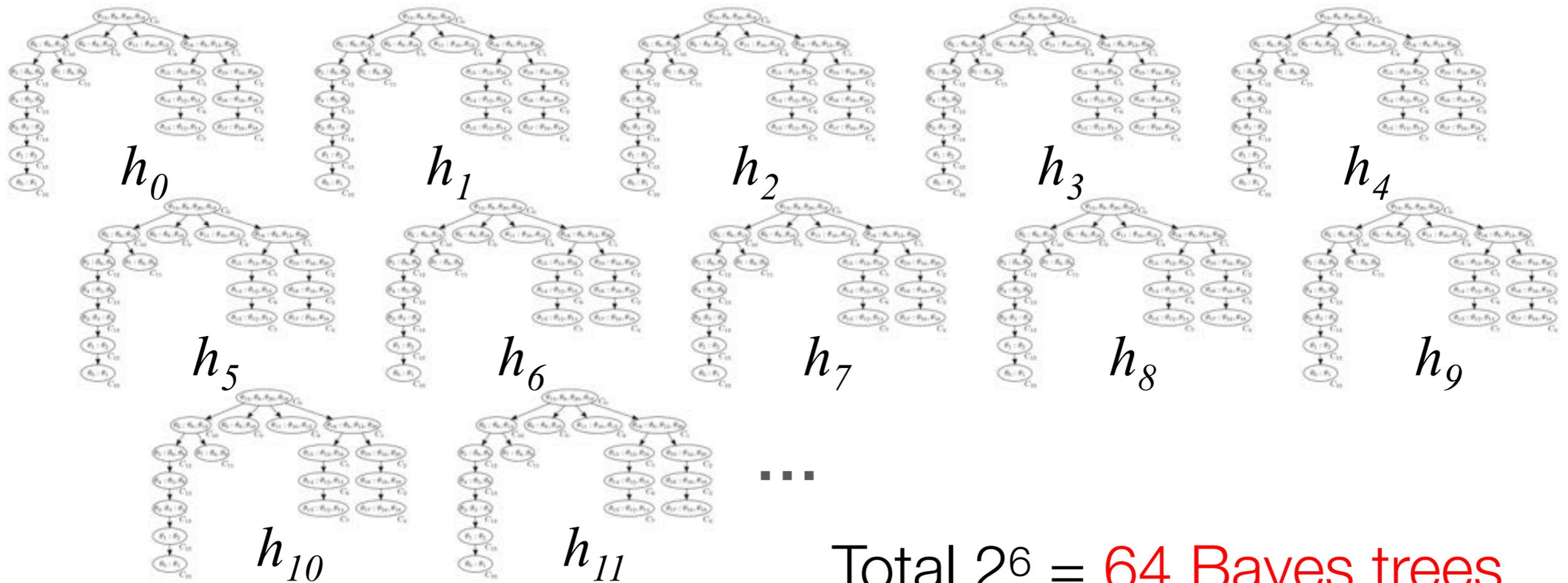
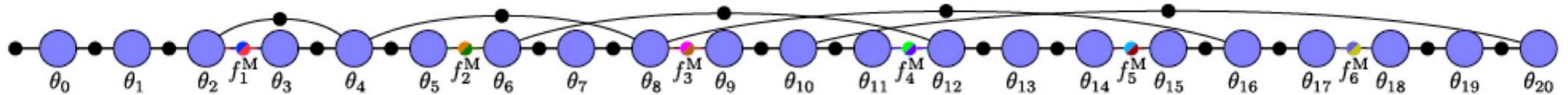


# Non-Gaussian inference using nonparametric belief propagation on the Bayes tree [Fourie et al 2016]



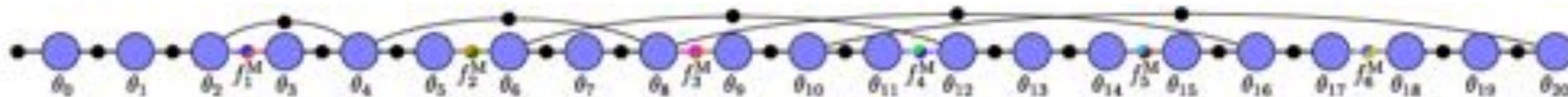
For Gaussian problems with ambiguity, multi-hypothesis tracking run multiple parallel instances of inference

- One Bayes tree per hypothesis
- Exponential growth: Pruning

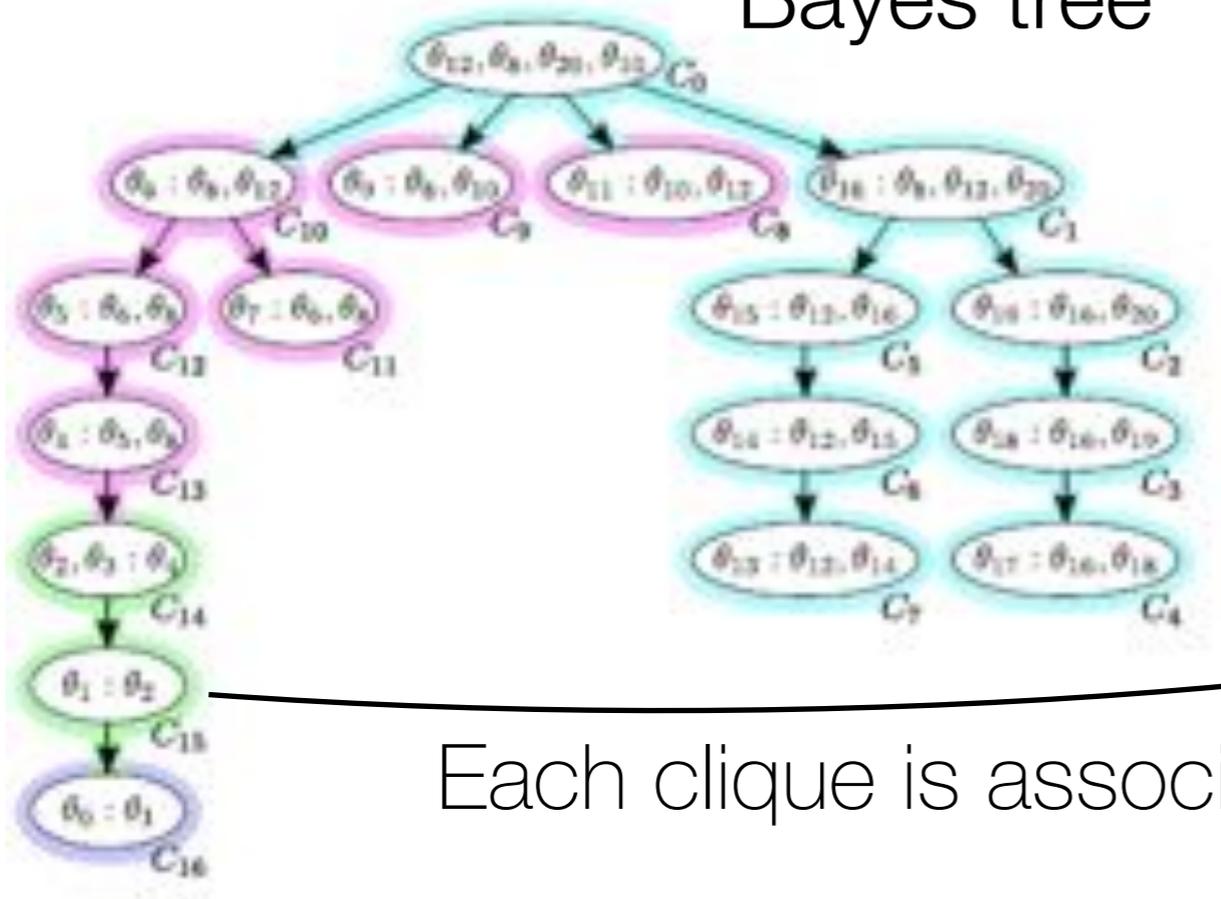


Total  $2^6 = 64$  Bayes trees

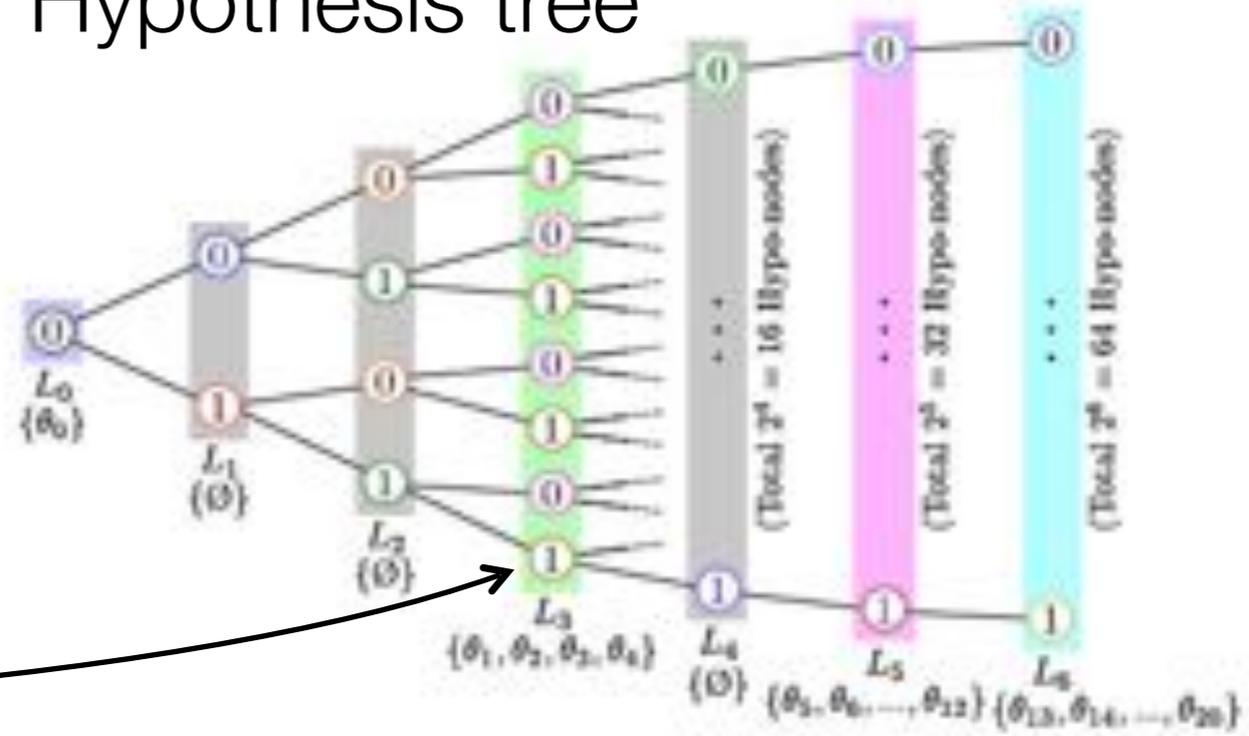
Multi-hypothesis Bayes tree saves computation by avoiding redundant computation [Hsiao et al 2019]



Bayes tree

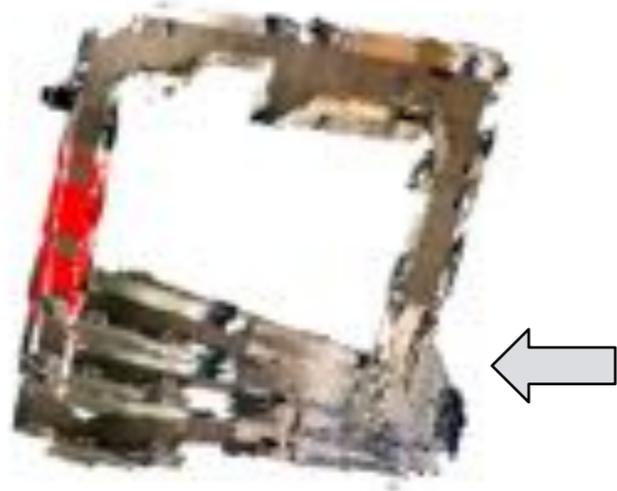


Hypothesis tree



Each clique is associated with one Hypo-layer

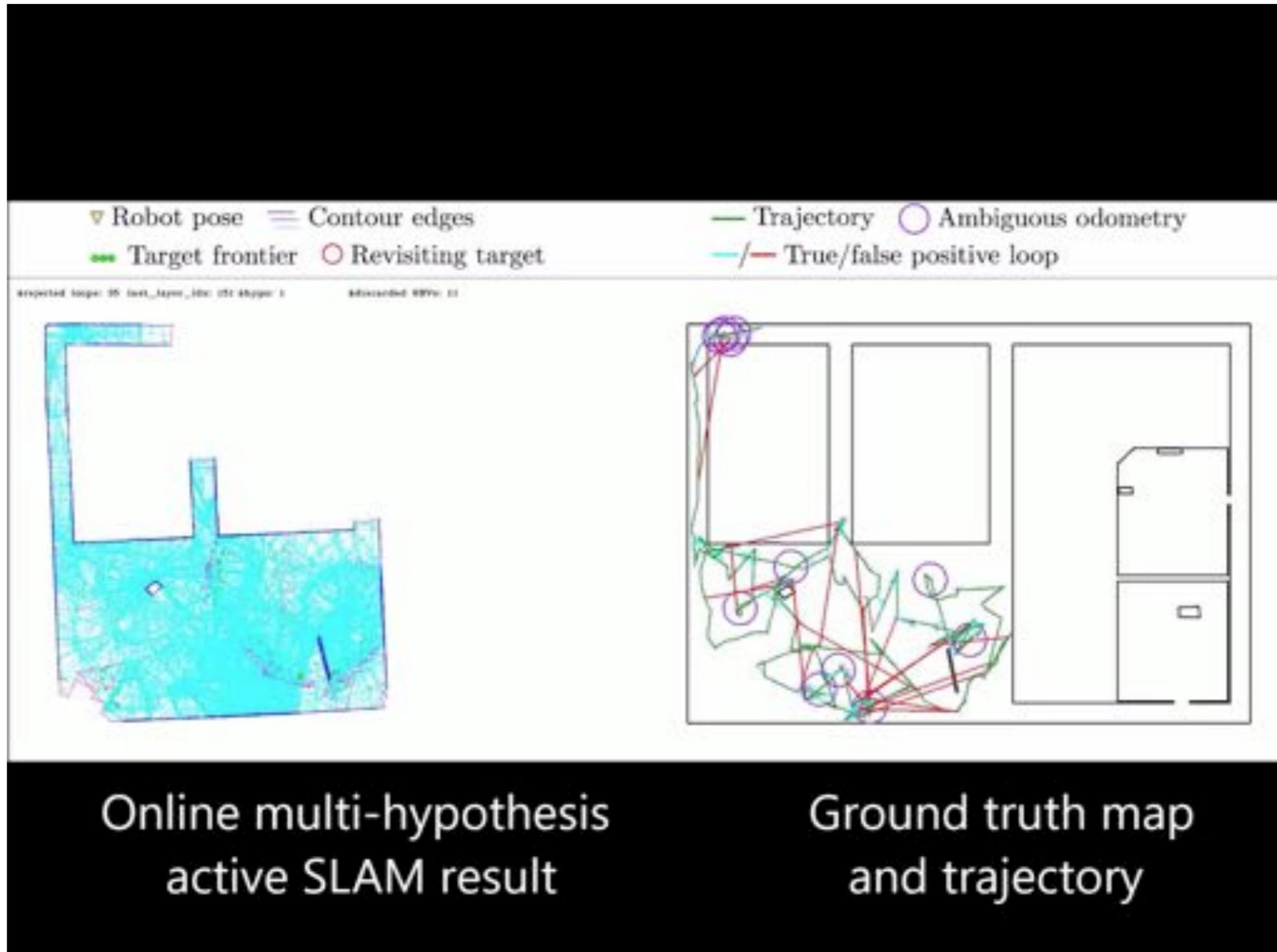
Multi-hypothesis RGB-D mapping avoids wrong decisions and can provide multiple plausible solutions



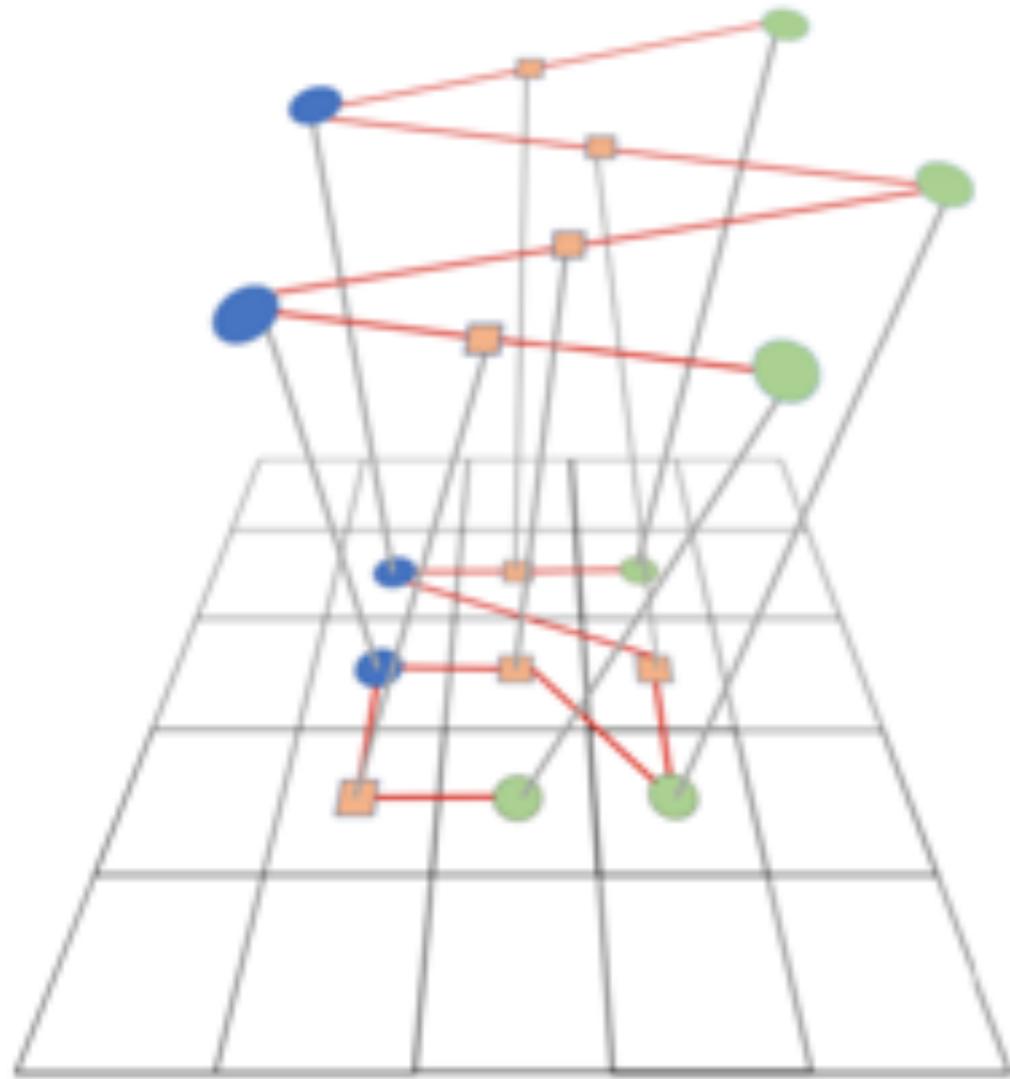
Multi-hypothesis

Single hypothesis

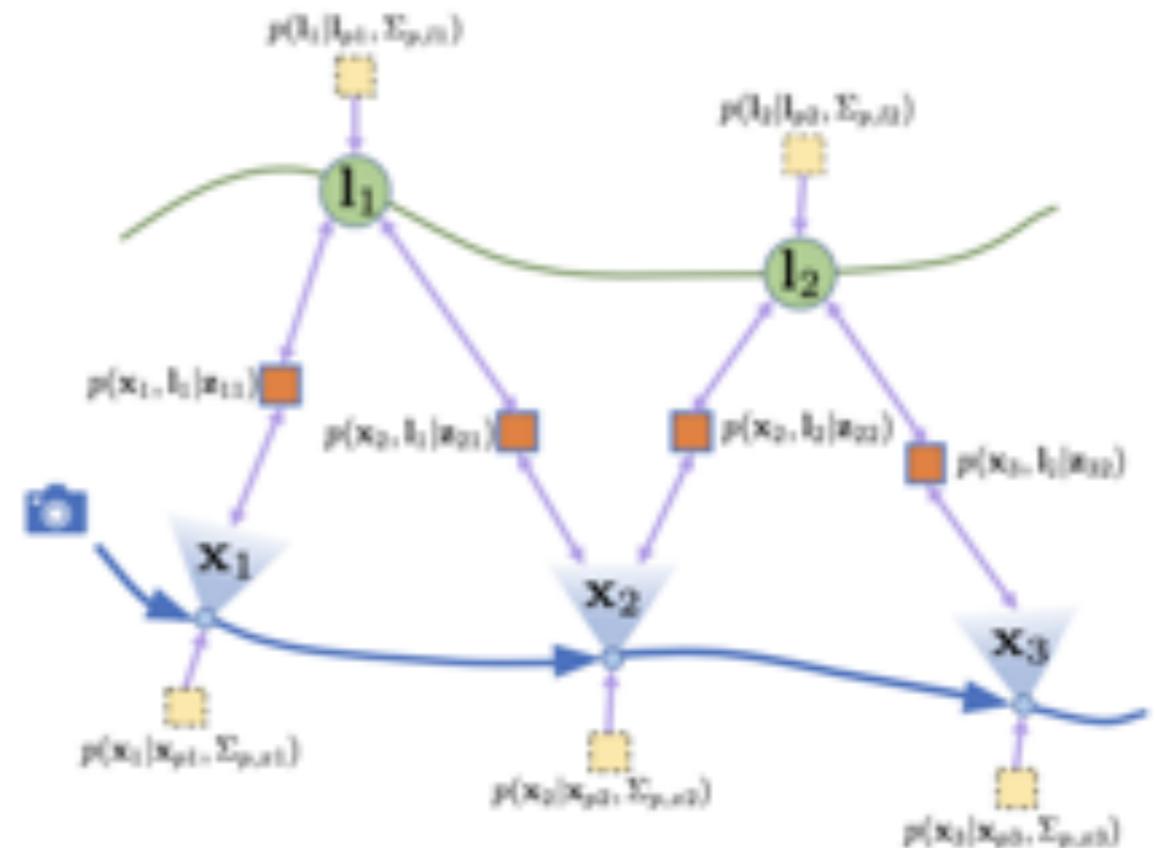
Knowledge about ambiguity is useful for planning including active SLAM [Hsiao et al 2020]



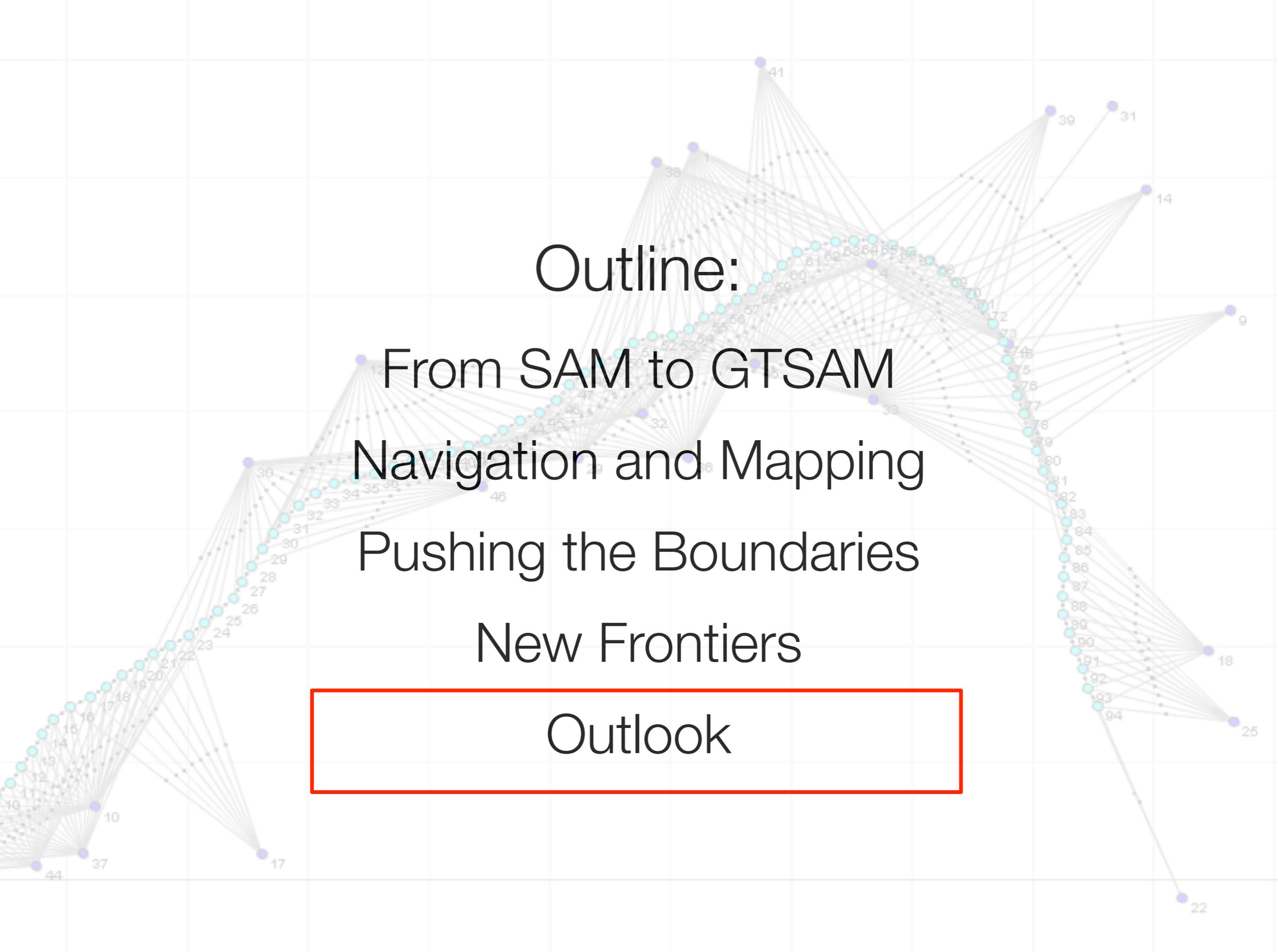
Is loopy belief propagation on factor graphs a better match to the hardware of the future [Davison et al 2020]?



**GRAPHCORE**



- Factor graphs on a graph processor
- Loopy belief propagation
- Well suited for parallel hardware
- CVPR: 30x faster SfM !



Outline:

From SAM to GTSAM

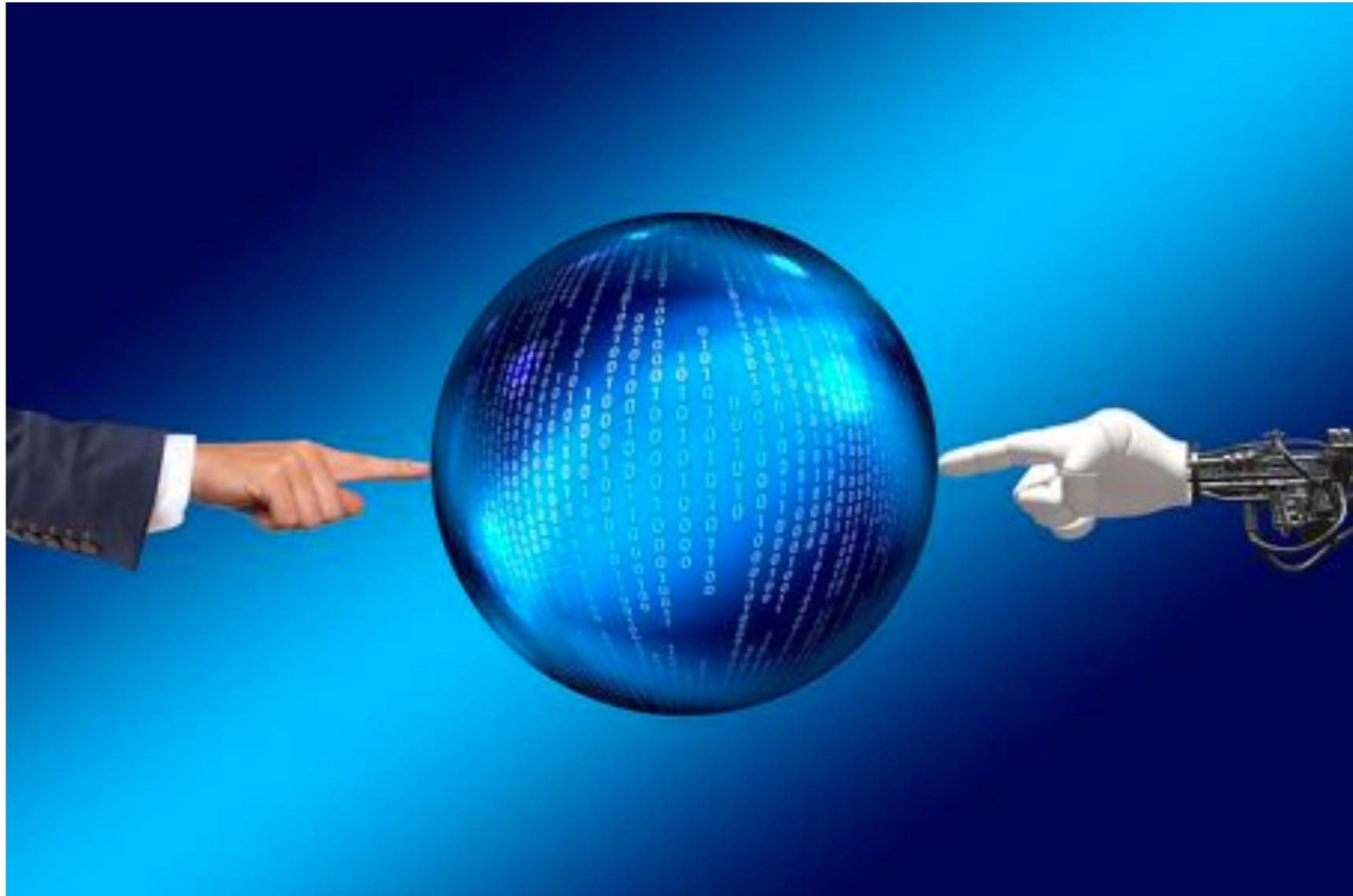
Navigation and Mapping

Pushing the Boundaries

New Frontiers

Outlook

Working on Square Root SAM 15 years ago we had no crystal ball, but we certainly imagined more robots around



The outlook for airborne autonomy is relatively positive, as the airspace environment is the easiest to conquer



- Planning state space is just 6D
- The airspace is relatively uncluttered
- Skydio has show convincing results
- Efforts by NASA/DARPA to assure safe airspace
- Crashes of light-weight drones are probably non-lethal

The timeline for self-driving cars is less clear, because of the “long tail”, bugs, and their possibly lethal consequences

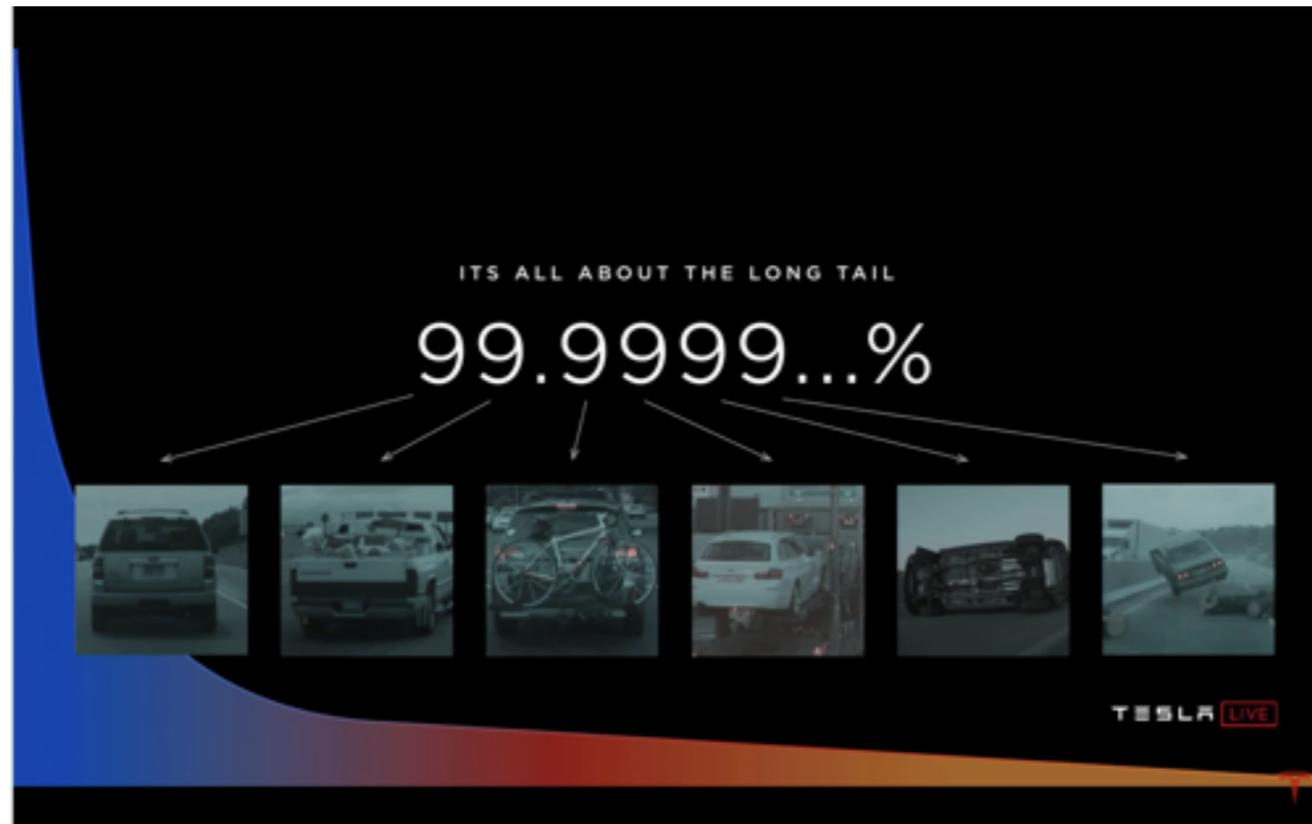
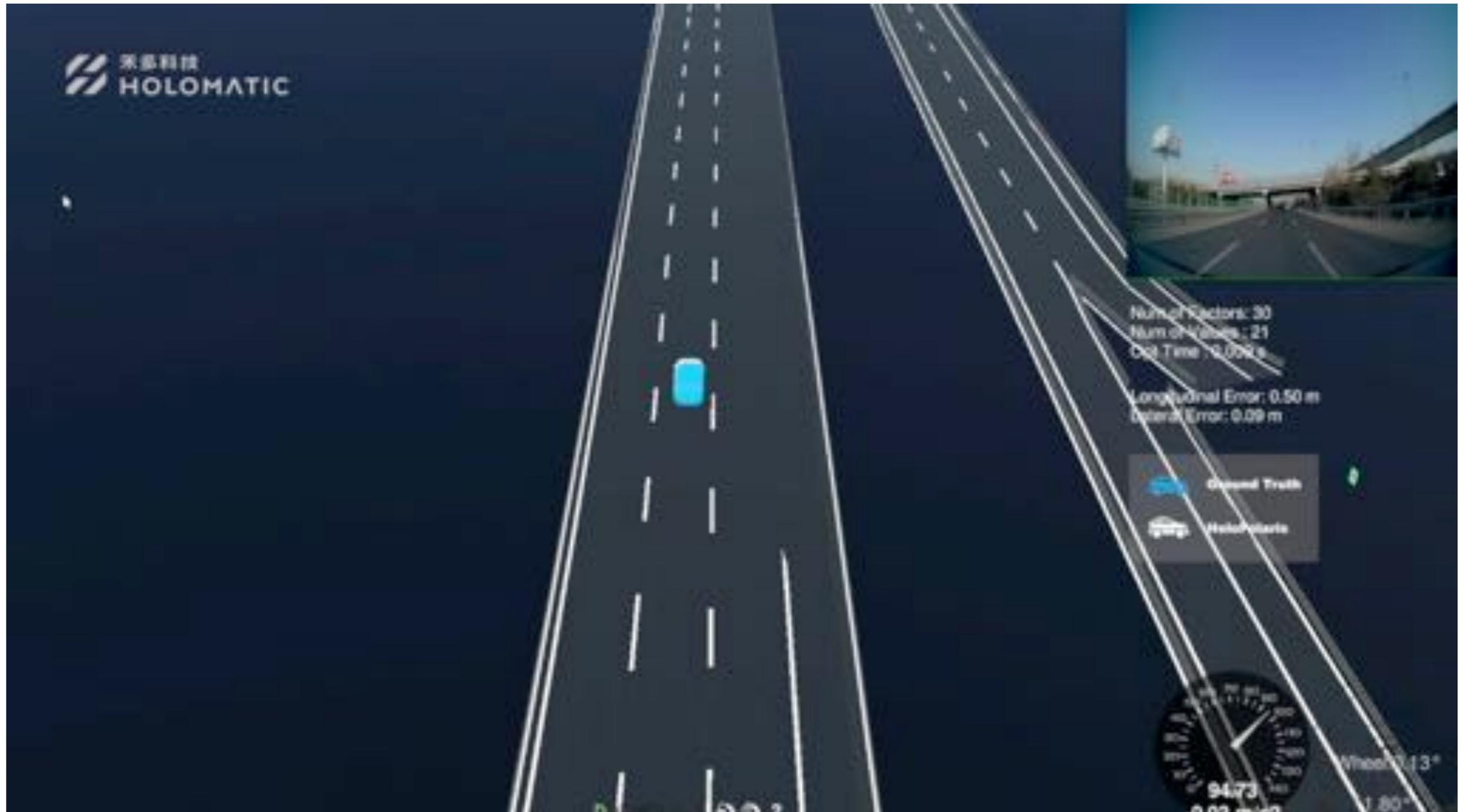


Image by Andrej Karpathy, Tesla



Factor graphs and GTSAM have been used in several autonomous driving companies, e.g., Zoox, Holomatic



We started a non-profit initiative, [OpenSAM.org](https://OpenSAM.org), to advance certifiable factor graphs for embedded applications

OpenSAM

[Get Started](#) [People](#) [News](#) [About](#)

# OpenSAM

Factor graphs for Sensor Fusion in Robotics.

[Get Started](#) >

The OpenSAM Foundation (OSF) is a non-profit organization that seeks to advance the use of factor graphs for sensor fusion in robotics and computer vision applications.

- GTSAM for back-office and Research
- OpenSAM: reference implementation for embedded systems
  - Collaboration with Holomatic and other companies...
  - Goal: fast, **certifiable** code for a subset of GTSAM functionality
  - Looking for industry memberships/collaborations!

**GTSAM is used by**

---



The most difficult environment to deploy robots in is the home, because of perception/manipulation/HRI



- Perception is very challenging due to clutter, occlusion...
- Manipulation in those environments is yet unsolved
- Expectations of people are mismatched

Our new effort, SwiftFusion, is focused on combining estimation and optimal control with the data-driven revolution

- Google collaboration: **SwiftFusion**
  - Seamless integration with TensorFlow
  - Fast, automatically differentiated factors
  - Sparse factor graphs and dense tensor processing in one language
- Which will enable:
  - Combine probabilistic estimation and optimal control with data-driven factors
- Collaborators wanted, DM me @fdellaert



<https://github.com/borglab/SwiftFusion>

All of this was only possible by amazing collaborations over the years, in academia, on github.com, and industry



# References: Navigation and Mapping

- Ni Kai, Dellaert F. 2010. Multi-Level Submap Based SLAM Using Nested Dissection. In IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)
- Kai Ni, and Frank Dellaert, HyperSfM, 2012. IEEE International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)
- Cunningham A, Indelman V, Dellaert F. 2013. DDF-SAM 2.0: Consistent Distributed Smoothing and Mapping. In IEEE Intl. Conf. on Robotics and Automation (ICRA)
- Kaess M, Johannsson H, Roberts R, Ila V, Leonard JJ, Dellaert F. 2012. iSAM2: Incremental Smoothing and Mapping Using the Bayes Tree. Intl. J. of Robotics Research (IJRR)
- Kim A, Eustice RM. 2013. Real-time visual SLAM for autonomous underwater hull inspection using visual saliency. IEEE Transactions on Robotics (TRO), 29 (3), 719-733.
- Forster C, Carlone L, Dellaert F, Scaramuzza D. 2017. On-manifold reintegration theory for fast and accurate visual-inertial navigation. IEEE Trans. on Robotics 33:1-21
- Hsiung J, Hsiao M, Westman E, Valencia R, Kaess M. 2018. Information Sparsification in Visual-Inertial Odometry. 1146-1153.

# References: Pushing the boundaries

Rosinol A, Abate M, Chang Y, Carlone L. 2020. Kimera: an Open-Source Library for Real-Time Metric-Semantic Localization and Mapping. In IEEE Intl. Conf. on Robotics and Automation (ICRA).

Rosinol A, Gupta A, Abate M, Shi J, Carlone L. 2020 3D Dynamic Scene Graphs: Actionable Spatial Perception with Places, Objects, and Humans. In Robotics: Science and Systems (RSS)

Czarnowski J, Laidlow T, Clark R, Davison A. 2020. DeepFactors: Real-time probabilistic dense monocular SLAM. IEEE Robotics and Automation Letters

Hartley R, Ghaffari Jadidi M, Gan L, Huang JK, Grizzle JW, Eustice RM. 2018. Hybrid Contact Preintegration for Visual-Inertial-contact State Estimation Using Factor Graphs. In IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)

Wisth D, Camurri M, Fallon M. 2019. Robust legged robot state estimation using factor graph optimization. IEEE Robotics and Automation Letters (RA-L)

Wisth D, Camurri M, Fallon M. 2020. Preintegrated Velocity Bias Estimation to Overcome Contact Nonlinearities in Legged Robot Odometry. In IEEE Intl. Conf. on Robotics and Automation (ICRA)

Mukadam M, Dong J, Yan X, Dellaert F, Boots B. 2018. Continuous-time Gaussian Process Motion Planning via Probabilistic Inference. Intl. J. of Robotics Research 37:1319-1340

Xie M, Dellaert F. 2020. Batch and Incremental Kinodynamic Motion Planning using Dynamic Factor Graphs. arXiv preprint arXiv:2005.12514

# References: New Frontiers

Sodhi P, Choudhury S, Mangelson J, Kaess M. 2020. ICS: Incremental Constrained Smoothing for State Estimation. In IEEE Intl. Conf. on Robotics and Automation (ICRA).

Hsiao M, Kaess M. 2019. MH-iSAM2: Multi-hypothesis iSAM using Bayes Tree and Hypo-tree. In IEEE Intl. Conf. on Robotics and Automation (ICRA), pp. 1274-1280. Montreal, Canada

Fourie D, Leonard J, Kaess M. 2016. A Nonparametric Belief Solution to the Bayes Tree. In IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS), Daejeon, Korea

Hsiao M, Mangelson J, Debrunner C, Kaess, M. 2020. ARAS: Ambiguity-aware robust active SLAM with multi-hypothesis mapping. In IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS), Las Vegas, NV. To appear

Davison AJ, Ortiz J. 2019. Futuremapping 2: Gaussian belief propagation for spatial AI. CoRR abs/1910.14139

Ortiz J, Pupilli M, Leutenegger S, Davison AJ. 2020. Bundle Adjustment on a Graph Processor. In IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)

# Questions?

