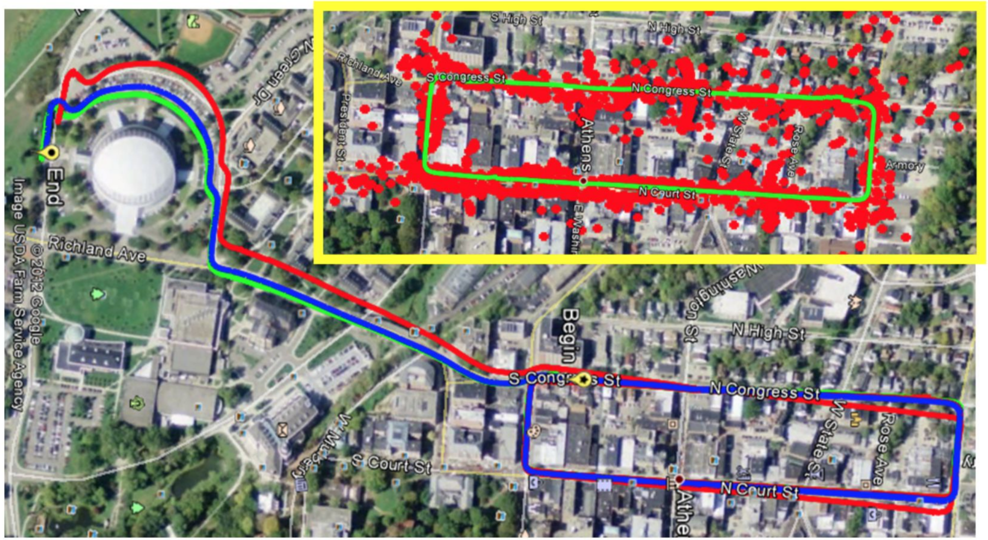
Graph SLAM

Limitations of EKF-SLAM

- Time complexity - **O(n2)**
- Memory complexity - **O(n2)**

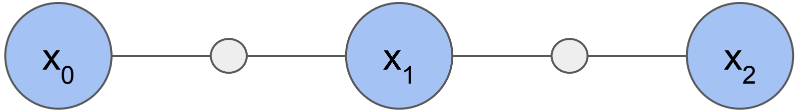
So this becomes difficult for EKF SLAM:



Large City Navigation Scenario (3.3 kilometers) from the DARPA ASPN project

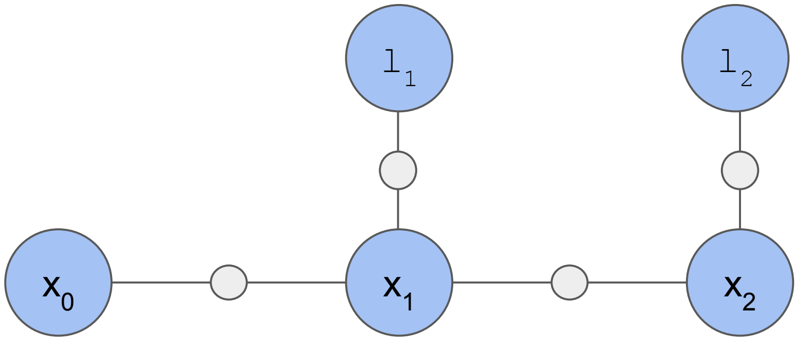
Graph SLAM Representation

- Let’s say that we have a robot moving through space



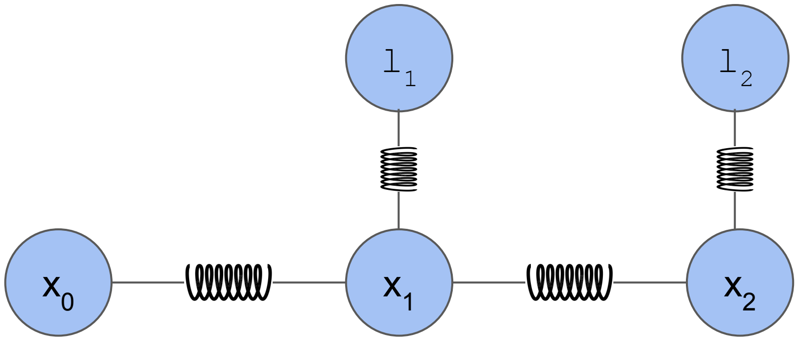
- Each of the **states** can be represented as a variable node in a graph.
- The **action** can be represented as the a constraint denoted by an edge.

Factor Graph Representation



- Each of the **landmarks** can be represented as a node in a graph.
- The **measurement** can be represented as the a constraint denoted by an edge.

Factor Graph Intuition



- Think of each of the constraints as springs
- The stiffness of the string will be the uncertainty.

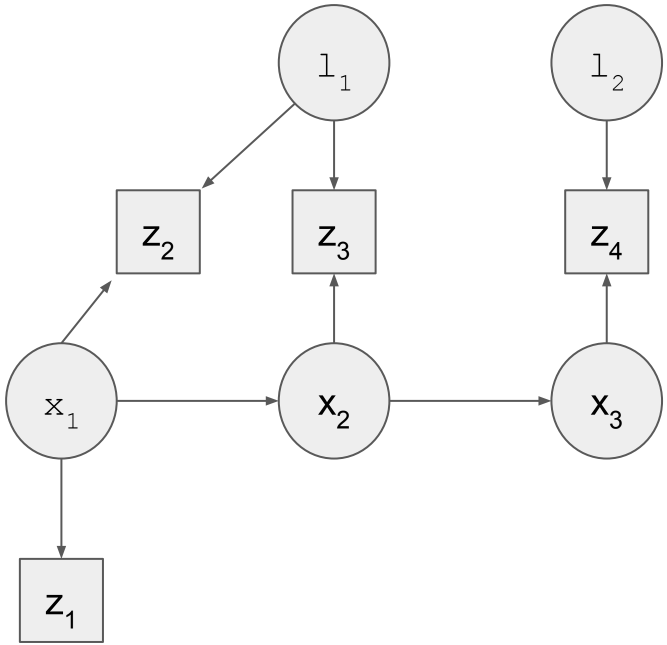
Victoria park dataset

Nebot Et Al

- iSAM paper which uses factor graphs solves the complete problem including data association problem in 7.7 mins, the sequence is 26 min long
- 3.3 times faster than real-time on a laptop computer

Toy Problem: Bayesian network

State: Robot location - x_1, x_2, x_3, l_1, l_2
Measurement: z_1, z_2, z_3, z_4



SLAM is basically figuring out the state given the measurements

So what we want is:

$$p(X|Z)$$

$$p(X|Z) = \frac{p(X,Z)}{P(Z)}$$

To get that we want: $$p(X, Z)$$

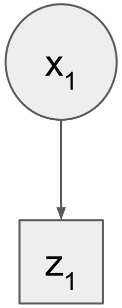
Toy problem: Introducing dynamics

x_1, x_2 is the state of the robot

There are no observation

$$p(X, Z) = p(x_1, x_2) = p(x_2|x_1)p(x_1)$$

Toy problem: Introducing Measurement

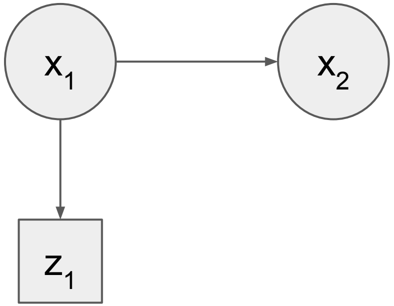


x_1 is the state of the robot

z_1 is the observation

$$p(X, Z) = p(x_1, z_1) = p(z_1|x_1)p(x_1)$$

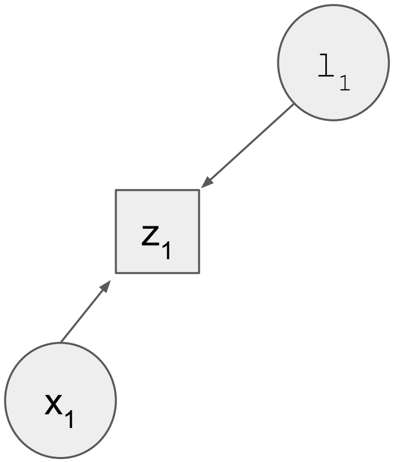
Toy problem: Dynamics and Measurement



x_1, x_2 is the state of the robot
z_1 is the observation

$$p(X, Z) = p(x_1, x_2, z_1) = p(x_2, z_1 | x_1) P(x_1)$$
$$= p(x_2 | x_1) p(z_1 | x_1) p(x_1)$$

Toy problem: Including landmarks



x_1, x_2 is the state of the robot
z_1 is the observation

$$p(X, Z) = p(x_1, l_1, z_1) = p(z_1 | x_1, l_1) p(x_1) p(l_1)$$

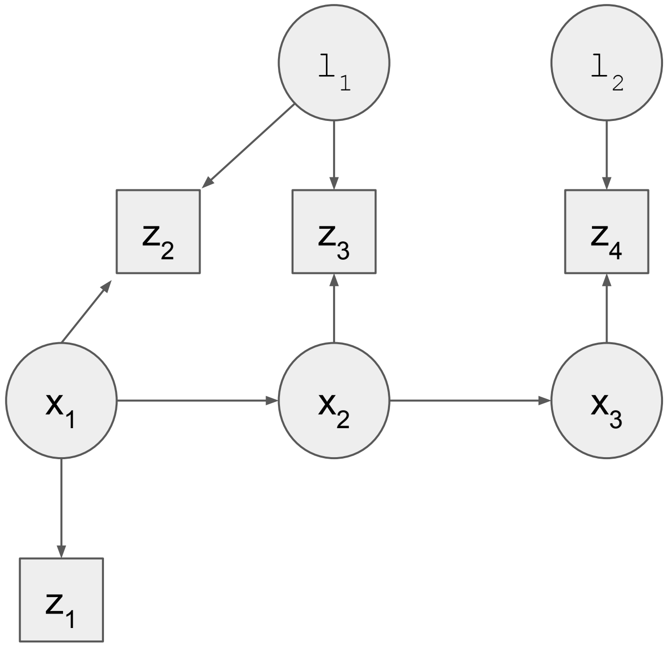
Toy problem:

x_1, x_2, x_3 are the states of the robot
z_i is the observation
l_i is the landmark

Similarly we get

$$p(X, Z) = p(x_1, x_2, x_3, l_1, l_2, z_1, z_2, z_3, z_4)$$

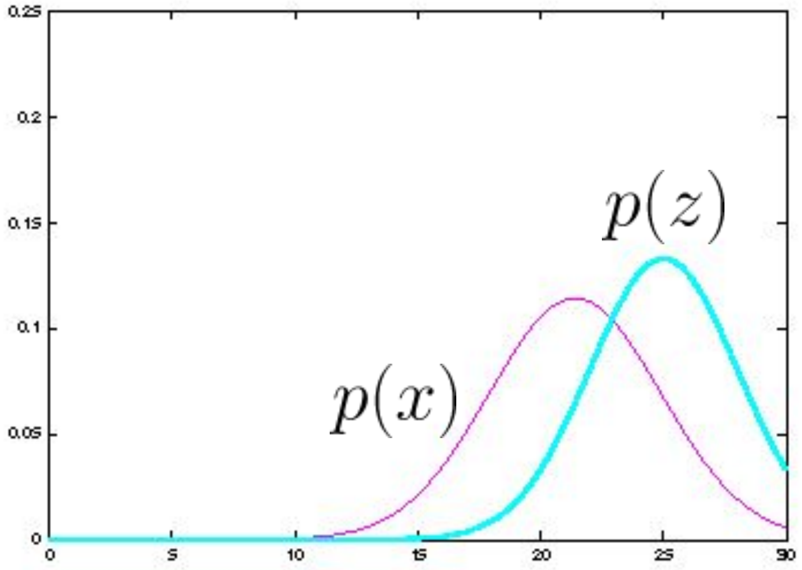
$$\begin{aligned} p(X, Z) = {} & p(x_1)p(x_2|x_1)p(x_3|x_2) \\ & \times p(l_1)p(l_2) \\ & \times p(z_1|x_1) \\ & \times p(z_2|x_1, l_1)p(z_3|x_2, l_1)p(z_4|x_3, l_2). \end{aligned}$$



Circles: Random variables
Square: Measurements

Gaussian Assumption

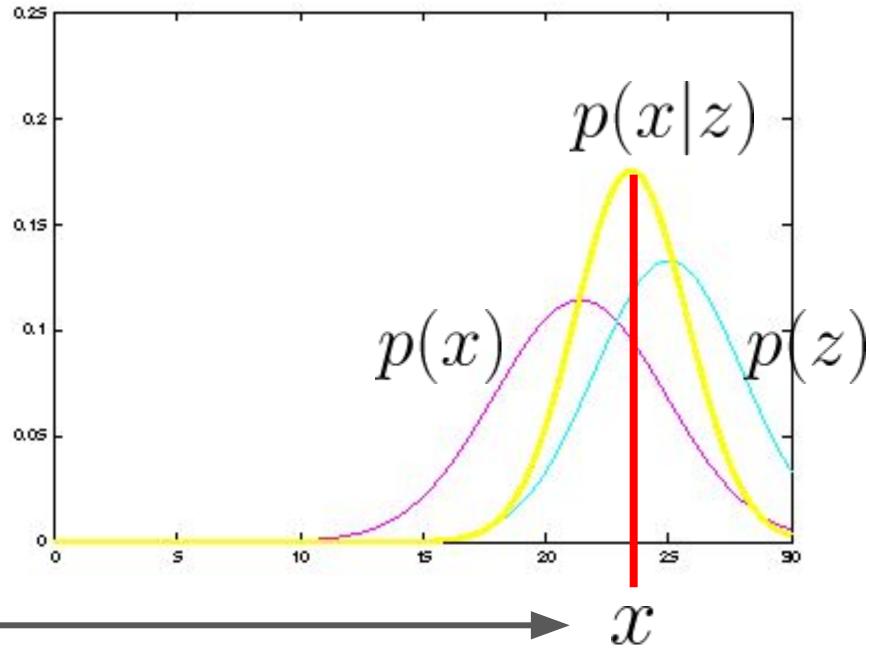
We will make an assumption that each of the probabilities are gaussians.



$$
\begin{aligned}
p(X, Z) = &\ p(x_1)p(x_2|x_1)p(x_3|x_2) \\
&\times p(l_1)p(l_2) \\
&\times p(z_1|x_1) \\
&\times p(z_2|x_1, l_1)p(z_3|x_2, l_1)p(z_4|x_3, l_2).
\end{aligned}
$$

Maximum a Posteriori Inference

For SLAM we want: $p(X|Z)$



We want this

Maximum a Posteriori Inference

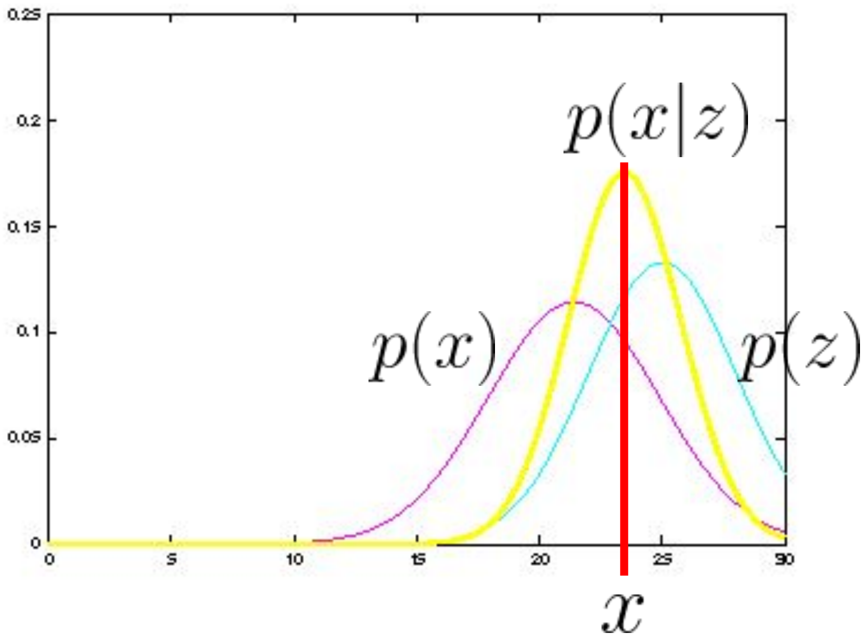
$$X^{MAP} = \underset{X}{\operatorname{argmax}}\, p(X|Z)$$

We know that:

$$p(X|Z) = \frac{p(X,Z)}{P(Z)}$$

Therefore:

$$X^{MAP} = \underset{X}{argmax}\left(p(X,Z)\right)$$



Maximum a Posteriori Inference

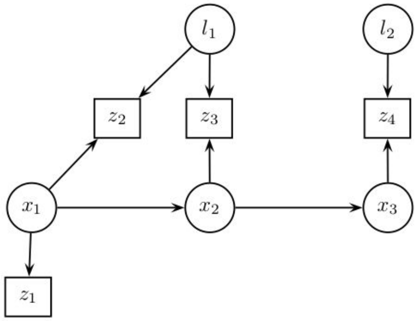
Remember that we had done this:

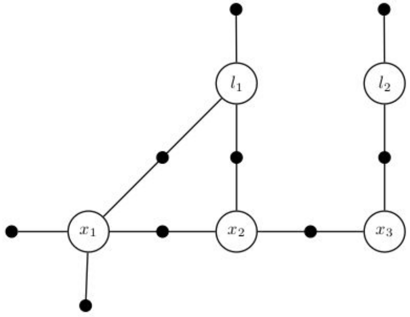
$$
\begin{aligned}
p(X, Z) = {} & p(x_1)p(x_2|x_1)p(x_3|x_2) \\
& \times p(l_1)p(l_2) \\
& \times p(z_1|x_1) \\
& \times p(z_2|x_1, l_1)p(z_3|x_2, l_1)p(z_4|x_3, l_2).
\end{aligned}
$$

$$
X^{MAP} = \underset{X}{argmax} \left(\begin{aligned}
& p(x_1)p(x_2|x_1)p(x_3|x_2) \\
& \times p(l_1)p(l_2) \\
& \times p(z_1|x_1) \\
& \times p(z_2|x_1, l_1)p(z_3|x_2, l_1)p(z_4|x_3, l_2).
\end{aligned} \right)
$$

Introducing factor graphs

- Each of the probability can be represented as a factor



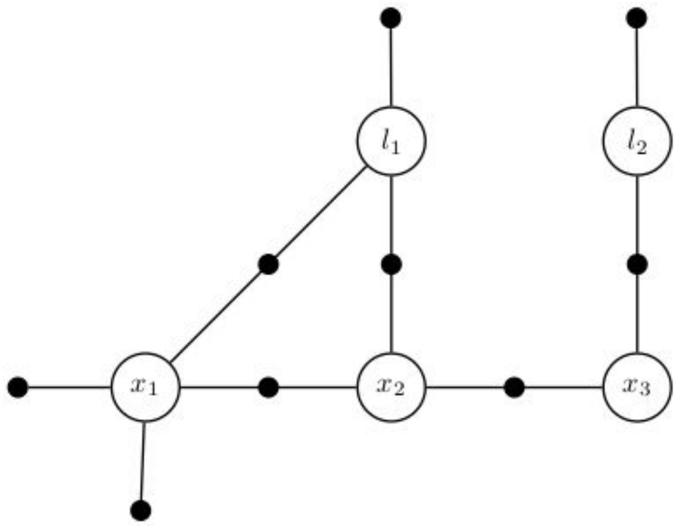


$$
\begin{aligned}
&= p(x_1)p(x_2|x_1)p(x_3|x_2) \\
&\times p(l_1)p(l_2) \\
&\times p(z_1|x_1) \\
&\times p(z_2|x_1, l_1)p(z_3|x_2, l_1)p(z_4|x_3, l_2).
\end{aligned}
$$

$$
\begin{aligned}
&= \phi_1(x_1)\phi_2(x_2, x_1)\phi_3(x_3, x_2) \\
&\times \phi_4(l_1)\phi_5(l_2) \\
&\times \phi_6(x_1) \\
&\times \phi_7(x_1, l_1)\phi_8(x_2, l_1)\phi_9(x_3, l_2),
\end{aligned}
$$

Introducing factor graphs

$$X^{MAP} = \underset{X}{\operatorname{argmax}}\, \phi(X)$$
$$= \underset{X}{\operatorname{argmax}} \prod_i \phi_i(X_i)$$



Expanding the factors

$$\phi(x,l) = p(z|x,l) = \mathcal{N}(z; h(x,l), R) = \frac{1}{\sqrt{|2\pi R|}} \exp\left\{-\frac{1}{2}\|h(x,l) - z\|_R^2\right\}$$

- Remember the Gaussian Assumption:

$$\mathcal{N}(\theta;\mu,\Sigma) = \frac{1}{\sqrt{|2\pi\Sigma|}} \exp\left\{-\frac{1}{2}\|\theta-\mu\|_\Sigma^2\right\},$$

where $\mu \in \mathbb{R}^n$ is the mean, Σ is an $n \times n$ covariance matrix, and

$$\|\theta-\mu\|_\Sigma^2 \stackrel{\Delta}{=} (\theta-\mu)^\top \Sigma^{-1} (\theta-\mu)$$

Expanding the factors

$$\phi(x_{t+1}, x_t) = p(x_{t+1} | x_t, u_t) =$$

$$\frac{1}{\sqrt{|2\pi Q|}} \exp\left\{-\frac{1}{2} \|g(x_t, u_t) - x_{t+1}\|_Q^2\right\}$$

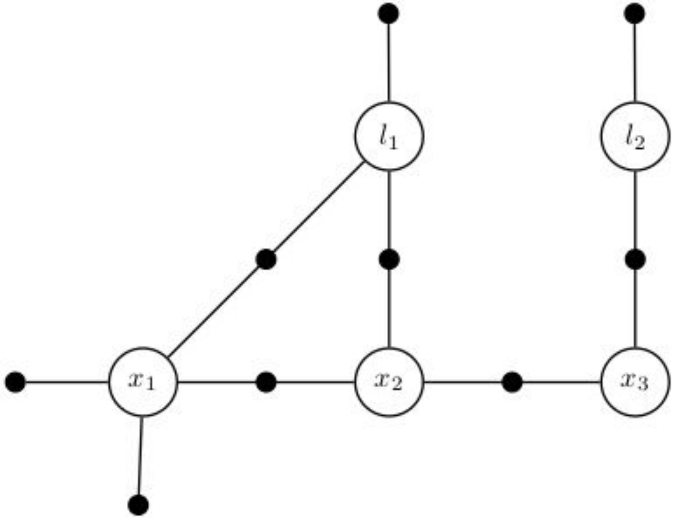
- Remember the Gaussian Assumption:

$$\mathcal{N}(\theta; \mu, \Sigma) = \frac{1}{\sqrt{|2\pi\Sigma|}} \exp\left\{-\frac{1}{2} \|\theta - \mu\|_\Sigma^2\right\},$$

where $\mu \in \mathbb{R}^n$ is the mean, Σ is an $n \times n$ covariance matrix, and

$$\|\theta - \mu\|_\Sigma^2 \stackrel{\Delta}{=} (\theta - \mu)^\top \Sigma^{-1} (\theta - \mu)$$

Expanding the factors



$$X^{MAP} = \underset{X}{\operatorname{argmax}} \prod_i \phi_i(X_i)$$

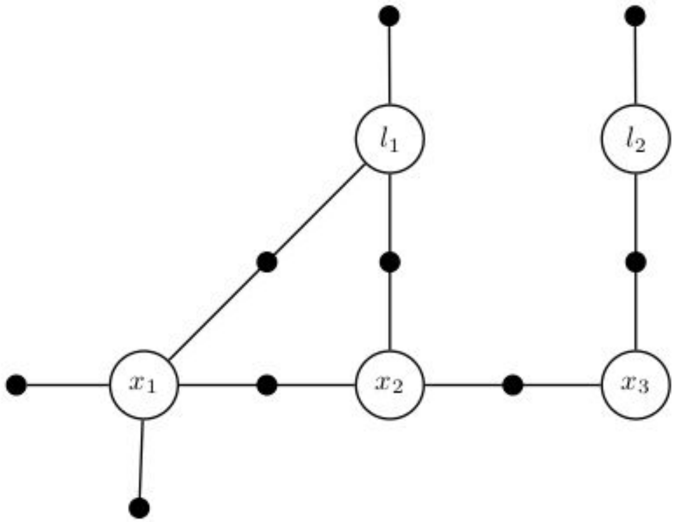
$$= \underset{X}{argmax} \Bigg(\boxed{\frac{1}{\sqrt{2\pi R}} \exp\left\{-\tfrac{1}{2}||h(x_1, l_1) - z_2||\right\}}$$

$$\boxed{\frac{1}{\sqrt{2\pi R}} \exp\left\{-\tfrac{1}{2}||h(x_2, l_1) - z_3||\right\}} \boxed{\frac{1}{\sqrt{2\pi R}} \exp\left\{-\tfrac{1}{2}||h(x_3, l_2) - z_4||\right\}}$$

$$\boxed{\frac{1}{\sqrt{2\pi R}} \exp\left\{-\tfrac{1}{2}||g(x_1, u_1) - x_2||\right\} \frac{1}{\sqrt{2\pi R}} \exp\left\{-\tfrac{1}{2}||g(x_2, u_2) - x_3||\right\}}$$

$$\boxed{\begin{array}{l} \frac{1}{\sqrt{2\pi R}} \exp\left\{-\tfrac{1}{2}||f_1(x_1) - x_1||\right\} \frac{1}{\sqrt{2\pi R}} \exp\left\{-\tfrac{1}{2}||f_2(x_1) - x_1||\right\} \\ \frac{1}{\sqrt{2\pi R}} \exp\left\{-\tfrac{1}{2}||o(l_1) - l_1||\right\} \ \frac{1}{\sqrt{2\pi R}} \exp\left\{-\tfrac{1}{2}||o(l_2) - l_2||\right\} \end{array}}$$

MAP



So, each of the factors can be represented as:

$$\phi_i(X_i) \propto \exp\left\{-\frac{1}{2}\|h_i(X_i) - z_i\|^2_{\Sigma_i}\right\},$$

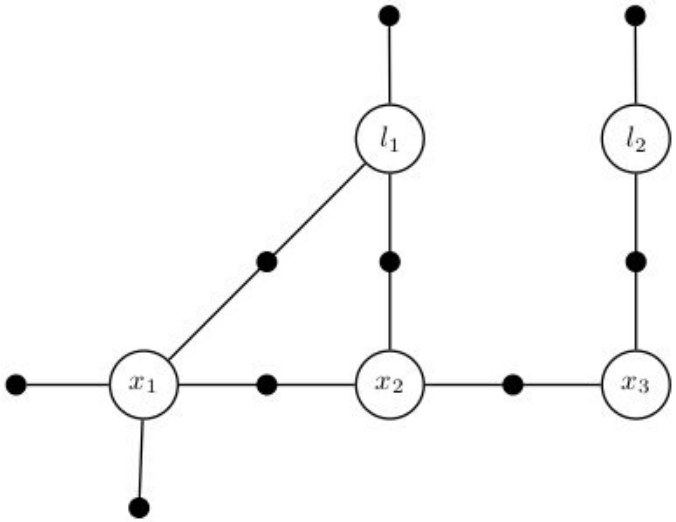
Can be rewritten as follows:

$$X^{MAP} = \underset{X}{\operatorname{argmin}} \sum_i \|h_i(X_i) - z_i\|^2_{\Sigma_i}.$$

We know that:

$$\begin{aligned} X^{MAP} &= \underset{X}{\operatorname{argmax}}\, \phi(X) \\ &= \underset{X}{\operatorname{argmax}} \prod_i \phi_i(X_i) \end{aligned}$$

Linearization



using a simple Taylor expansion, we get:

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

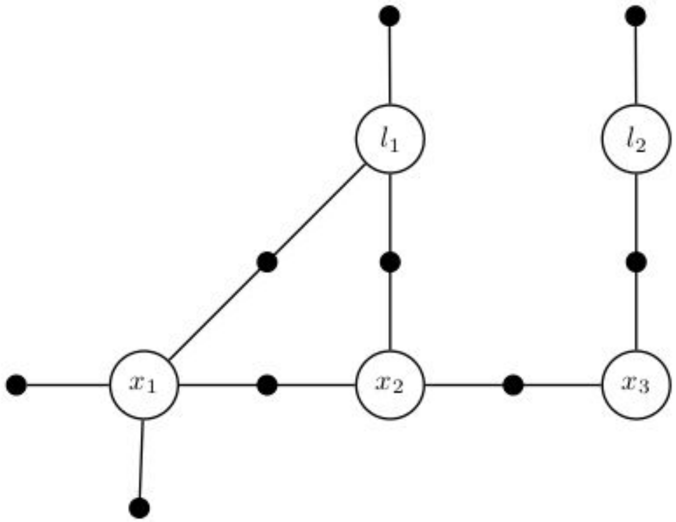
- H is the measurement Jacobian, which is written as:

$$H_i \stackrel{\Delta}{=} \left. \frac{\partial h_i(X_i)}{\partial X_i} \right|_{X_i^0},$$

- Δ is the state update vector, which is written as:

$$\Delta_i \stackrel{\Delta}{=} X_i - X_i^0$$

Linearization



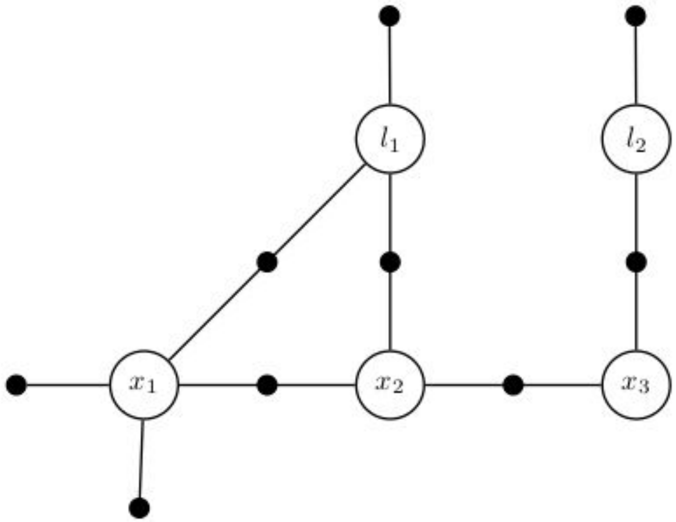
$$
\begin{aligned}
\Delta^* &= \underset{\Delta}{\operatorname{argmin}} \sum_i \left\| h_i(X_i^0) + H_i\Delta_i - z_i \right\|_{\Sigma_i}^2 \\
&= \underset{\Delta}{\operatorname{argmin}} \sum_i \Big\| \underbrace{H_i\Delta_i - \left\{ z_i - h_i(X_i^0) \right\}}_{e} \Big\|_{\Sigma_i}^2 ,
\end{aligned}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

- Rewrite the Mahalanobis norm as follows:

$$\|e\|_\Sigma^2 \stackrel{\Delta}{=} e^\top \Sigma^{-1} e = \left(\Sigma^{-1/2} e\right)^\top \left(\Sigma^{-1/2} e\right) = \left\|\Sigma^{-1/2} e\right\|_2^2 .$$

$$\Delta^* = \underset{\Delta}{argmin} \sum_i ||\Sigma_i^{1/2} H_i \Delta_i - \Sigma_i^{1/2} \left\{ z_i - h_i(X_i^0) \right\} ||_2^2$$

Converting to Least Squares



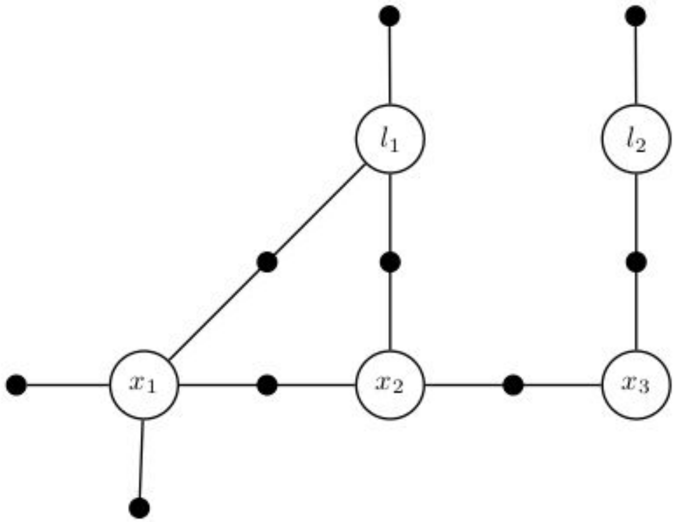
Let:

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

We finally arrive at the form:

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

Solving the Least Squares



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

$$||A\Delta - b||_2^2 = (A\Delta - b)^T (A\Delta - b)$$

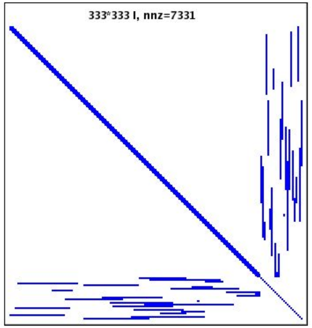
$$||A\Delta - b||_2^2 = \Delta^T A^T A \Delta - 2\Delta^T A^T b + b^t b$$

In order to minimize the error we do:

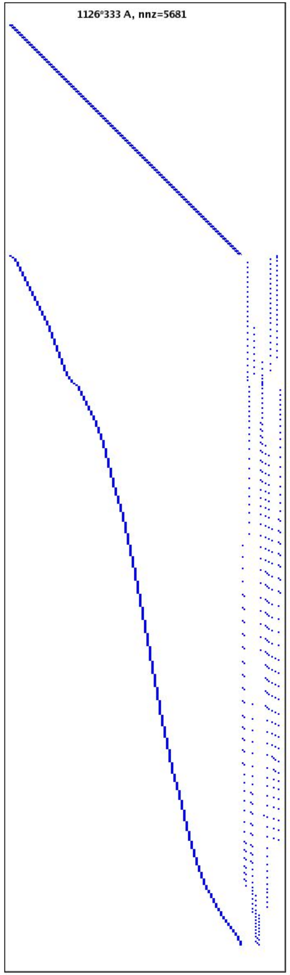
$$\frac{\partial ||A\Delta - b||}{\partial \Delta} = 0 \quad \Rightarrow \quad A^T A \Delta = A^T b$$

The Measurement jacobian

- Each factor represent a constraint between two variables.
- Therefore, the measurement Jacobian is a sparse matrix
- As A is sparse the A^TA is also sparse
- We can use sparse methods which are fast



Information matrix A^TA



Matrix A

Methods for solving the least-squares problem

$$A^T A \Delta = A^T b$$

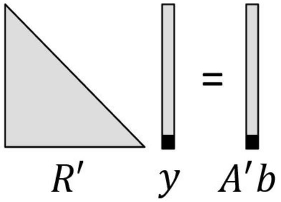
$$\xcancel{\Delta = (A^T A)^{-1} A^T b}$$

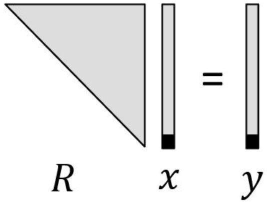
Calculating $(A^TA)^{-1}$ is a bad idea -> $O(n^3)$

$$R^T R \Delta = A^T b$$

Use the Cholesky decomposition $A^T A = R^T R$

- For sparse matrices - $O(m^{1.5})$ to $O(m^2)$





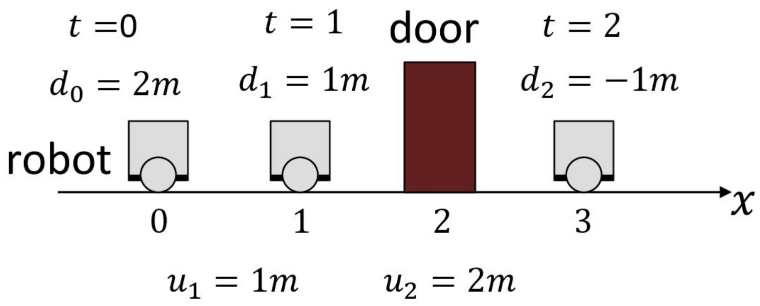
- Finally use the forward substitution and backward substitution to solve

Graph SLAM Summary

- Full SLAM technique
- Graph SLAM leads to sparse matrices
- Suited for Large scale SLAM.
- Batch optimization of multiple sensor measurements.

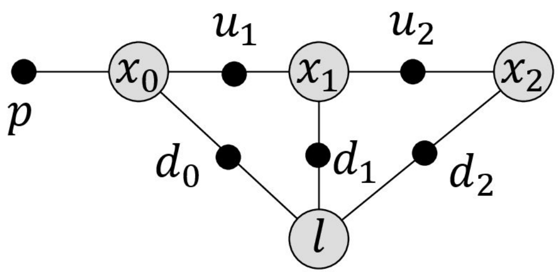
SLAM Least-Squares Example

Localize robot and door based on 1D range measurements



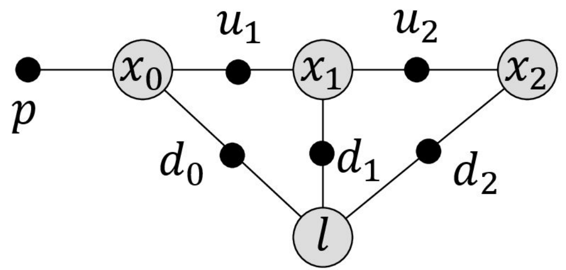
Measurements: distance to the door, signed

Factor graph:



Borrowed from Prof. Michael Keass

Least Square Example



We know that:

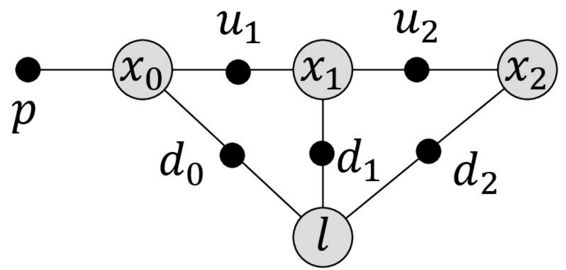
$$\theta^* = \underset{\theta}{argmin} \sum ||h_i(\theta) - z_i||^2_{\Sigma_i}$$

$$= \underset{\theta}{argmin}(||h_p(x_0) - p||^2_{\Sigma_p} +$$

$$||h_u(x_0, x_1) - u_1||^2_{\Sigma_u} + ||h_u(x_1, x_2) - u_2||^2_{\Sigma_u} +$$

$$||h_d(x_0, l) - d_0||^2_{\Sigma_d} + ||h_d(x_1, l) - d_1||^2_{\Sigma_d} + ||h_d(x_2, l) - d_2||^2_{\Sigma_d})$$

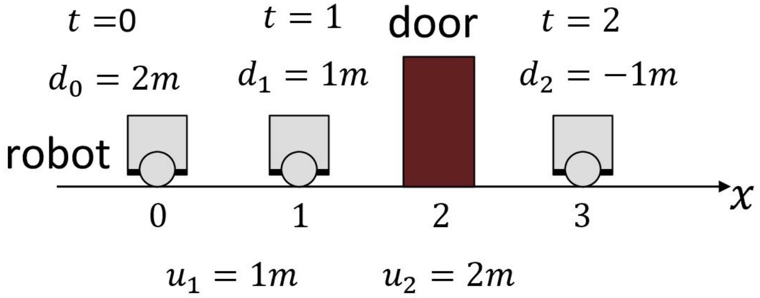
Least Square Example



$$h_u(x_a, x_b) = x_b - x_a \qquad \sigma_u = 0.1m$$

$$h_d(x, l) = l - x \qquad \sigma_d = 0.01m$$

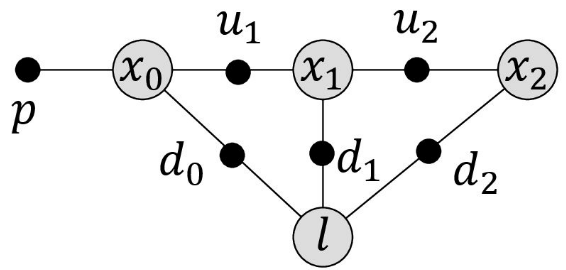
$$h_p(x) = x \qquad \sigma_p = 0.1m$$



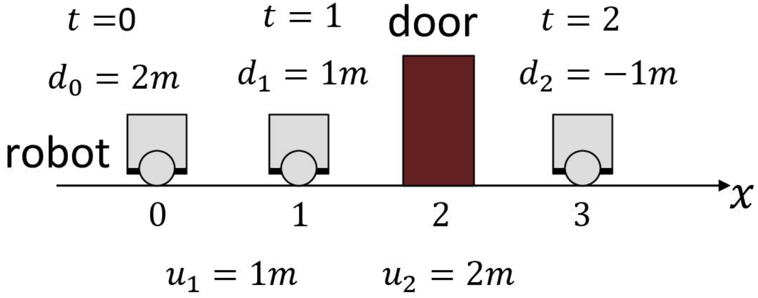
$$||h(x_0, x_1) - u_1||^2_{\Sigma_u} = ||\tfrac{x_1 - x_0}{\sigma_u} - \tfrac{u_1}{\sigma_u}||^2_2$$

$$= || - 10x_0 + 10x_1 - 10||^2_2$$

Least Square Example



$$A = \begin{bmatrix} 10 & 0 & 0 & 0 \\ -10 & 10 & 0 & 0 \\ 0 & -10 & 0 & 0 \\ -100 & 0 & 0 & 100 \\ 0 & -100 & 0 & 100 \\ 0 & 0 & -100 & 100 \end{bmatrix} \quad b = \begin{bmatrix} 0 \\ 10 \\ 20 \\ 200 \\ 100 \\ -100 \end{bmatrix}$$

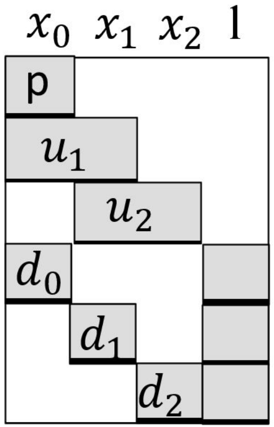


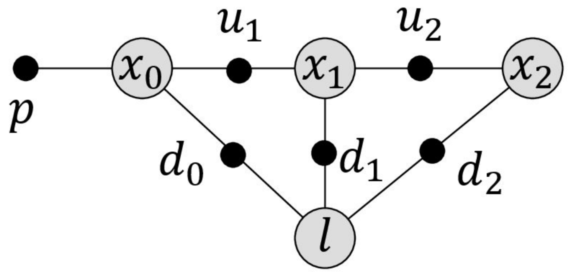
- Solve the following least squares problem

SLAM Least-Squares Example

Localize robot and door based on 1D range measurements

Matrix A:
Each row corresponds to a factor
Each column to a variable
A is sparse!





Borrowed from Prof. Michael Keass

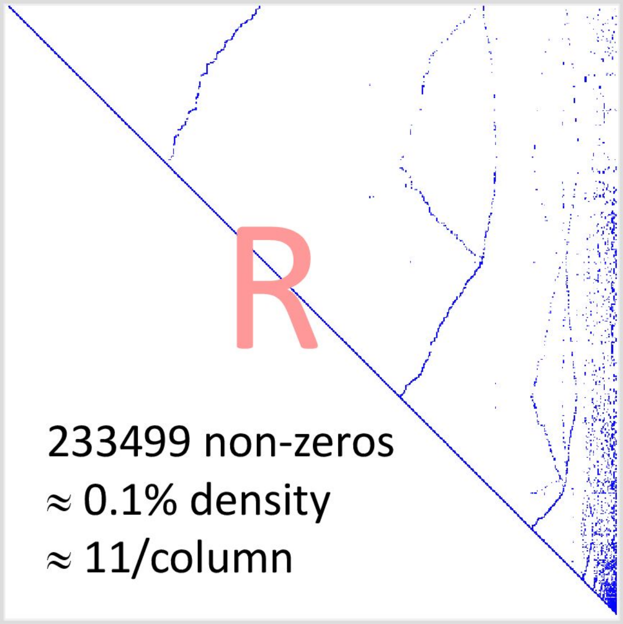
Sparse Factorization Example

Example from real sequence:

Square root inf. matrix

Side length: 21000 variables

Dense: 1.7GB, sparse: 1MB

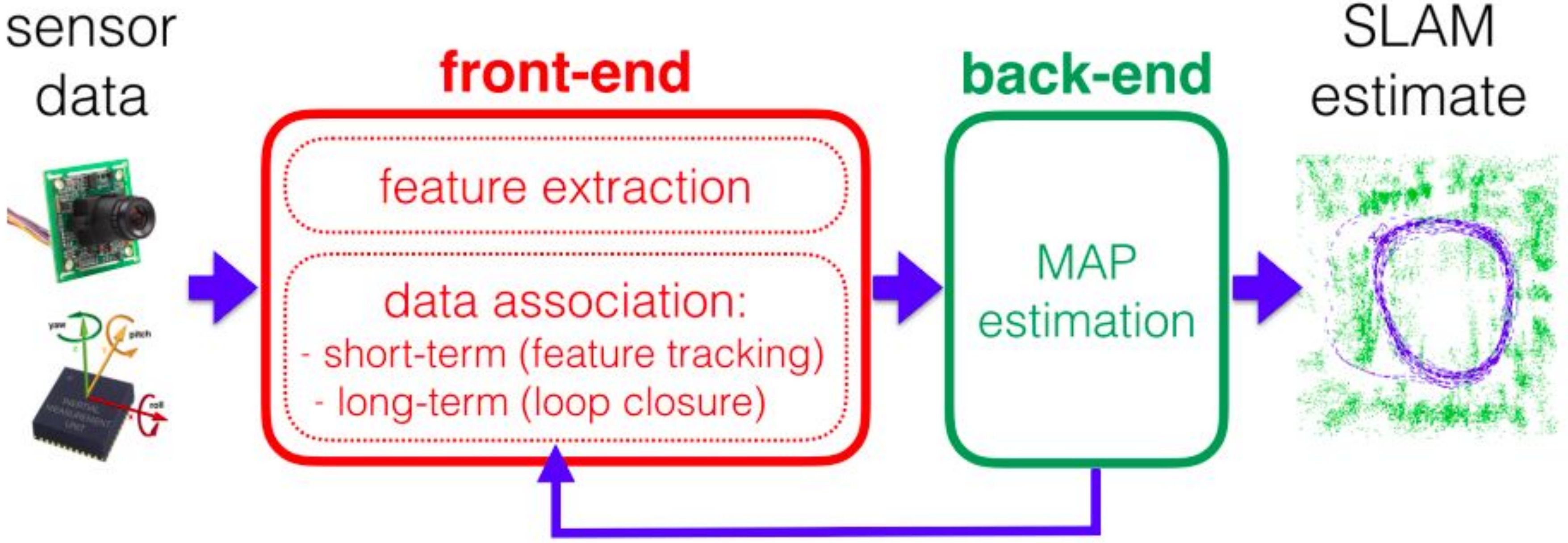


Graph-based SLAM - Intel, 2011

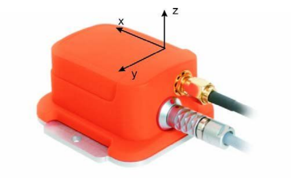
Modern SLAM

Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I. and Leonard, J.J., 2016. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. IEEE Transactions on robotics, 32(6), pp.1309-1332.

A typical SLAM system…



Visual Inertial SLAM - Options



Front End

RGB Cameras:
Direct methods
Indirect methods

Feature Tracking:
KLT tracker

IMU:
IMU preintegration

Loop Closures

Back End

Smoothing and Mapping

Filtering

Visual Inertial SLAM - Front end

- Extracts relevant features from the sensor data.

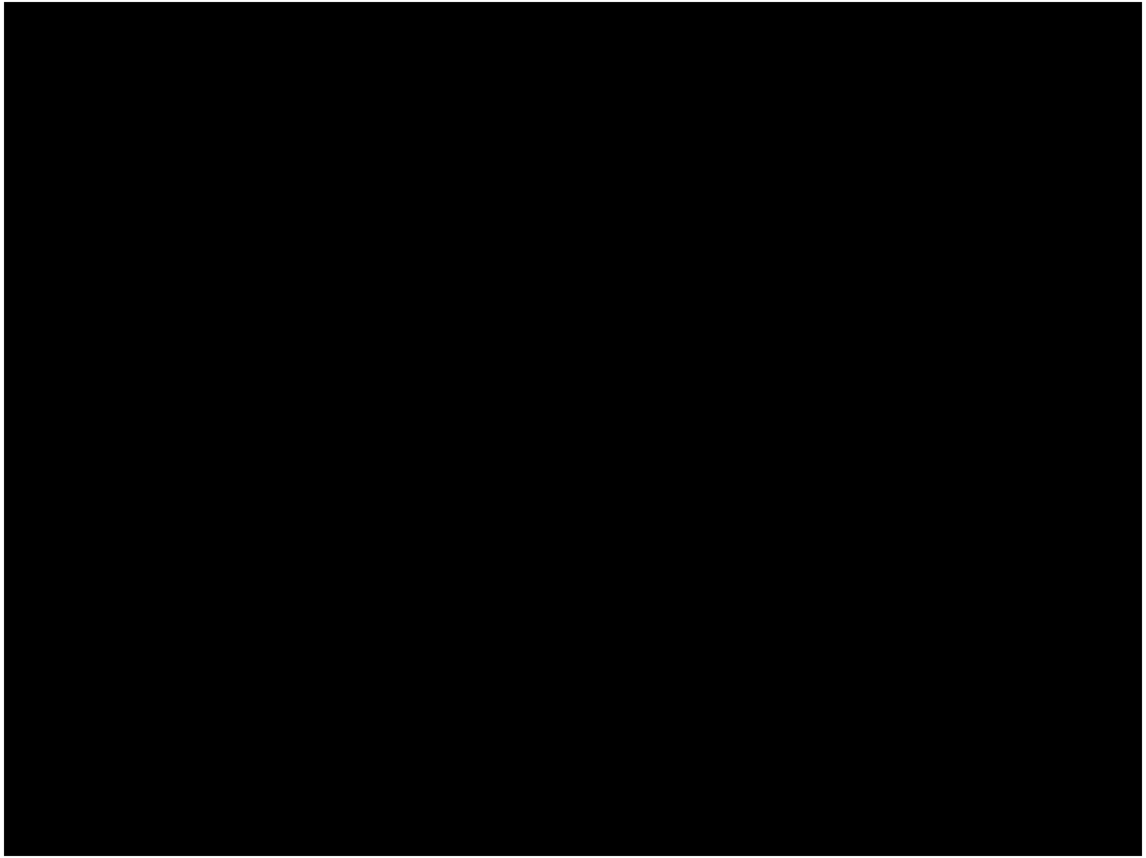
Front end - Direct vs Indirect methods

Indirect Methods	Direct Methods
● Feature-based approaches are quite mature, with a long history of success ● System depends on the availability of features in the environment, the reliance on detection and matching thresholds. ● E.g ORB-SLAM	● System works with the raw pixel information and dense-direct methods exploit all the information in the image. ● Can outperform feature-based methods in scenes with poor texture, defocus, and motion blur. ● Require high computing power (GPUs) for real-time performance. ● E.g. DSO-SLAM

Hybrid Methods: SVO

- The algorithm uses sparse model-based image alignment for motion estimation
- The algorithm uses point-features for BA

Direct vs Indirect methods

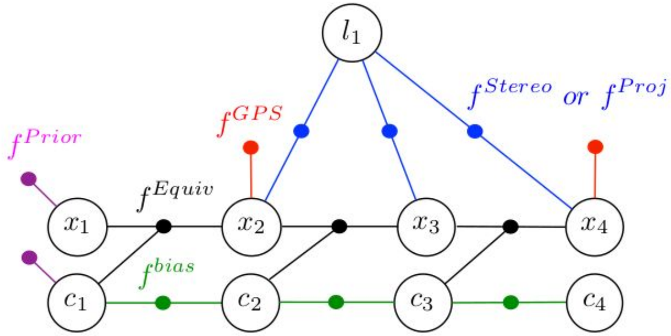


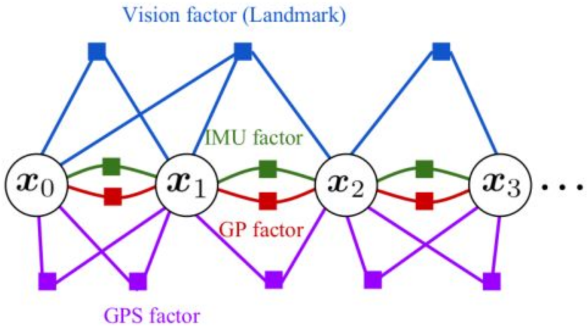
https://youtu.be/C6-xwSOOdqQ

Back-end and comparisons

Smoothing and Mapping	Filtering
● Enables an insightful visualization of the problem. ● Factor graphs can model complex inference problems ● The connectivity of the factor graph in turn influences the sparsity of the resulting SLAM problem	● Proven to be less accurate and efficient compared to smoothing methods ● Some of the SLAM systems based on EKF have also been demonstrated to attain state-of-the-art performance. E.g. Multistate Constraint Kalman Filter.

Backend examples…





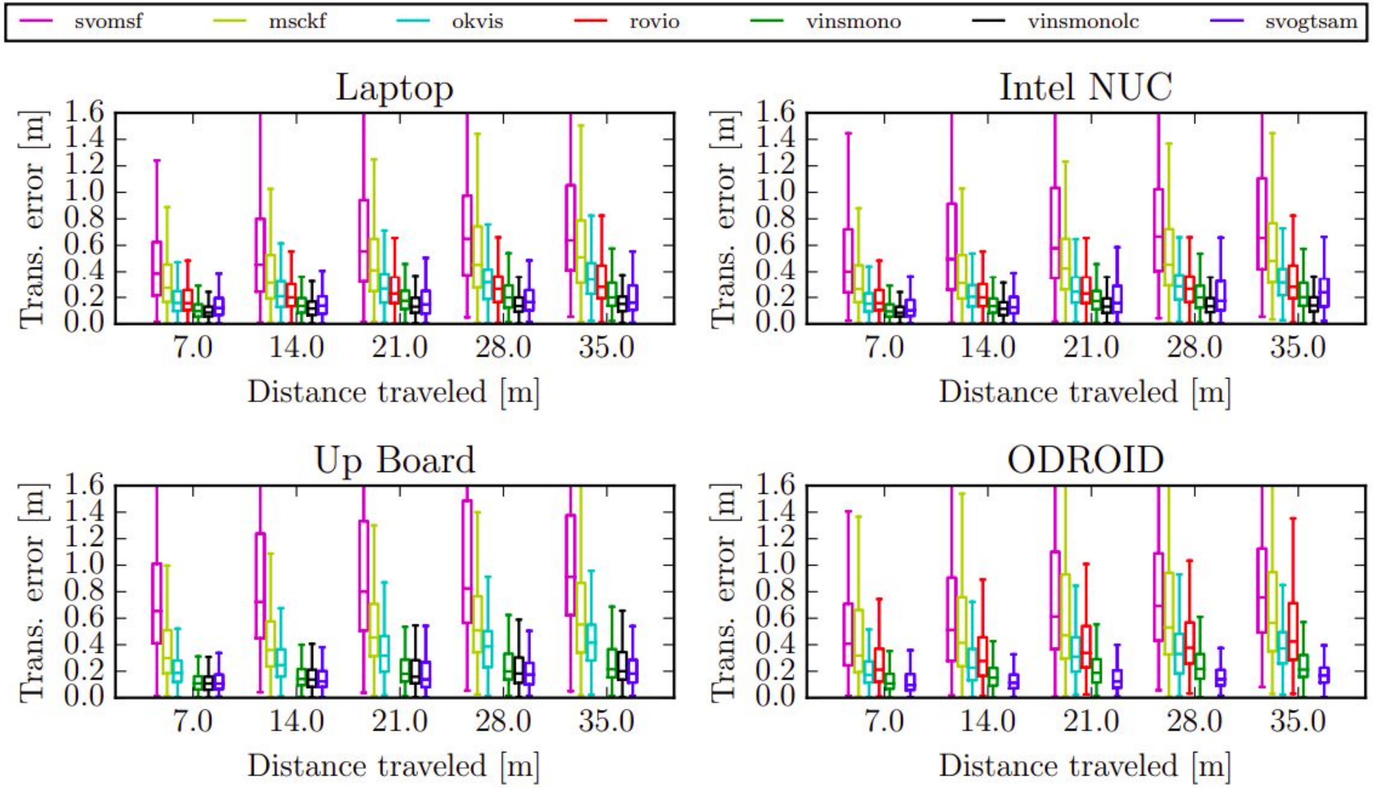
Comparison of Monocular Visual-Inertial Odometry

J. Delmerico and D. Scaramuzza, "A Benchmark Comparison of Monocular Visual-Inertial Odometry Algorithms for Flying Robots," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, QLD, Australia, 2018, pp. 2502-2509, doi: 10.1109/ICRA.2018.8460664.

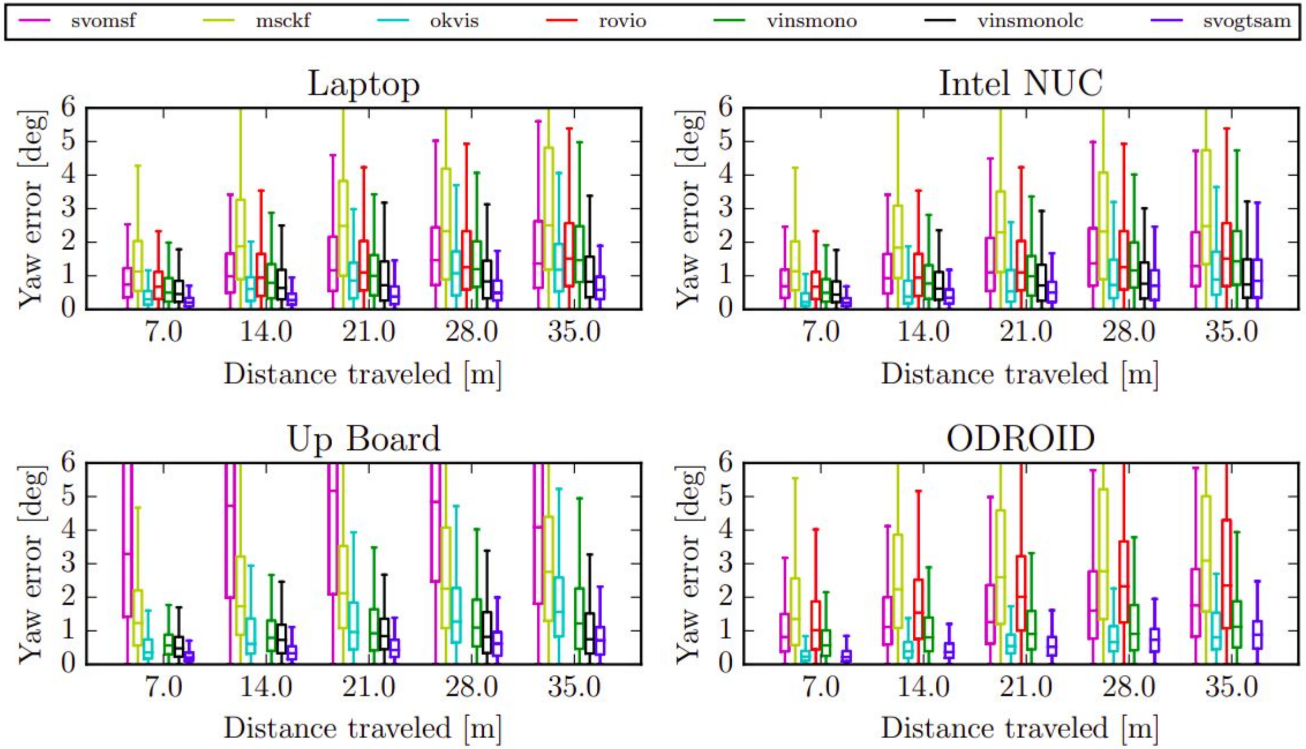
Algorithms being compared

- MSCKF: An Extended Kalman Filter (EKF)-based algorithm for real-time vision-aided inertial navigation [2007].
- Open Keyframe-based Visual-Inertial SLAM (OKVIS) utilizes non-linear optimization on a sliding window of keyframe poses.
- ROVIO: Visual-Inertial state estimator based on an extended Kalman Filter (EKF), which proposed several novelties.
- VINS-Mono: A nonlinear-optimization-based sliding window estimator using pre-integrated IMU factors.
- SVO+GTSAM: SVO in front end paired with a full-smoothing backend performing online factor graph optimization using iSAM2.

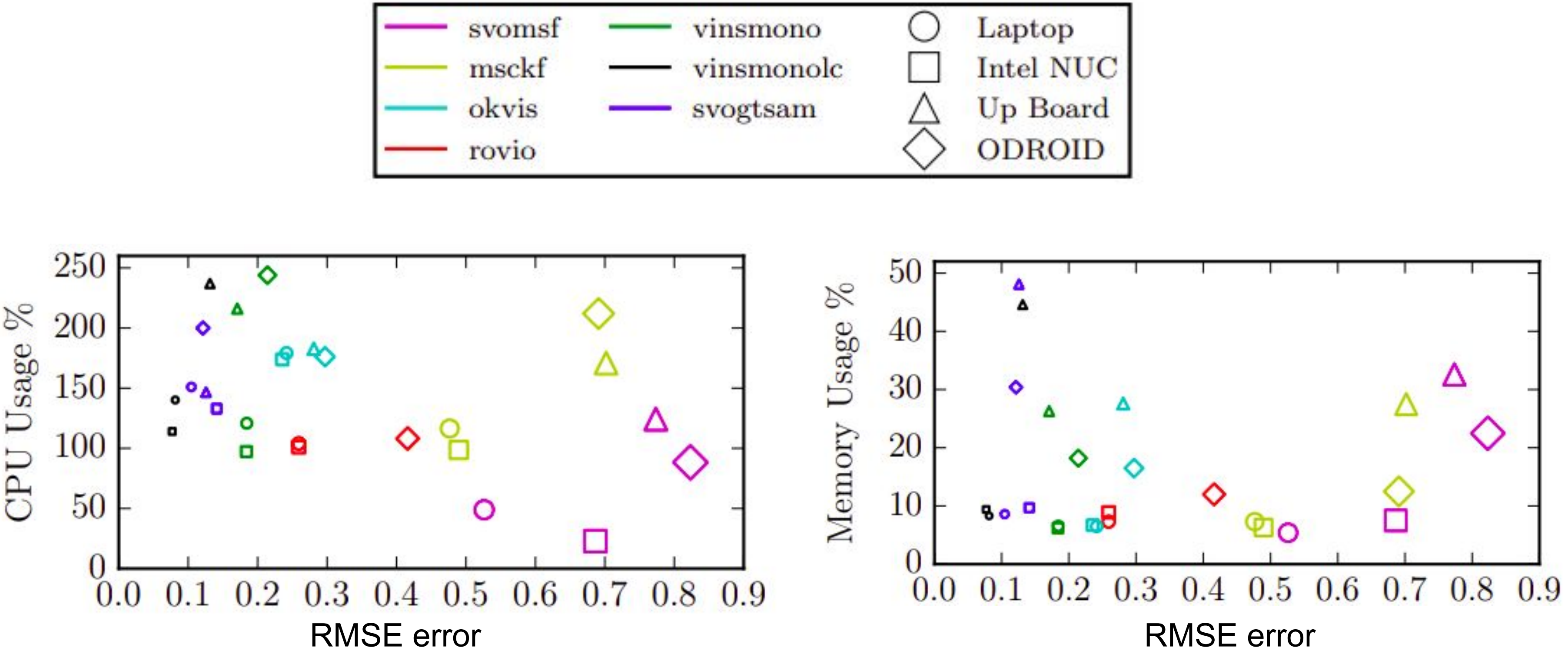
Comparison of Translation Error



Comparison of Yaw Errors

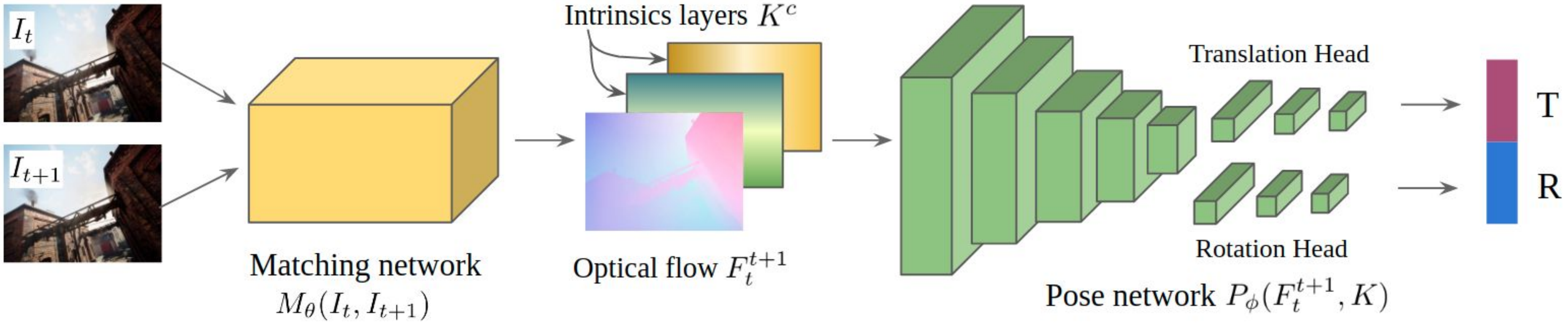


Algorithm Efficiency



Deep Learning for SLAM

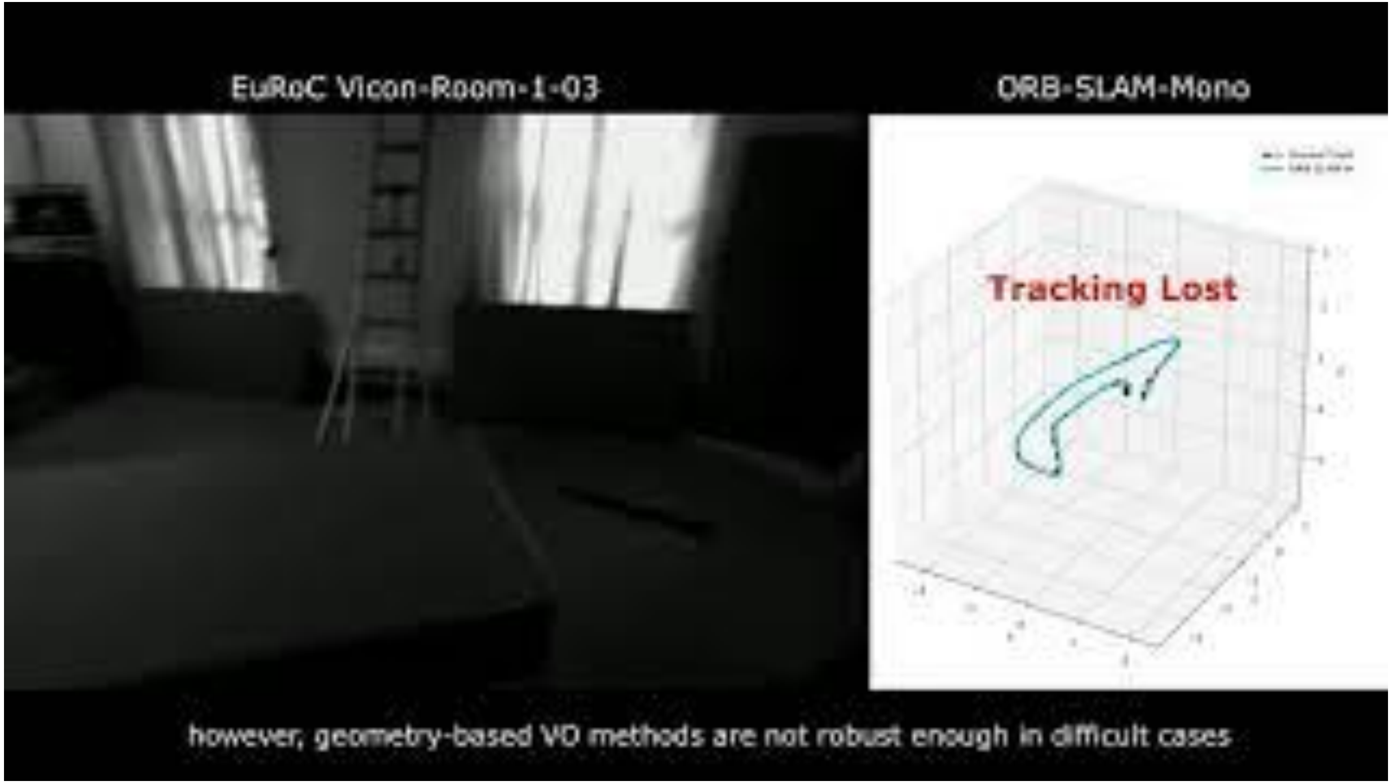
TartanVO



	Seq.	MH-04	MH-05	VR1-02	VR1-03	VR2-02	VR2-03
Geometry-based *	SVO [46]	1.36	0.51	0.47	x	0.47	x
	ORB-SLAM [3]	**0.20**	0.19	x	x	**0.07**	x
	DSO [5]	0.25	**0.11**	**0.11**	0.93	0.13	1.16
	LSD-SLAM [2]	2.13	0.85	1.11	x	x	x
Learning-based †	TartanVO (ours)	0.74	0.68	0.45	**0.64**	0.67	**1.04**

* These results are from [46]. † Other learning-based methods [36] did not report numerical results.

TartanVO


EuRoC Vicon-Room-1-03
ORB-SLAM-Mono
Tracking Lost
however, geometry-based VO methods are not robust enough in difficult cases