Robotics

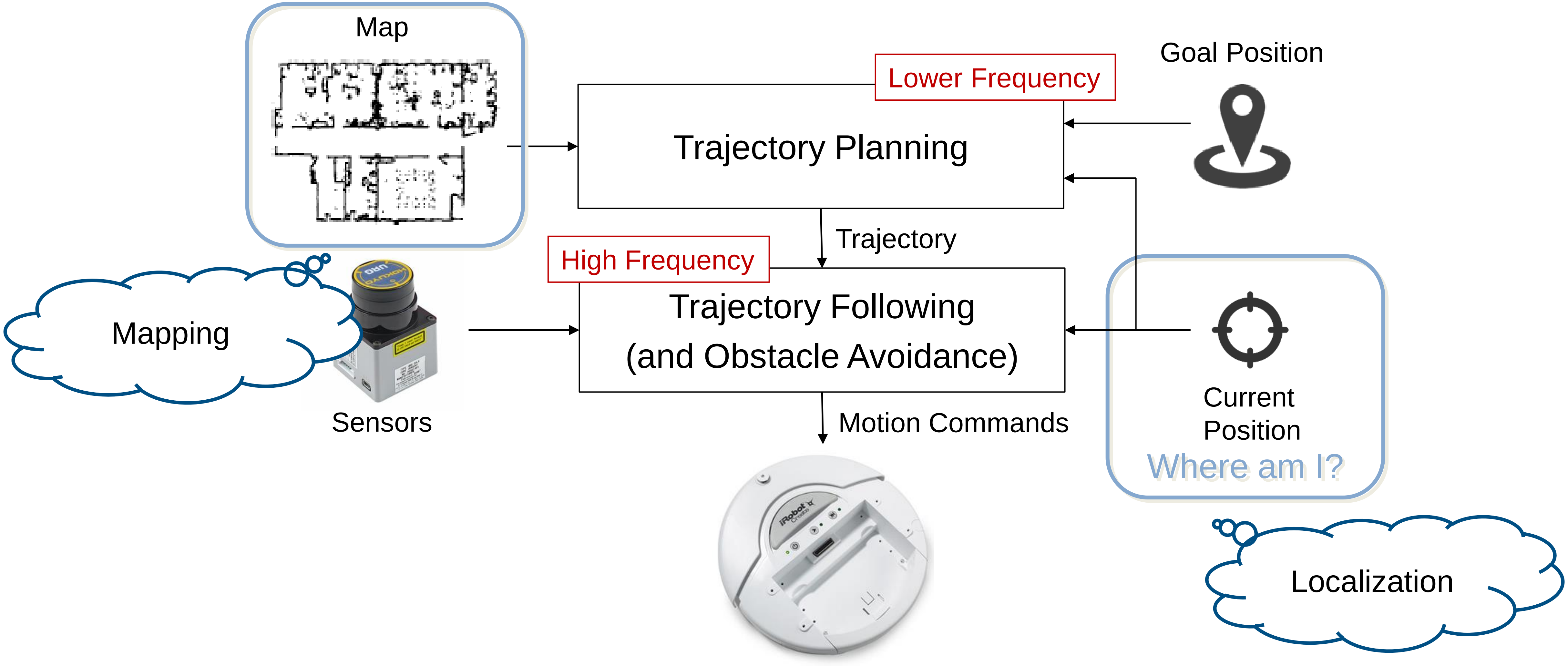
Simultaneous Localization and Mapping

Matteo Matteucci

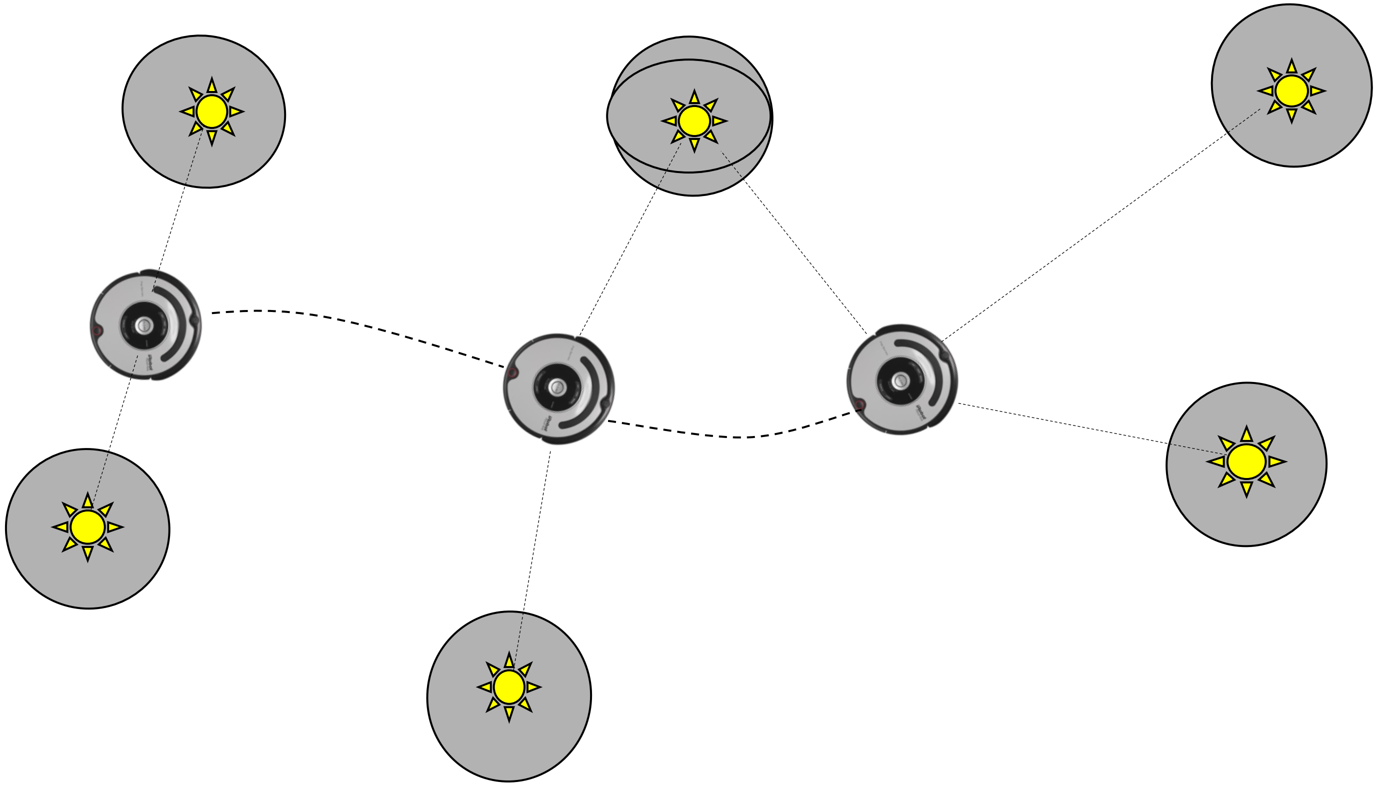
matteo.matteucci@polimi.it

Artificial Intelligence and Robotics Lab - Politecnico di Milano

A Simplified Sense-Plan-Act Architecture

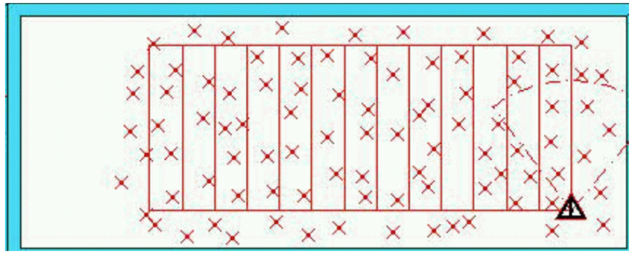


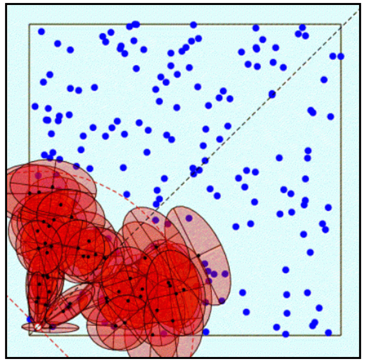
Mapping with Known Poses

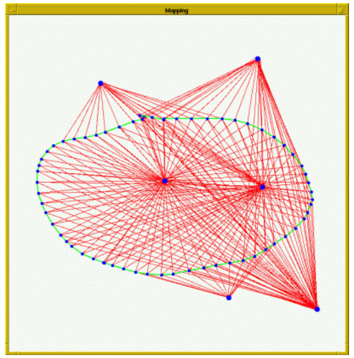


Representations

Landmark-based

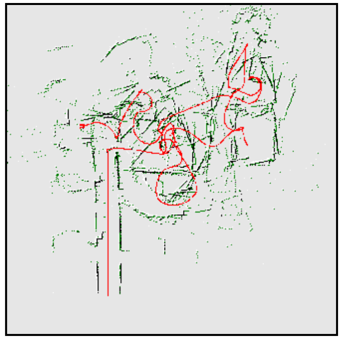


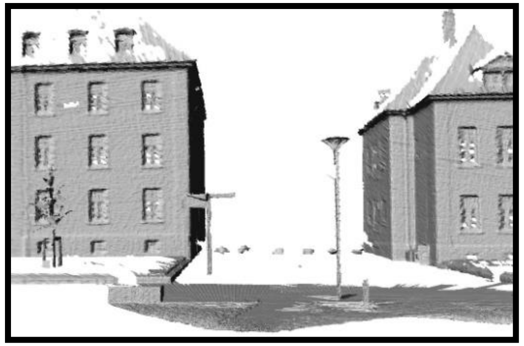


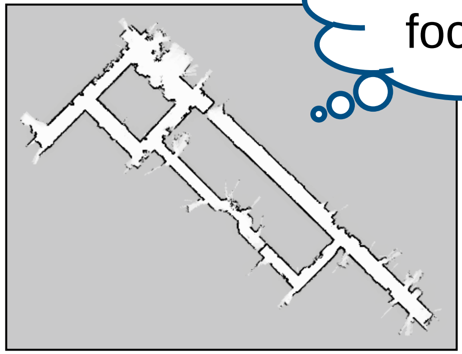


[Leonard et al., 98; Castelanos et al., 99: Dissanayake et al., 2001; Montemerlo et al., 2002;…]

Grid maps or scans







We'll mostly focus on these

[Lu & Milios, 97; Gutmann, 98: Thrun 98; Burgard, 99; Konolige & al., 00; Thrun, 00; Arras, 99; Haehnel, 01;…]

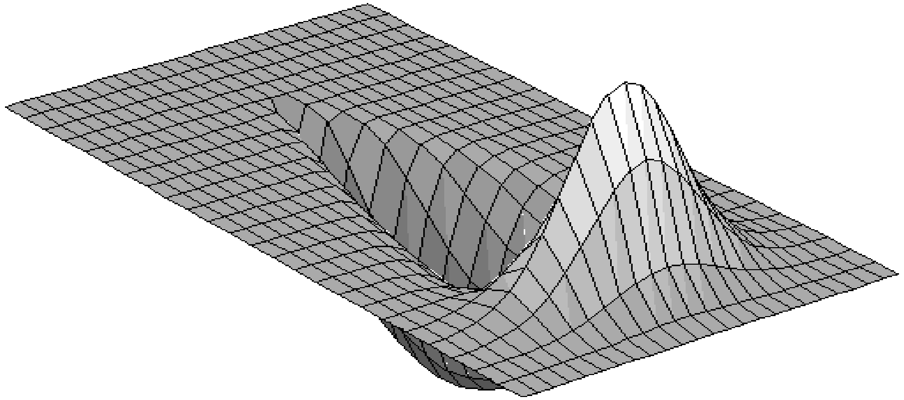
Occupancy from Sonar Return (the origins)

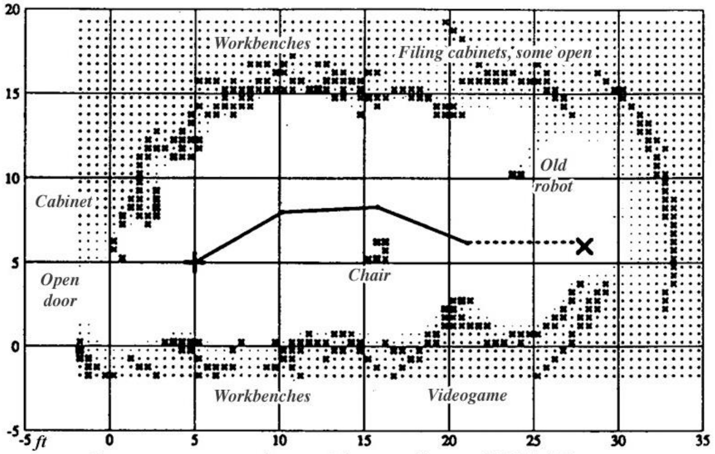
The most simple occupancy model used sonars

- A 2D Gaussian for information about occupancy
- Another 2D Gaussian for free space

Sonar sensors present several issues

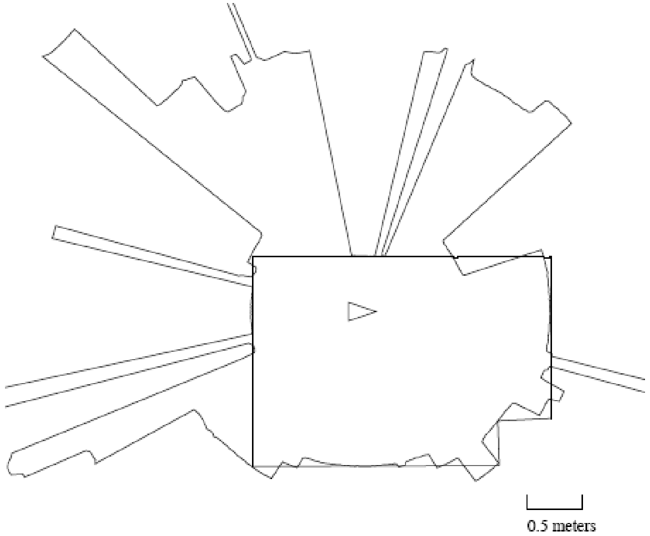
- A wide sonar cone creates noisy maps
- Specular (multi-path) reflections generates unrealistic measurements





Room traverse by grid map from SONAR

Moravec 1984



2D Occupancy Grids

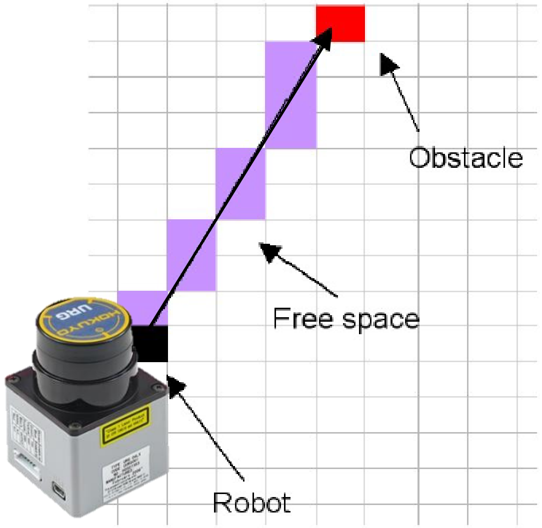
A simple 2D representation for maps

- Each cell is assumed independent
- Probability of a cell of being occupied estimated using Bayes theorem

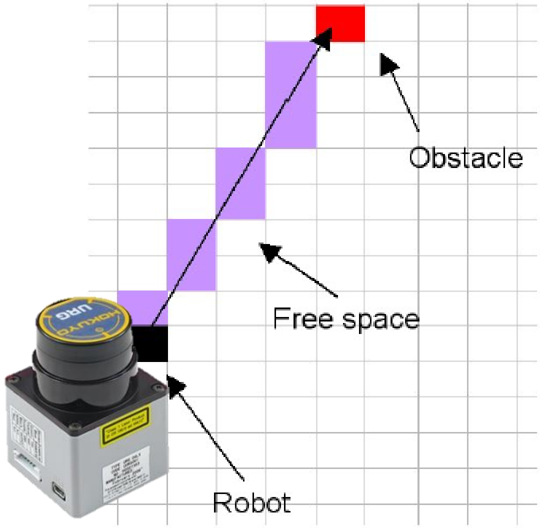
$$P(A|B) = \frac{P(B|A)P(A)}{P(B|A)P(A) + P(B|\sim A)P(\sim A)}$$

Maps the environment as an array of cells

- Usual cell size 5 to 50cm
- Each cells holds the probability of the cell to be occupied
- Useful to combine different sensor scans and different sensor modalities



Occupancy Grid Cell Update



Let $occ(i,j)$ mean cell C_{ij} is occupied, we have

- Probability: $\mathrm{P}(occ(i,j))$ has range $[0,1]$
- Odds: $\mathrm{o}(occ(i,j))$ has range $[0,\infty]$

$$\mathrm{o}\big(occ(i,j)\big) = \mathrm{P}(occ(i,j))/\mathrm{P}(\neg occ(i,j))$$

- Log odds: $\log \mathrm{o}(occ(i,j))$ has range $[-\infty,\infty]$

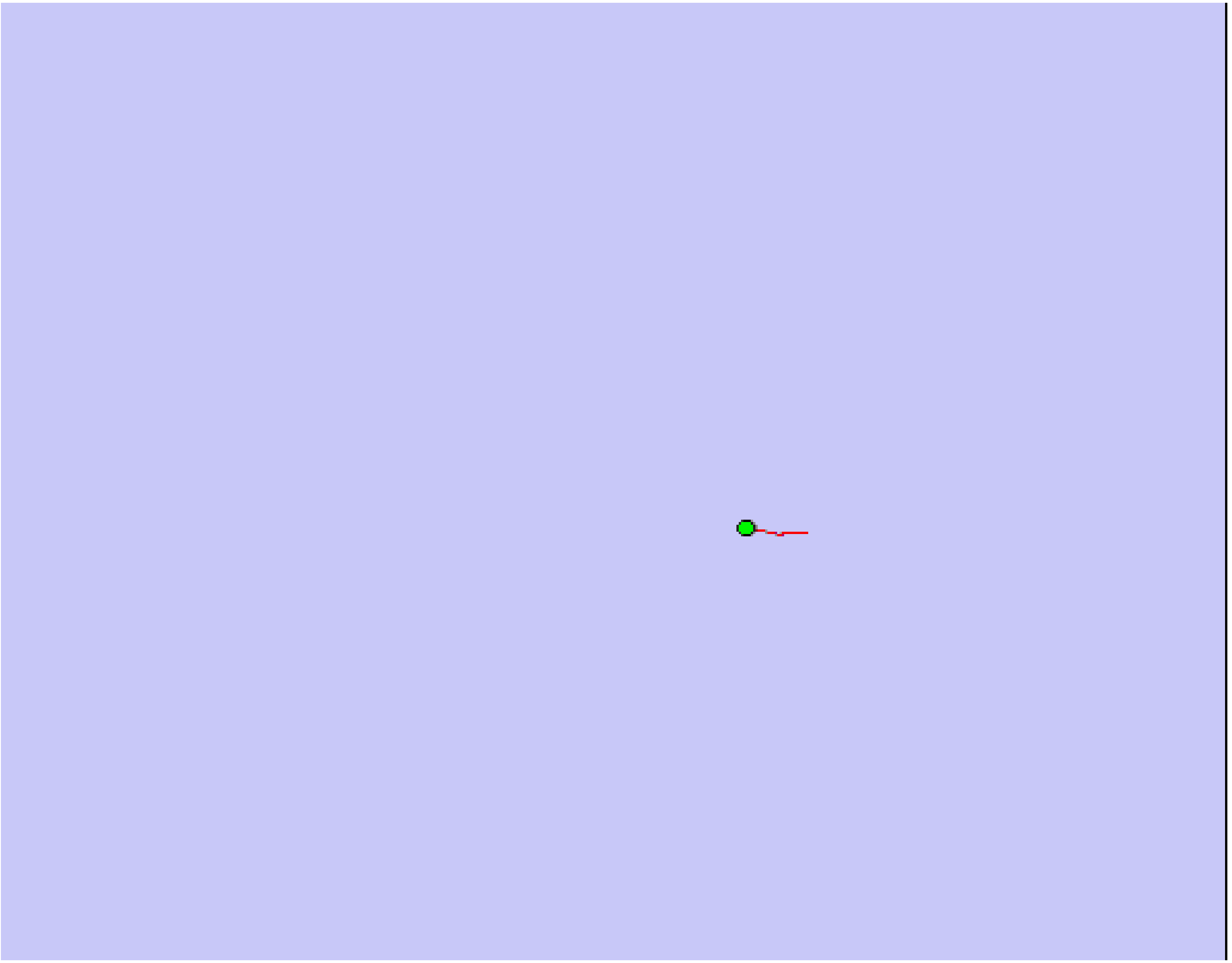
Each cell C_{ij} holds the value $\log \mathrm{o}(occ(i,j))$, $C_{ij} = 0$ corresponds to $\mathrm{P}\big(occ(i,j)\big) = 0.5$

Cells are updated recursively by applying the Bayes theorem

- $A = occ(i,j)$
- $B = measure(i,j)$

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Mapping with Raw Odometry (assuming known poses)



Scan Matching

$\hat{x}^{[t]}$

Correct odometry by maximizing the likelihood of pose *t* based on the estimates of pose and map at time *t-1*.

$$\hat{x}_t = \arg\max_{x_t} \left\{ p(z_t \mid x_t, \hat{m}^{[t-1]}) \cdot p(x_t \mid u_{t-1}, \hat{x}_{t-1}) \right\}$$

current measurement

robot motion

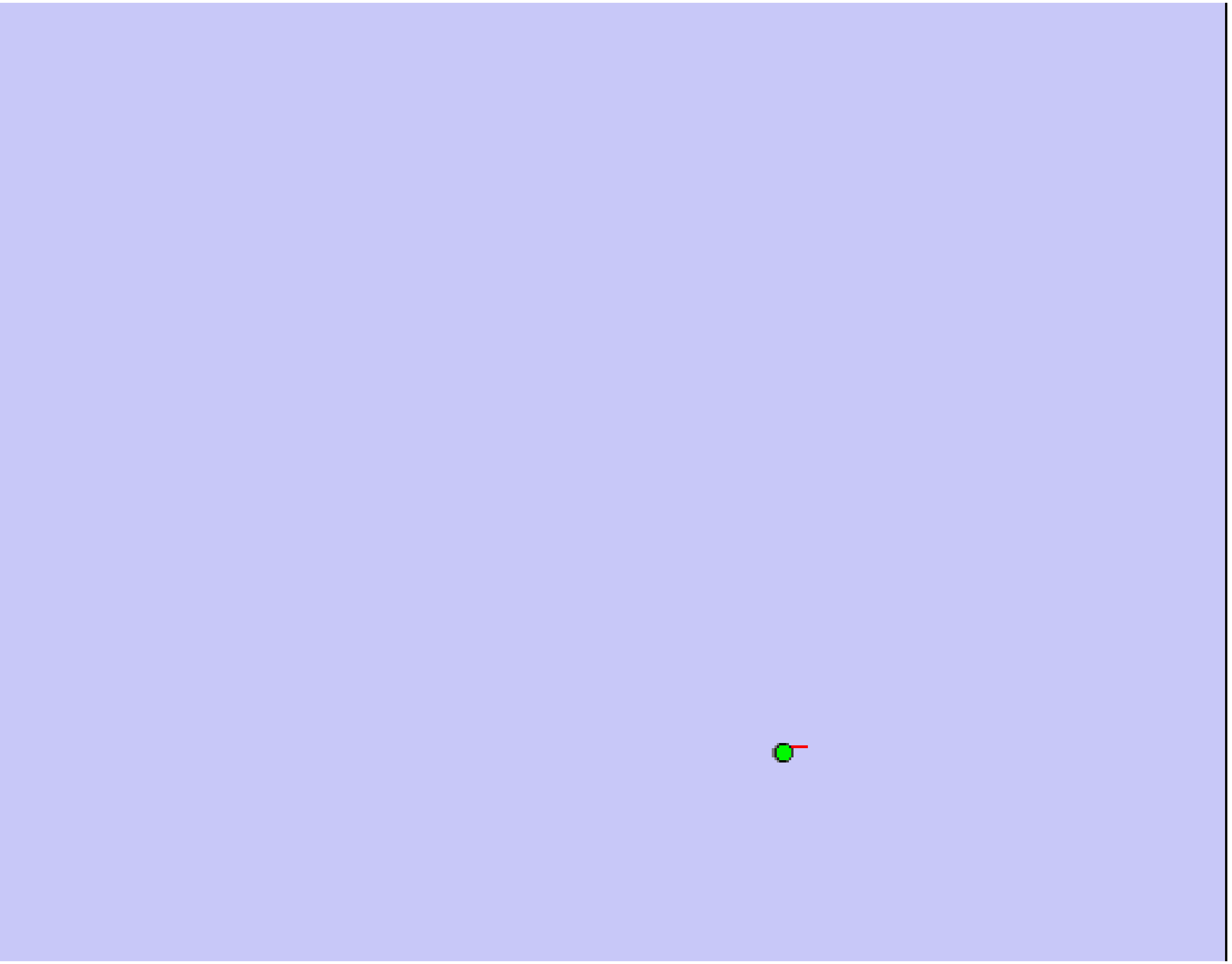
map constructed so far

$\hat{m}^{[t]}$

Then compute the map $\hat{m}^{[t]}$ according to “mapping with known poses” based on the new pose and current observations.

Iterate alternating the two steps of localization and mapping …

Scan Matching Example



Scan Matching

$\hat{x}^{[t]}$

Correct odometry by maximizing the likelihood of pose *t* based on the estimates of pose and map at time *t-1*.

$$\hat{x}_t = \arg\max_{x_t} \left\{ p(z_t \mid x_t, \hat{m}^{[t-1]}) \cdot p(x_t \mid u_{t-1}, \hat{x}_{t-1}) \right\}$$

current mea

motion

map con cted so far

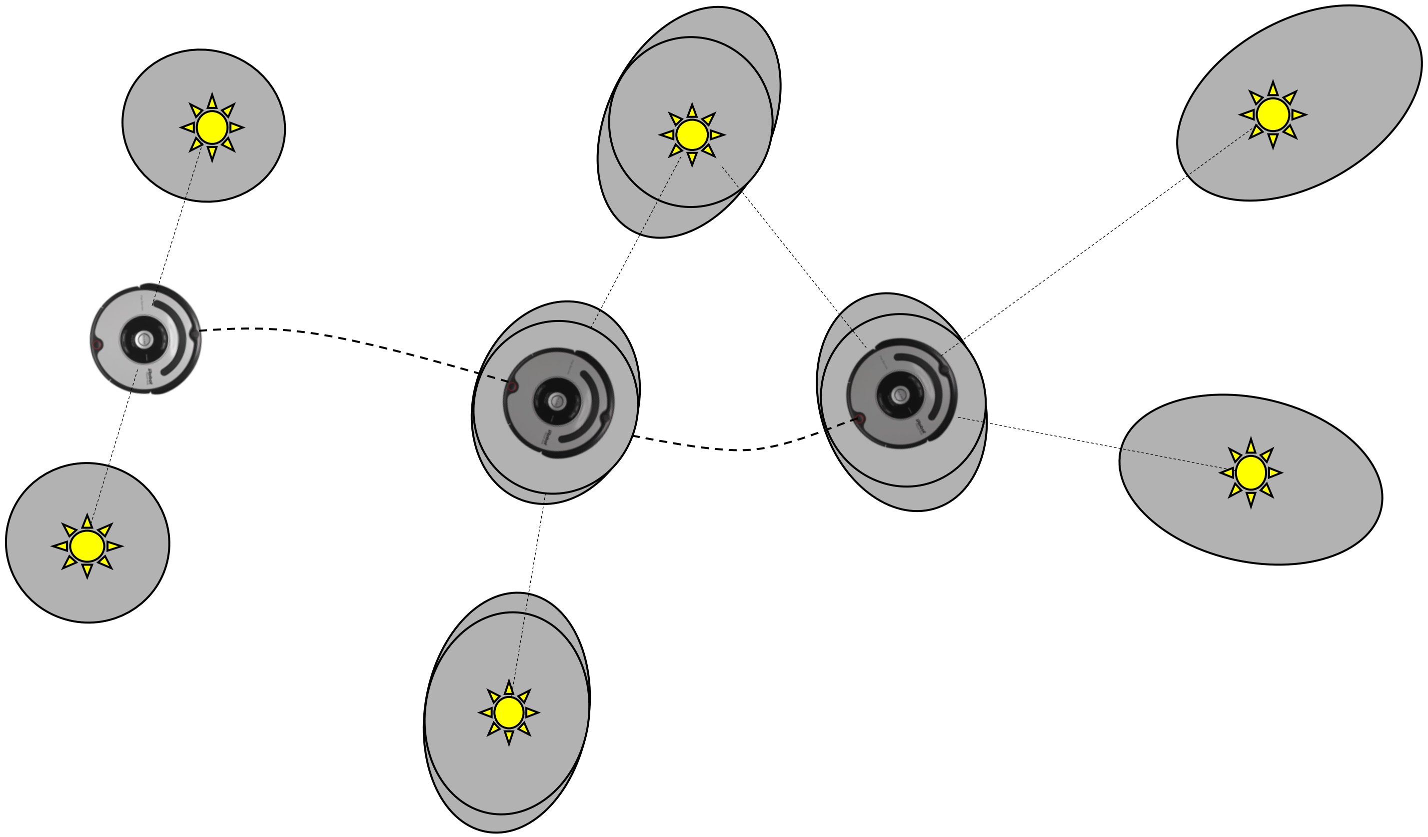
Does not keep track of the uncertainty in the process

$\hat{m}^{[t]}$

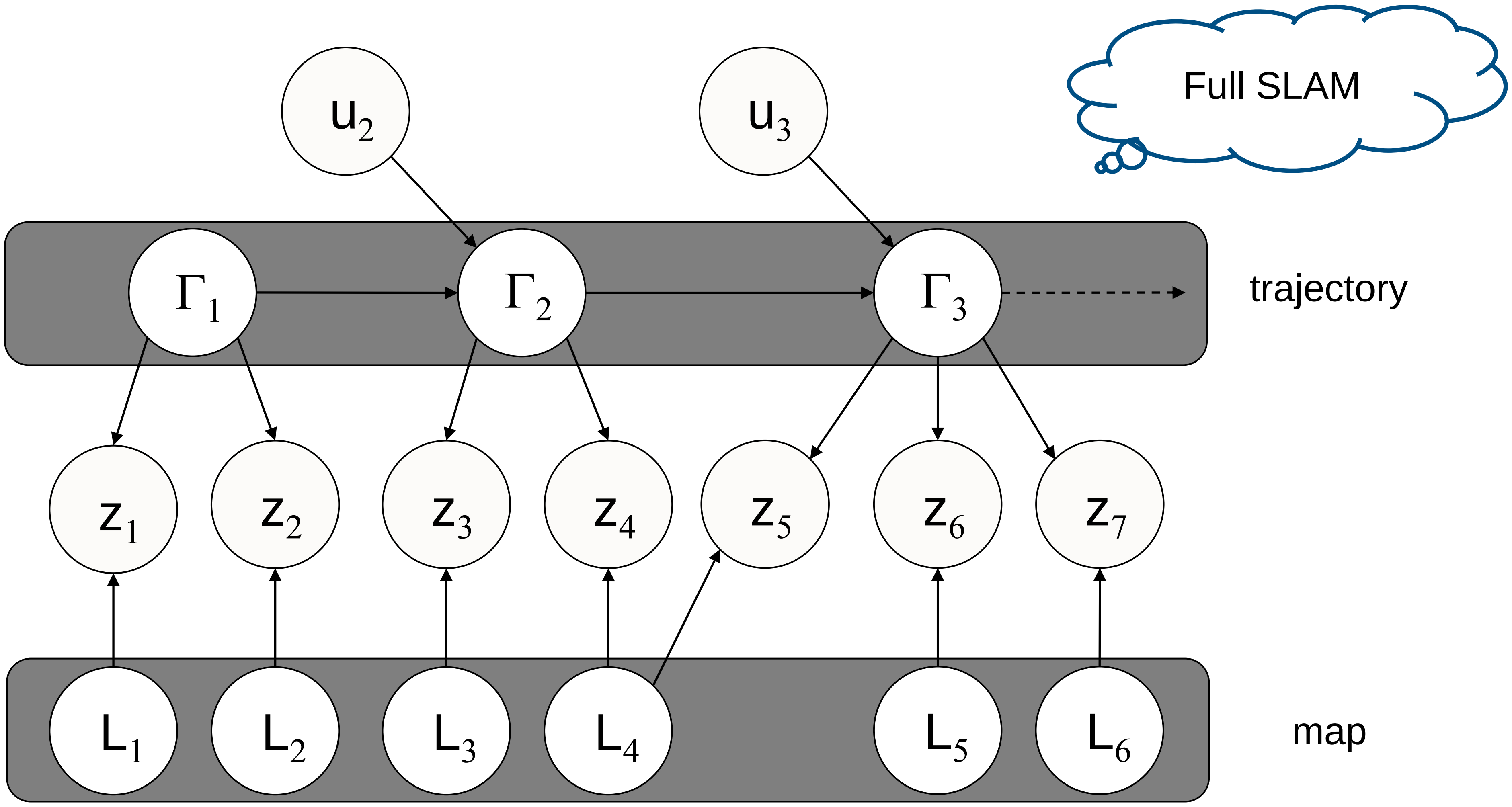
The compute the map $\hat{m}^{[t]}$ according to “mapping with known poses” based on the new pose and current observations.

Iterate alternating the two steps of localization and mapping …

Simultaneous Localization and Mapping

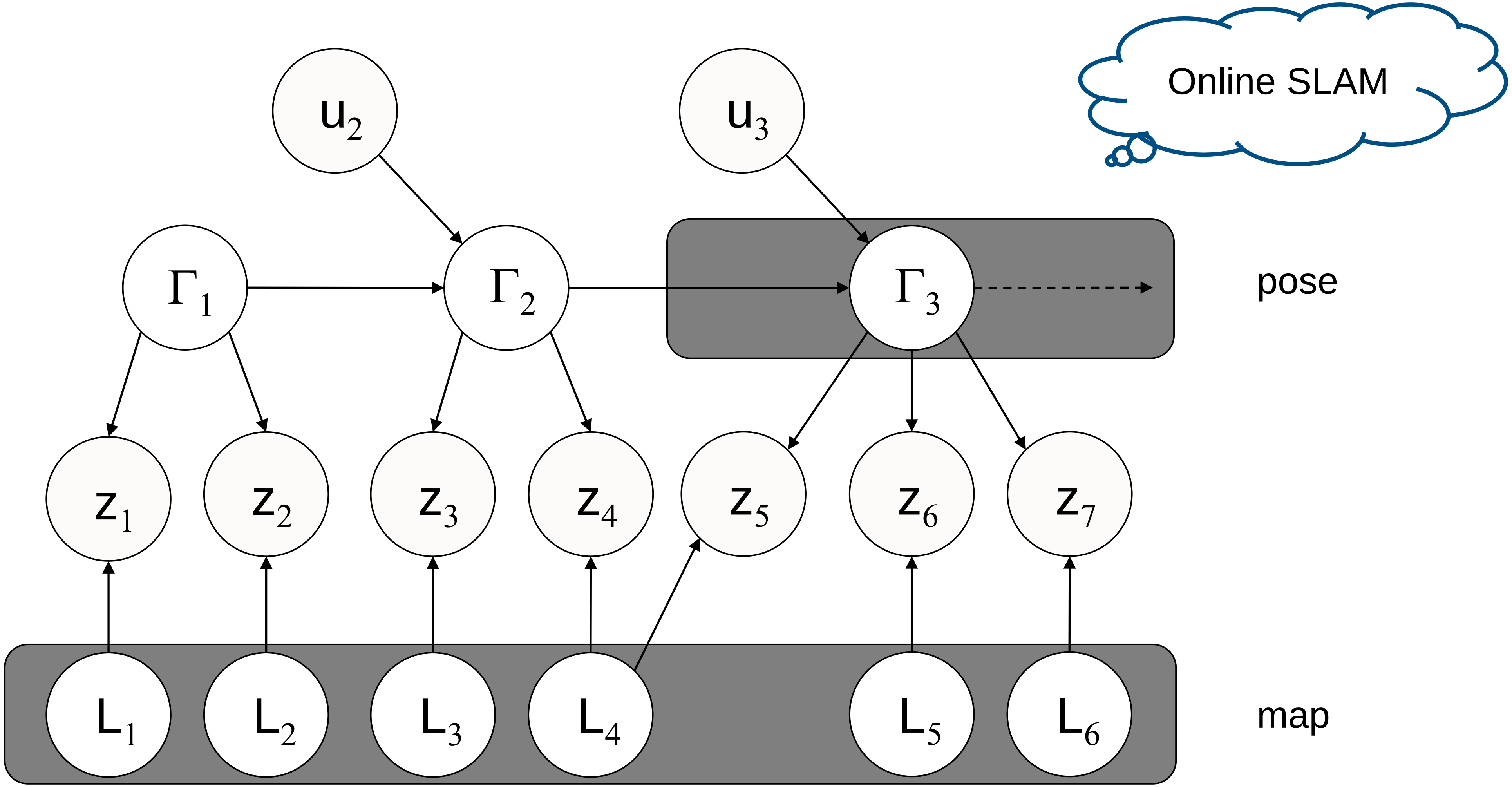


Dynamic Bayesian Networks and (Full) SLAM



$$\text{Smoothing} \quad : \quad p(\Gamma_{1:t}, l_1, \ldots, l_N \mid Z_{1:t}, U_{1:t})$$

Dynamic Bayesian Networks and (Online) SLAM



$$\text{Filtering} \quad : \quad p(\Gamma_t, l_1, \ldots, l_N \mid Z_{1:t}, U_{1:t}) = \iiint_{1:t-1} p(\Gamma_{1:t}, l_1, \ldots, l_N \mid Z_{1:t}, U_{1:t})$$

SLAM: Simultaneous Localization and Mapping

Full SLAM: $p(x_{1:t}, m \mid z_{1:t}, u_{1:t})$

Simultaneous estimate of path and map

Integrals computed one at the time

Online SLAM: $p(x_t, m \mid z_{1:t}, u_{1:t}) = \int\int \ldots \int p(x_{1:t}, m \mid z_{1:t}, u_{1:t})\, dx_1 dx_2 \ldots dx_{t-1}$

Simultaneous estimate of most recent pose and map

SLAM: Simultaneous Localization and Mapping

Full SLAM: $p(x_{1:t}, m \mid z_{1:t}, u_{1:t})$

ited

e

Online SLAM: p x_{t-1}

Two famous examples!

Extended Kalman Filter (EKF) SLAM

- Uses a linearized Gaussian probability distribution
- Solves the Online SLAM problem

FastSLAM

- Uses a sampled particle filter distribution model
- Solves the Full SLAM problem

Map with N landmarks:(3+2N)-dimensional Gaussian

Pose estimate

The map is part of the state

$$
Bel(x_t, m_t) = \left\langle \begin{pmatrix} x \\ y \\ \theta \\ l_1 \\ l_2 \\ \vdots \\ l_N \end{pmatrix}, \begin{pmatrix}
\sigma_x^2 & \sigma_{xy} & \sigma_{x\theta} & \sigma_{xl_1} & \sigma_{xl_2} & \cdots & \sigma_{xl_N} \\
\sigma_{xy} & \sigma_y^2 & \sigma_{y\theta} & \sigma_{yl_1} & \sigma_{yl_2} & \cdots & \sigma_{yl_N} \\
\sigma_{x\theta} & \sigma_{y\theta} & \sigma_\theta^2 & \sigma_{\theta l_1} & \sigma_{\theta l_2} & \cdots & \sigma_{\theta l_N} \\
\sigma_{xl_1} & \sigma_{yl_1} & \sigma_{\theta l_1} & \sigma_{l_1}^2 & \sigma_{l_1 l_2} & \cdots & \sigma_{l_1 l_N} \\
\sigma_{xl_2} & \sigma_{yl_2} & \sigma_{\theta l_2} & \sigma_{l_1 l_2} & \sigma_{l_2}^2 & \cdots & \sigma_{l_2 l_N} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
\sigma_{xl_N} & \sigma_{yl_N} & \sigma_{\theta l_N} & \sigma_{l_1 l_N} & \sigma_{l_2 l_N} & \cdots & \sigma_{l_N}^2
\end{pmatrix} \right\rangle
$$

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t$$

$$p(x_t | u_t, x_{t-1}) = N(A_t x_{t-1} + B_t u_t, R_t)$$

(E)KF-SLAM

Map with N landmarks:(3+2N)-dimensional Gaussian

Pose and map features correlate (and mesurements correct both)

Pose estimate

The map is part of the state

$$
Bel(x_t, m_t) = \left\langle \begin{pmatrix} x \\ y \\ \theta \\ l_1 \\ l_2 \\ \vdots \\ l_N \end{pmatrix}, \begin{pmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{x\theta} & \sigma_{xl_1} & \sigma_{xl_2} & \cdots & \sigma_{xl_N} \\ \sigma_{xy} & \sigma_y^2 & \sigma_{y\theta} & \sigma_{yl_1} & \sigma_{yl_2} & \cdots & \sigma_{yl_N} \\ \sigma_{x\theta} & \sigma_{y\theta} & \sigma_\theta^2 & \sigma_{\theta l_1} & \sigma_{\theta l_2} & \cdots & \sigma_{\theta l_N} \\ \sigma_{xl_1} & \sigma_{yl_1} & \sigma_{\theta l_1} & \sigma_{l_1}^2 & \sigma_{l_1 l_2} & \cdots & \sigma_{l_1 l_N} \\ \sigma_{xl_2} & \sigma_{yl_2} & \sigma_{\theta l_2} & \sigma_{l_1 l_2} & \sigma_{l_2}^2 & \cdots & \sigma_{l_2 l_N} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \sigma_{xl_N} & \sigma_{yl_N} & \sigma_{\theta l_N} & \sigma_{l_1 l_N} & \sigma_{l_2 l_N} & \cdots & \sigma_{l_N}^2 \end{pmatrix} \right\rangle
$$

$$
z_t = C_t x_t + \delta_t
$$

$$
p(z_t | x_t) = N(C_t x_t, Q_t)
$$

Bayes Filter: The Algorithm

$$\boxed{Bel(x_t) = \eta \; P(z_t \mid x_t) \int P(x_t \mid u_t, x_{t-1}) \; Bel(x_{t-1}) \; dx_{t-1}}$$

Algorithm Bayes_filter(*Bel(x), d*):

If *d* is a perceptual data item *z* then

For all *x* do

$$Bel'(x) = P(z \mid x) Bel(x)$$

correction

Else if *d* is an action data item *u* then

For all *x* do

$$Bel'(x) = \int P(x \mid u, x') \; Bel(x') \; dx'$$

prediction

Return *Bel'(x)*

Kalman Filter Algorithm

Algorithm Kalman_filter(μ_{t-1}, Σ_{t-1}, u_t, z_t):

Prediction:
$$\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$$
$$\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$$

Correction:
$$K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$$
$$\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$$
$$\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$$

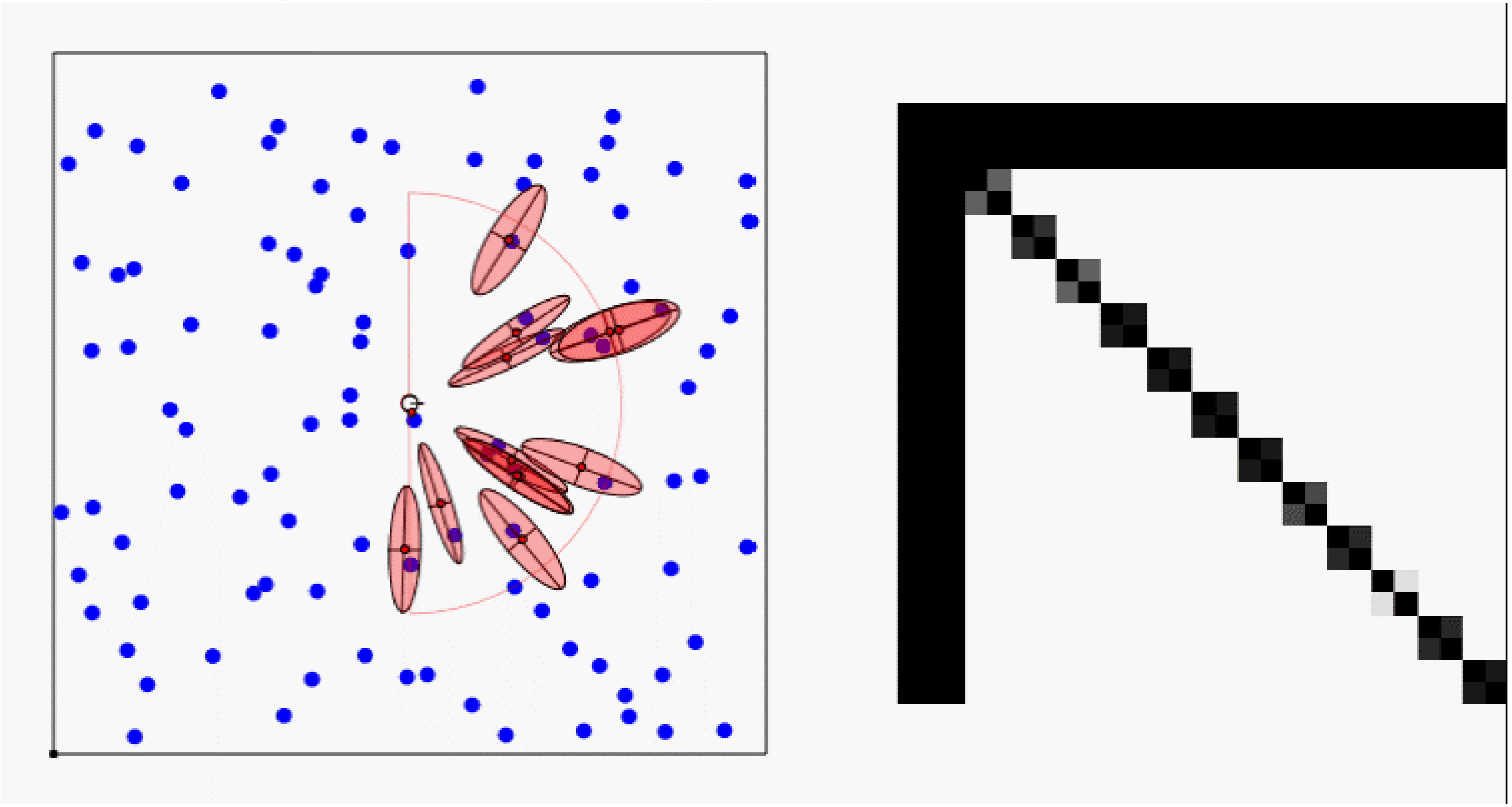
Return μ_t, Σ_t

Not much different from standard EKF ... but the state dimention increases!!

$$Bel(x_t, m_t) = \left\langle \begin{pmatrix} x \\ y \\ \theta \\ l_1 \\ l_2 \\ \vdots \\ l_N \end{pmatrix}, \begin{pmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{x\theta} & \sigma_{xl_1} & \sigma_{xl_2} & \cdots & \sigma_{xl_N} \\ \sigma_{xy} & \sigma_y^2 & \sigma_{y\theta} & \sigma_{yl_1} & \sigma_{yl_2} & \cdots & \sigma_{yl_N} \\ \sigma_{x\theta} & \sigma_{y\theta} & \sigma_\theta^2 & \sigma_{\theta l_1} & \sigma_{\theta l_2} & \cdots & \sigma_{\theta l_N} \\ \sigma_{xl_1} & \sigma_{yl_1} & \sigma_{\theta l_1} & \sigma_{l_1}^2 & \sigma_{l_1 l_2} & \cdots & \sigma_{l_1 l_N} \\ \sigma_{xl_2} & \sigma_{yl_2} & \sigma_{\theta l_2} & \sigma_{l_1 l_2} & \sigma_{l_2}^2 & \cdots & \sigma_{l_2 l_N} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \sigma_{xl_N} & \sigma_{yl_N} & \sigma_{\theta l_N} & \sigma_{l_1 l_N} & \sigma_{l_2 l_N} & \cdots & \sigma_{l_N}^2 \end{pmatrix} \right\rangle$$

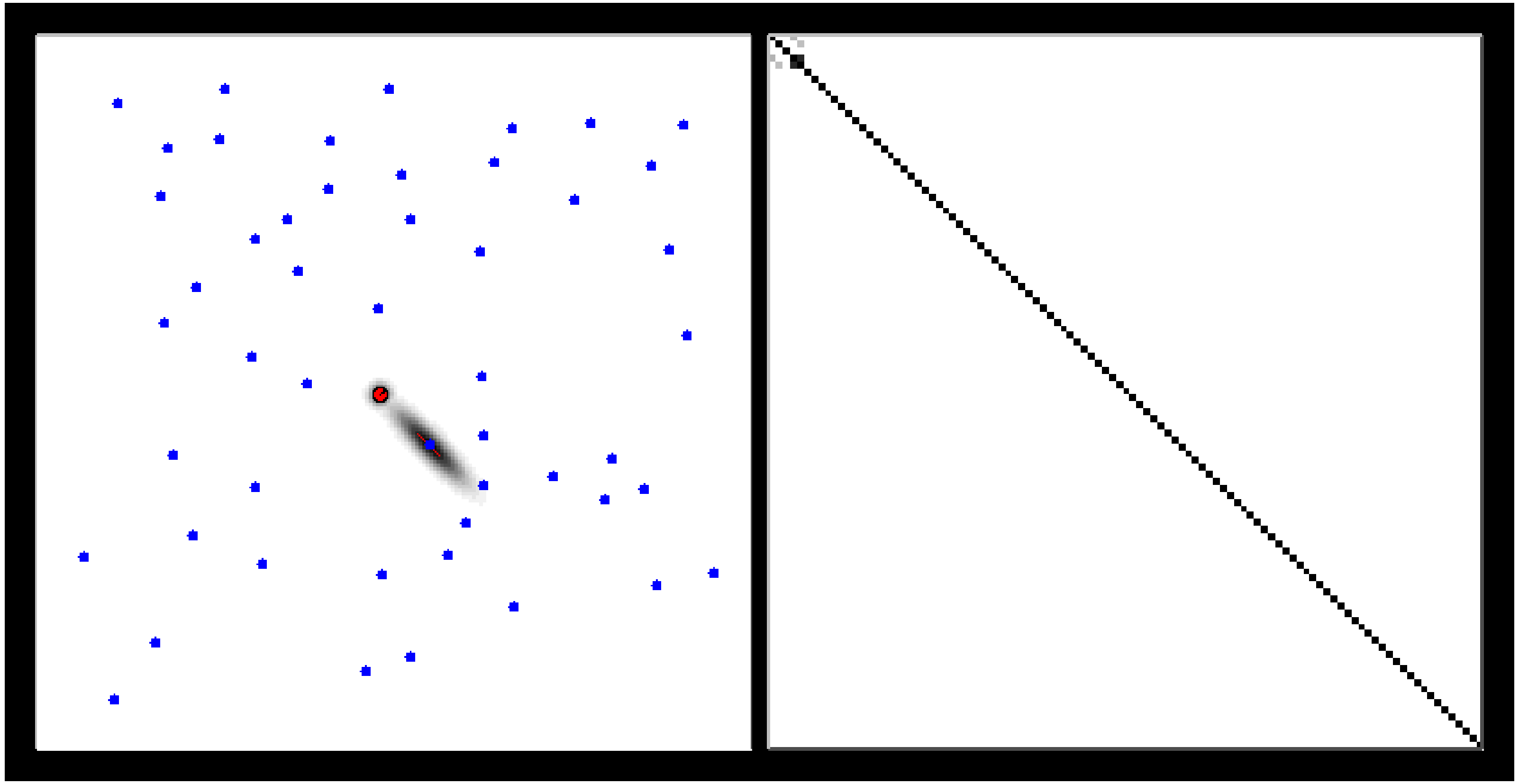
Classical Solution – The EKF

Approximate the SLAM posterior with a high-dimensional Gaussian



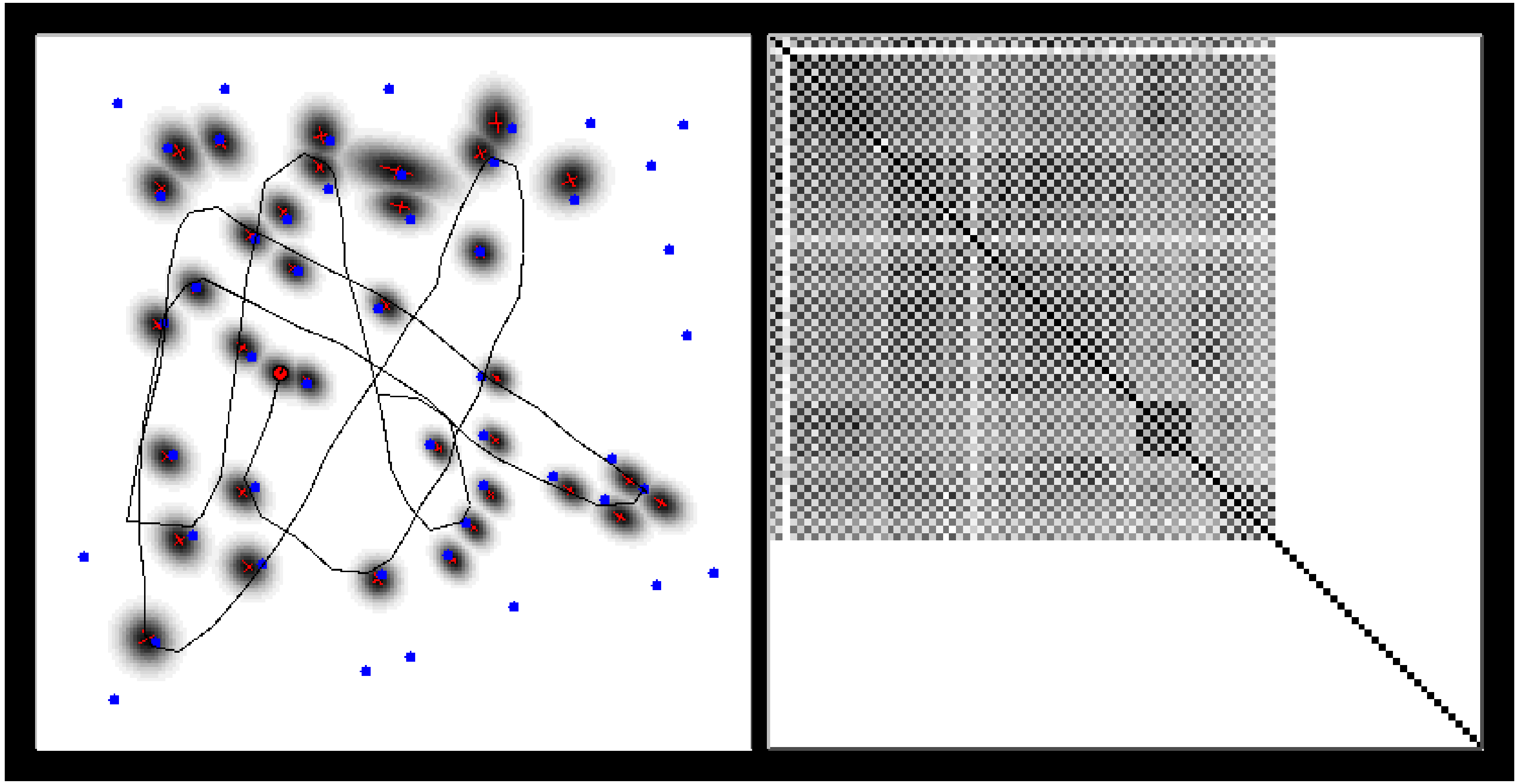
Blue path = true path **Red path** = estimated path **Black path** = odometry

EKF-SLAM



Map Correlation matrix

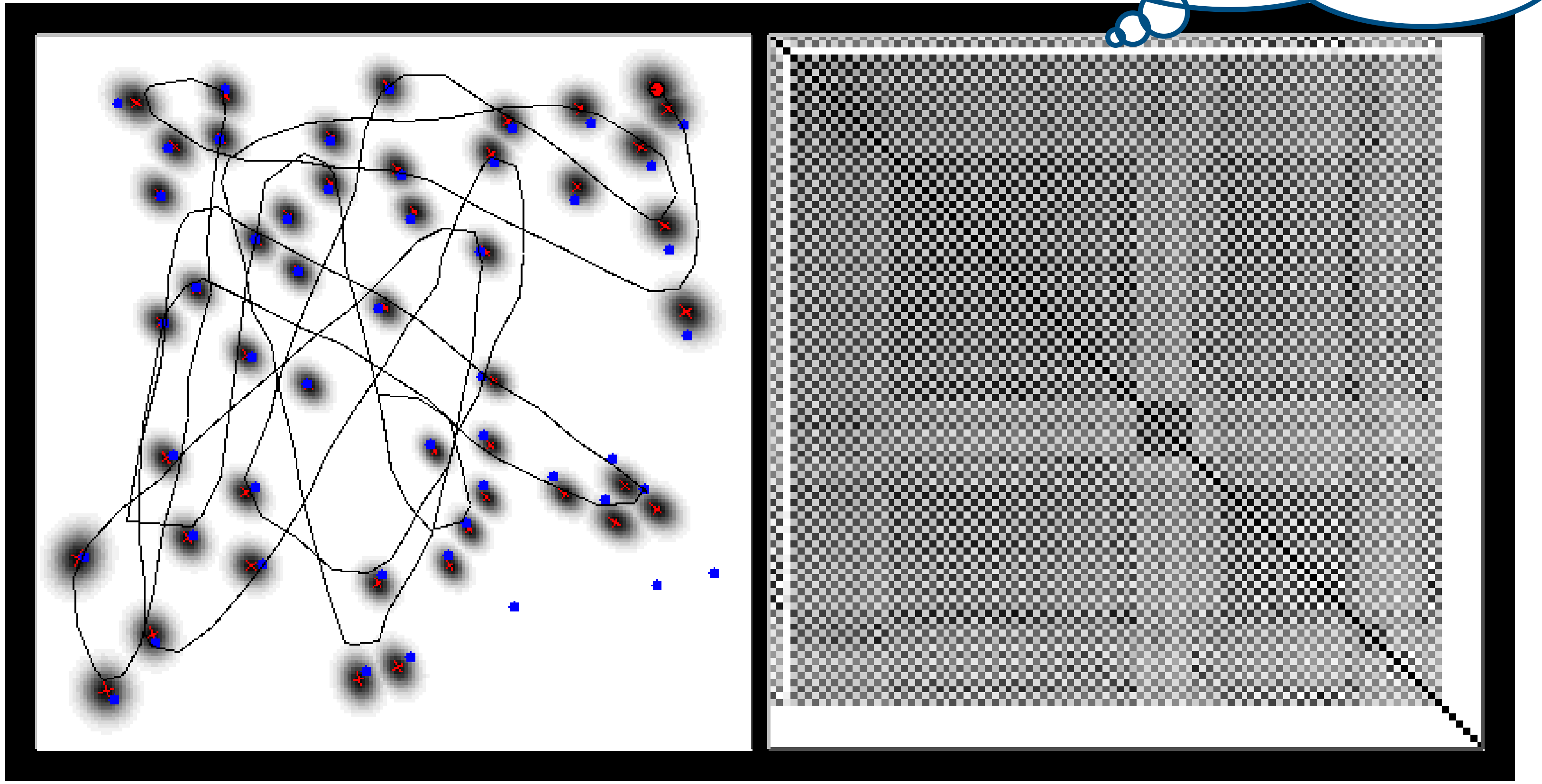
EKF-SLAM



Map Correlation matrix

EKF-SLAM

Landmark positions uncorrelated with the robot orientation ...



Map Correlation matrix

Properties of KF-SLAM (Linear Case)

Theorem: The determinant of any sub-matrix of the map covariance matrix decreases monotonically as successive observations are made.

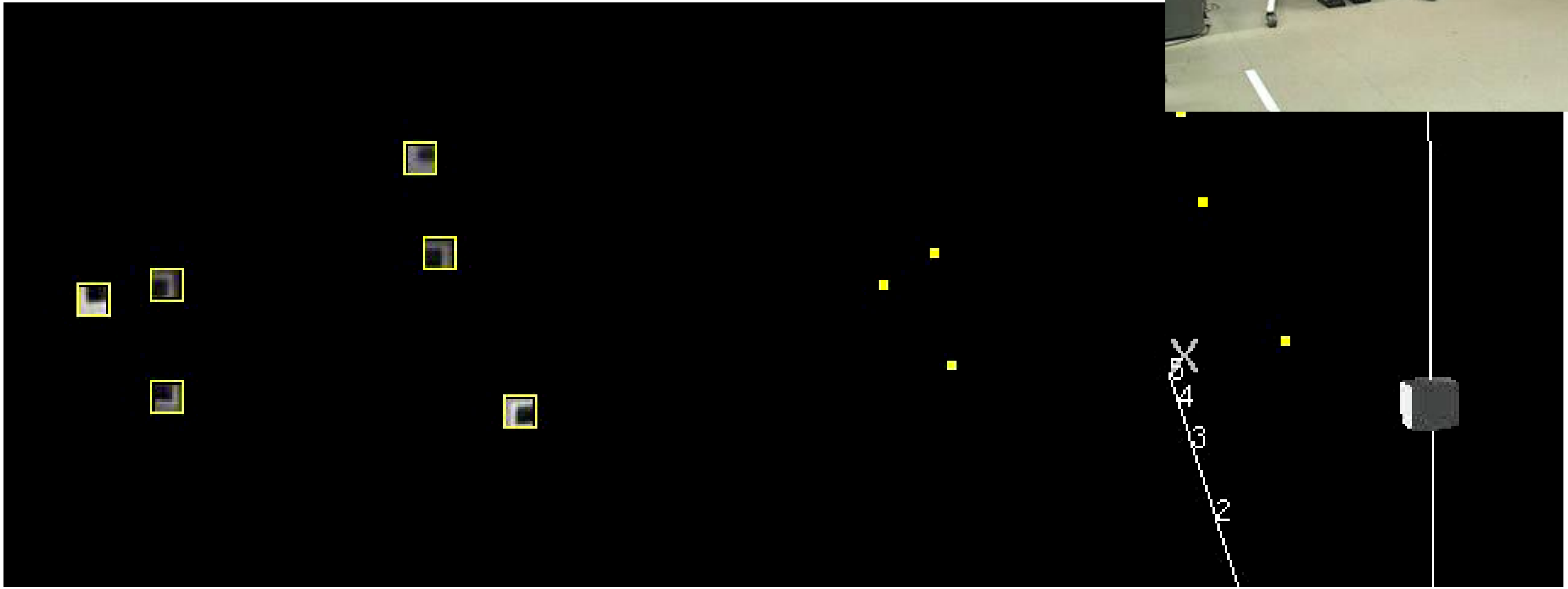
Theorem: In the limit the landmark estimates become fully correlated

[Dissanayake et al., 2001]

Are we happy about this?

- Quadratic in the number of landmarks: $O(n^2)$
- Convergence results for the linear case
- Can diverge if nonlinearities are large!
- Have been applied successfully in large-scale environments.
- Approximations reduce the computational complexity.

Monocular SLAM Origins …



Monocular SLAM Origins …

Larger size environments …

Beyond EKF-SLAM

EKF-SLAM works pretty well but ...

- EKF-SLAM employs linearized models of nonlinear motion and observation models and so inherits many caveats.
- Computational effort is demand because computation grows quadratically with the number of landmarks.

Possible solutions

- Local submaps [Leonard & al 99, Bosse & al 02, Newman & al 03]
- Sparse links (correlations) [Lu & Milios 97, Guivant & Nebot 01]
- Sparse extended information filters [Frese et al. 01, Thrun et al. 02]
- Rao-Blackwellisation (FastSLAM) [Murphy 99, Montemerlo et al. 02, ...]
  - Represents nonlinear process and non-Gaussian uncertainty
  - Rao-Blackwellized method reduces computation

Our Full SLAM solution

The FastSLAM Idea (Full SLAM)

In the general case we have

$$p(x_t, m \mid z_t) \neq P(x_t \mid z_t) P(m \mid z_t)$$

However if we consider the full trajectory X_t rather than the single pose x_t

$$p(X_t, m \mid z_t) = P(X_t \mid z_t) P(m \mid X_t, z_t)$$

In FastSLAM, the trajectory X_t is represented by particles $X_t(i)$ while the map is represented by a factorization called Rao-Blackwellized Filter

- $P(X_t|z_t)$ through particles
- $P(m|X_t^{(i)}, z_t)$ using an EKF

$$P(\underset{\text{map}}{m} \mid \underset{\text{poses}}{X_t^{(i)}}, z_t) = \prod_j^M P(\underset{\text{landmarks / grid cells}}{m_j} \mid X_t^{(i)}, z_t)$$

FastSLAM Formulation

Decouple map of features from poses ...

- Each particle represents a robot trajectory
- Feature measurements are correlated thought the robot trajectory
- If the robot trajectory is known all of the features would be uncorrelated
- Treat each pose particle as if it is the true trajectory, processing all of the feature measurements independently

poses map observations & movements

$$\underbrace{p(x_{1:t}, l_{1:m} \mid z_{1:t}, u_{0:t-1})}_{\text{SLAM posterior}} = \underbrace{p(x_{1:t} \mid z_{1:t}, u_{0:t-1})}_{\text{Robot path posterior}} \cdot \underbrace{p(l_{1:m} \mid x_{1:t}, z_{1:t})}_{\text{Landmark positions}}$$

Factored Posterior: Rao-Blackwellization

$$
\begin{aligned}
& p(x_{1:t}, l_{1:m} \mid z_{1:t}, u_{0:t-1}) \\
& \quad = \; p(x_{1:t} \mid z_{1:t}, u_{0:t-1}) \cdot p(l_{1:m} \mid x_{1:t}, z_{1:t}) \\
& \quad = \; p(x_{1:t} \mid z_{1:t}, u_{0:t-1}) \cdot \prod_{i=1}^{M} p(l_i \mid x_{1:t}, z_{1:t})
\end{aligned}
$$

Robot path posterior
(localization problem)

Conditionally independent
landmark positions

Dimension of state space is reduced by factorization making particle filtering possible

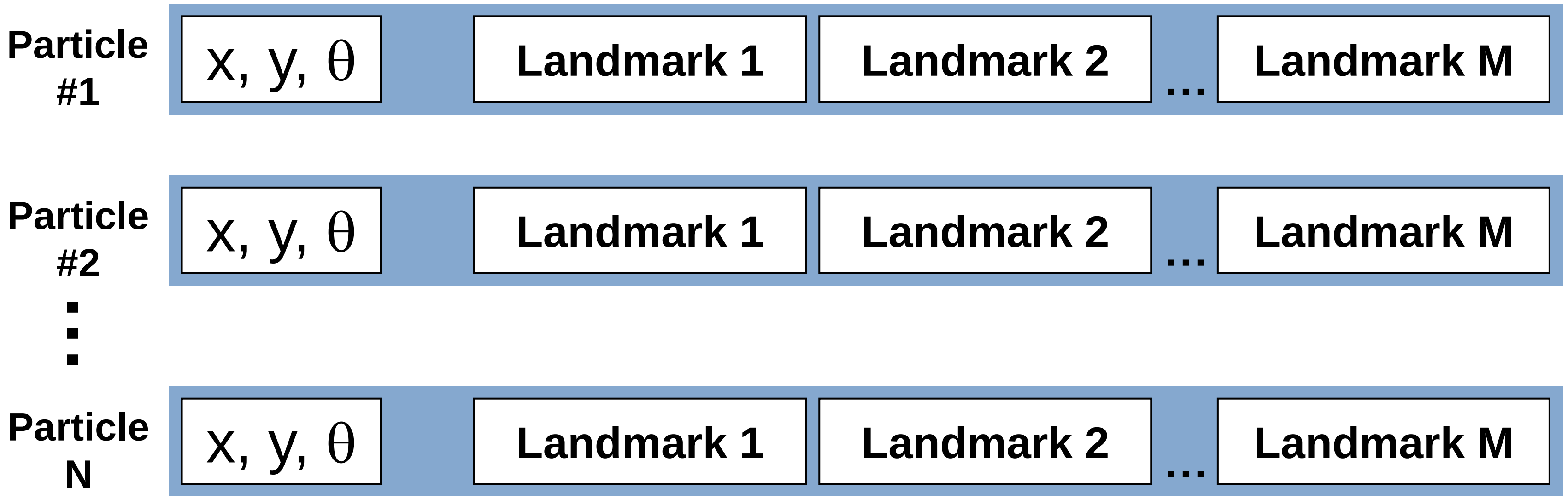
$$
p(x_{1:t}, l_{1:m} \mid z_{1:t}, u_{0:t-1}) =
$$

$$
p(x_{1:t} \mid z_{1:t}, u_{0:t-1}) \cdot \prod_{i=1}^{M} p(l_i \mid x_{1:t}, z_{1:t})
$$

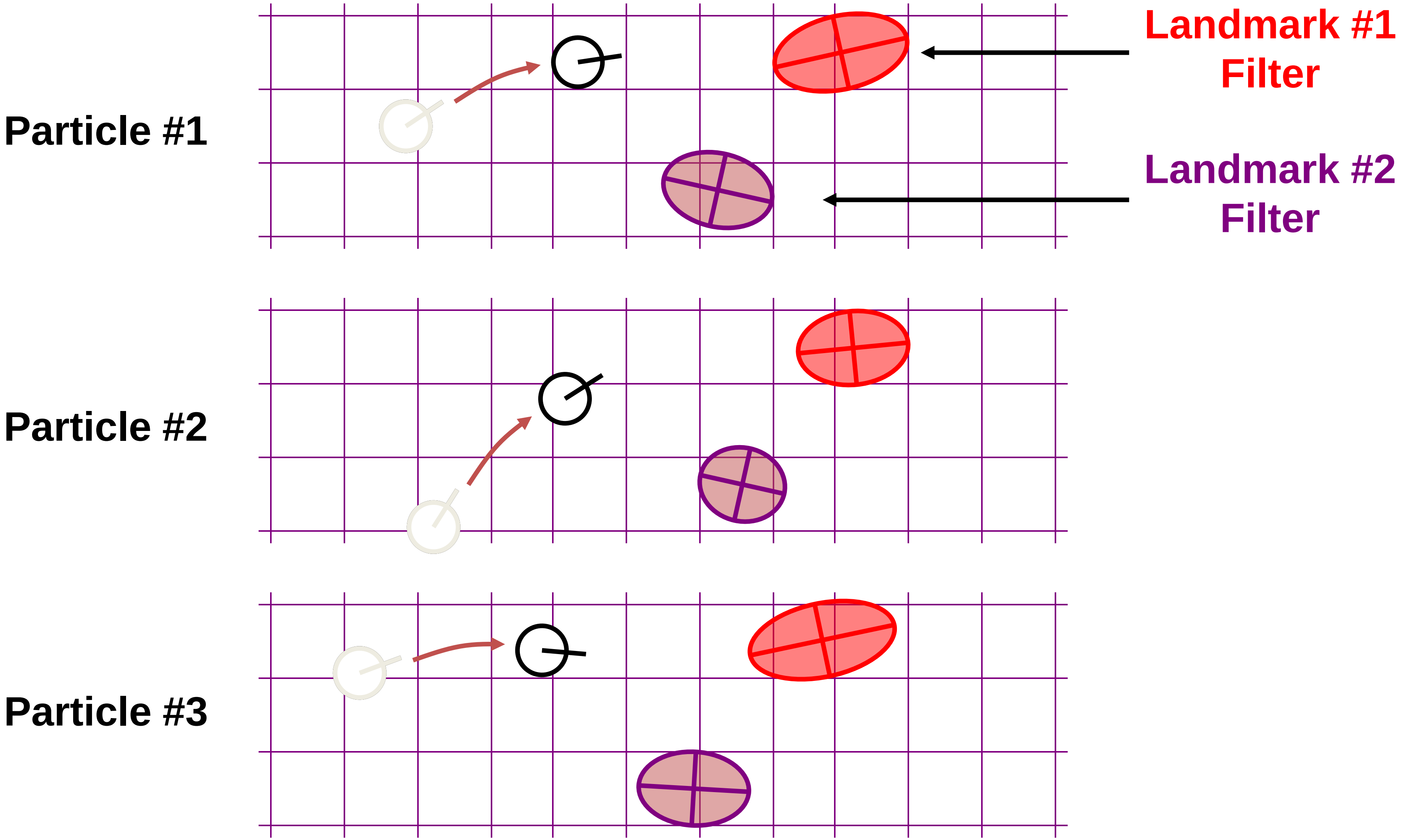
FastSLAM in Practice

Rao-Blackwellized particle filtering based on landmarks [Montemerlo et al., 2002]

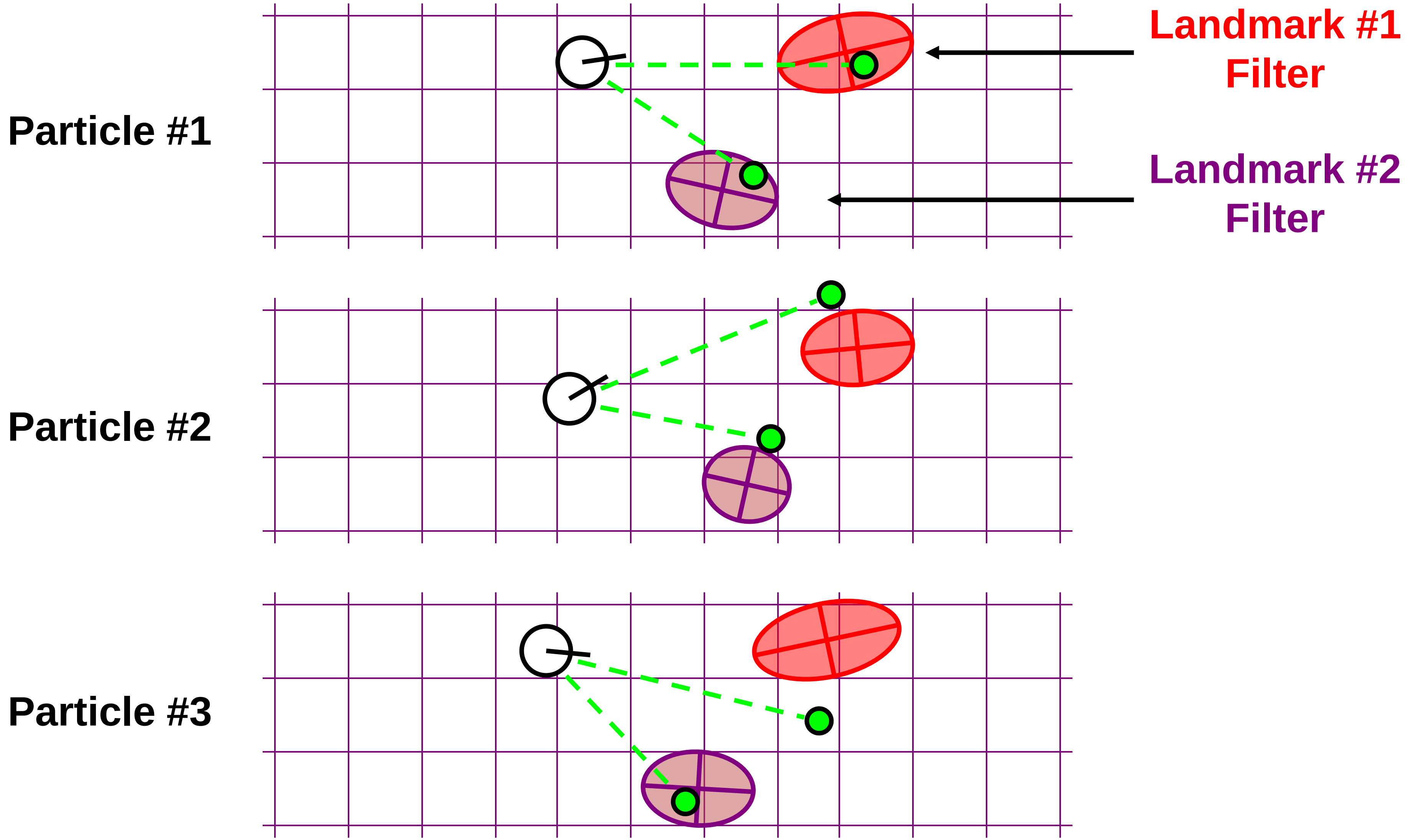
- Each particle is a trajectory (last pose + reference to previous)
- Each landmark is represented by a 2x2 Extended Kalman Filter (EKF)
- Each particle therefore has to maintain M EKFs



FastSLAM – Action Update

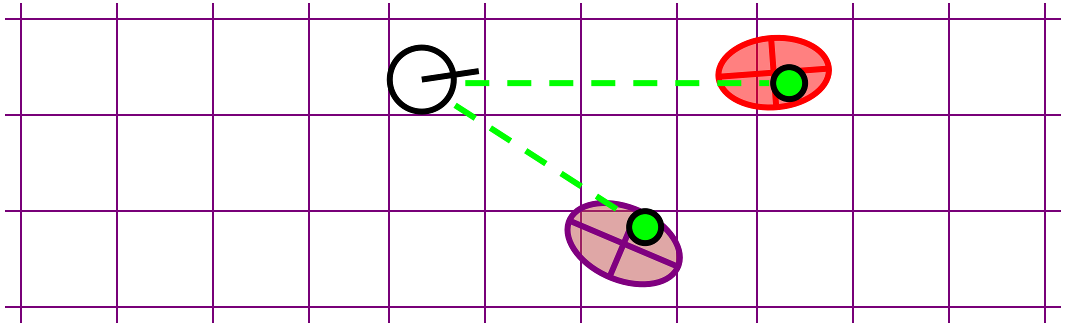


FastSLAM – Sensor Update



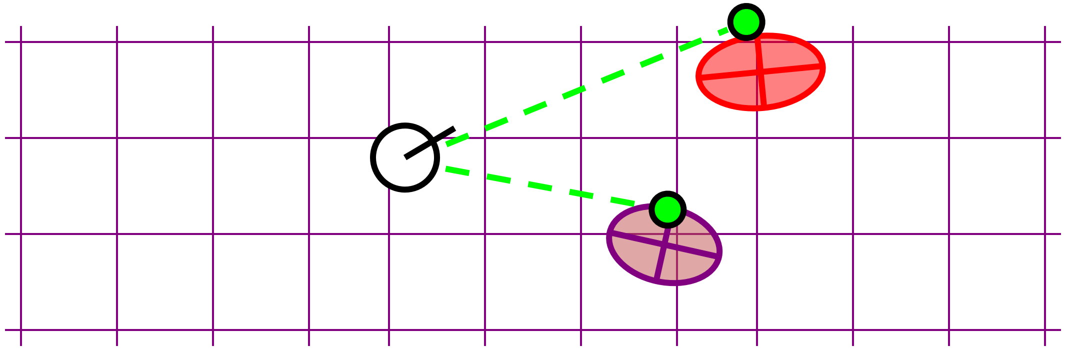
FastSLAM – Sensor Update

Particle #1



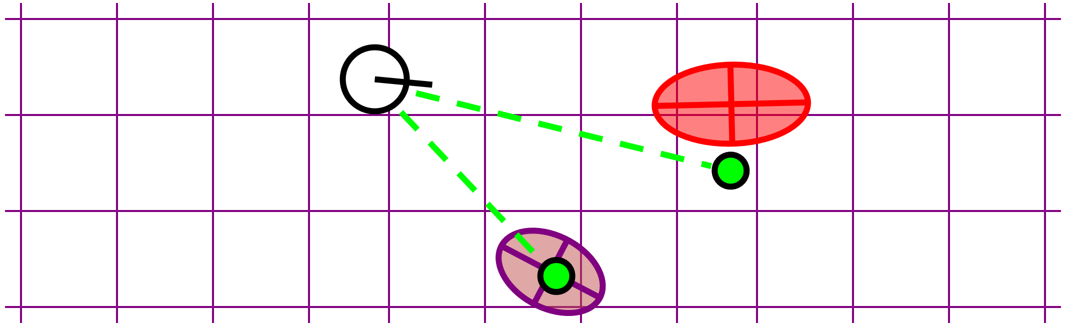
Weight = 0.8

Particle #2



Weight = 0.4

Particle #3



Weight = 0.1

FastSLAM Complexity

Update robot particles based on control u_{t-1} $O(N)$ **Constant time per particle**

Incorporate observation z_t into Kalman filters $O(N \cdot \log(M))$ **Log time per particle**

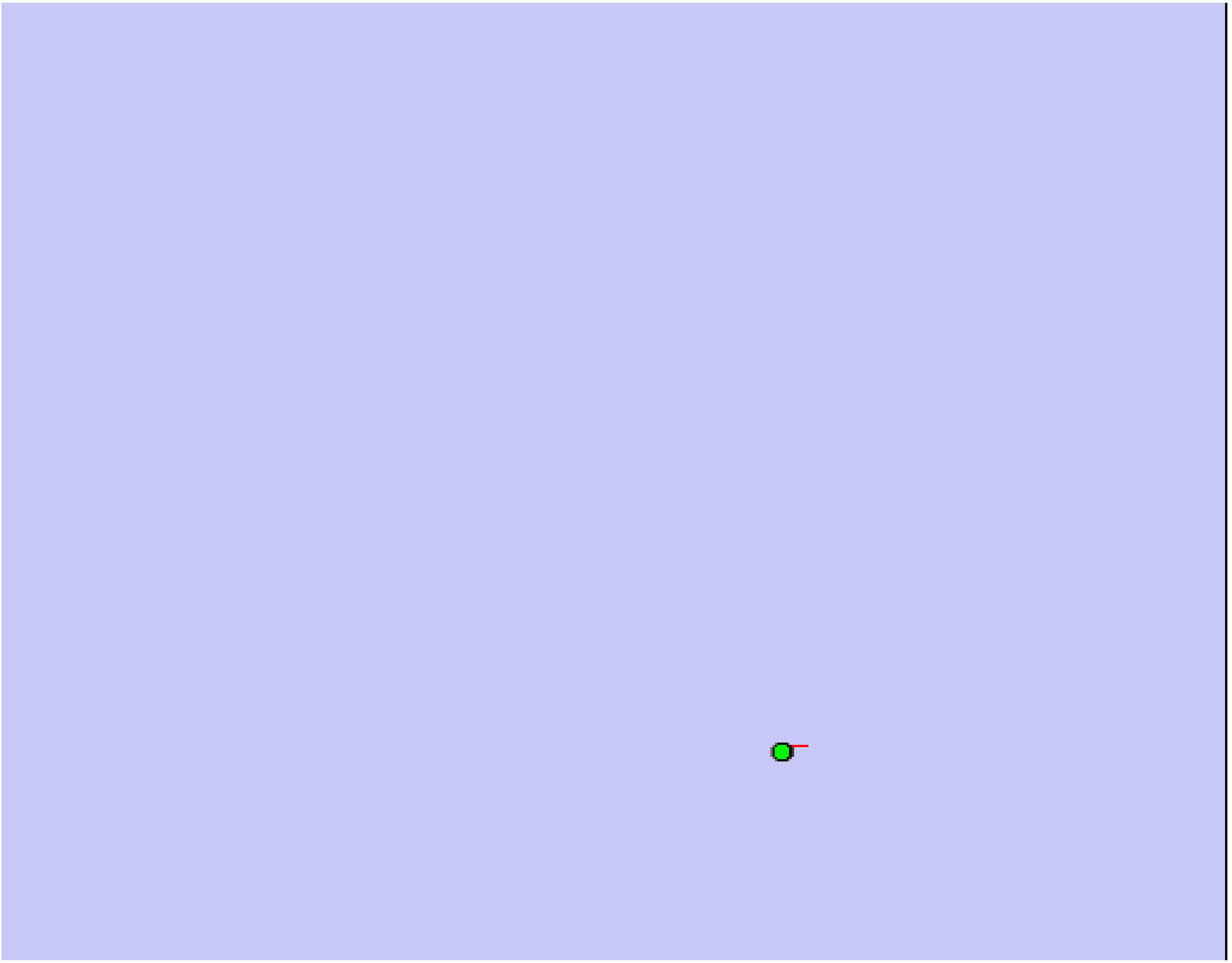
Resample particle set $O(N \cdot \log(M))$ **Log time per particle**

$O(N \cdot \log(M))$

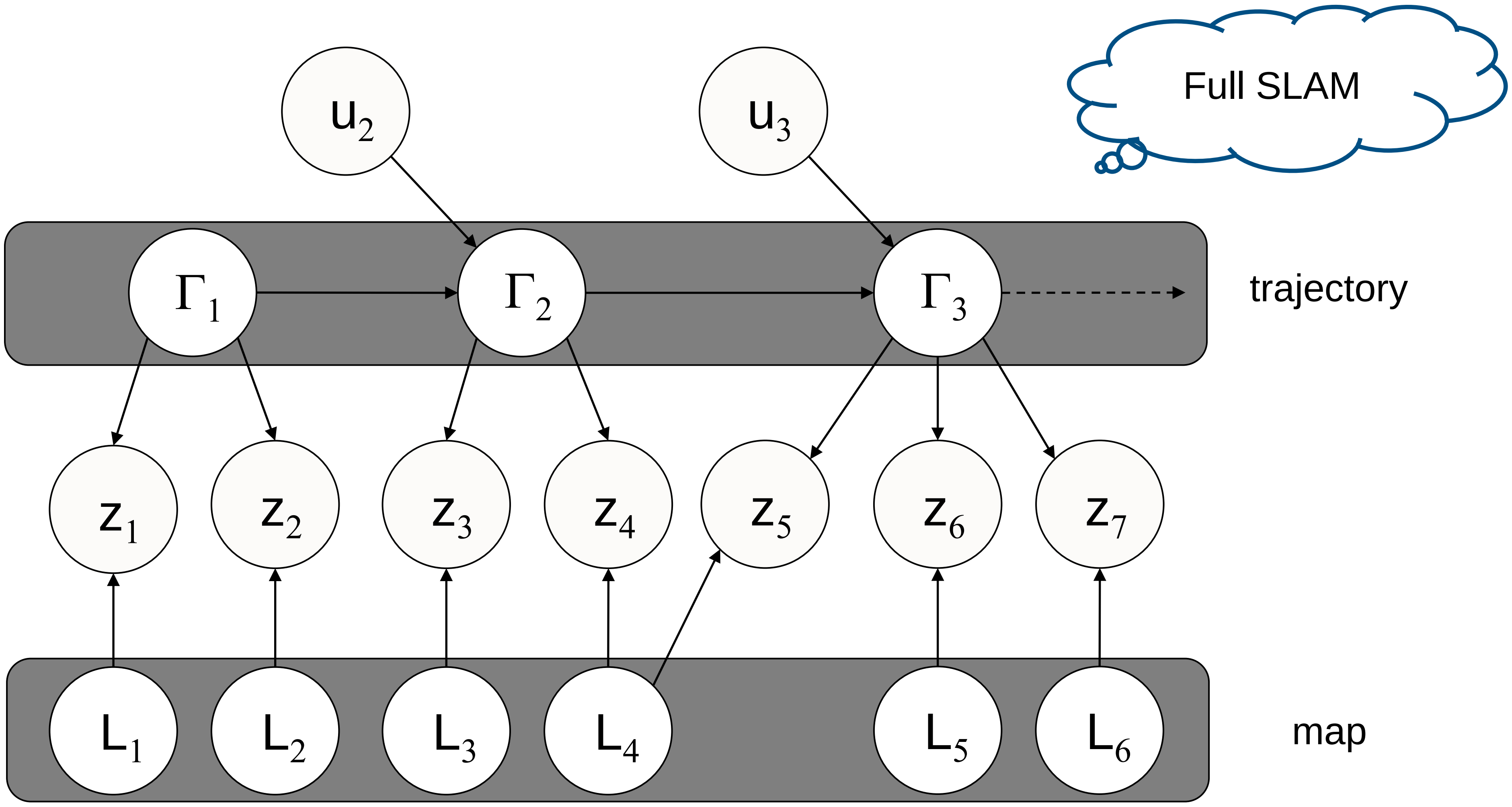
Log time per particle

N = Number of particles
M = Number of map features

Fast-SLAM Example



Dynamic Bayesian Networks and (Full) SLAM



Smoothing: $p(\Gamma_{1:t}, L_1, \ldots, L_N | Z_{1:t}, U_{1:t})$

Bayesian Networks and Maximum a Posteriori

In Full SLAM we model

$X = \Gamma_{0:t}, L_1, \dots, L_n$

$$p(X|Z,U) = p(\Gamma_{0:t}, L_1, \dots, L_n | z_{1:t}, u_{1:t})$$

then we look for the most likely solution

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

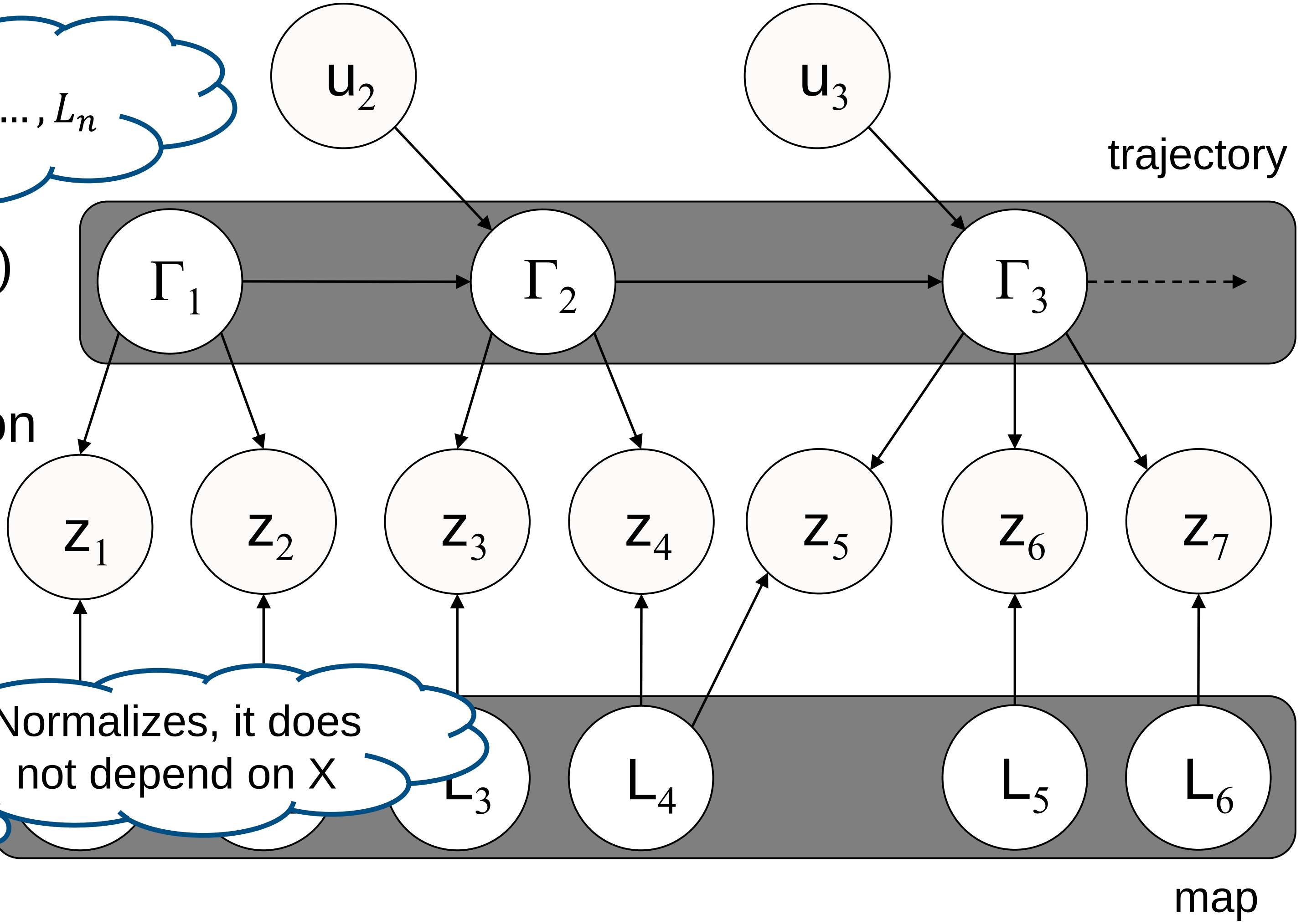
This can be rewritten as

$$X^{MAP} = \underset{X}{argmax}\; p(X,Z,U)$$

Normalizes, it does not depend on X

$$p(X|Z,U) = p(X,Z,U)/p(Z,U)$$

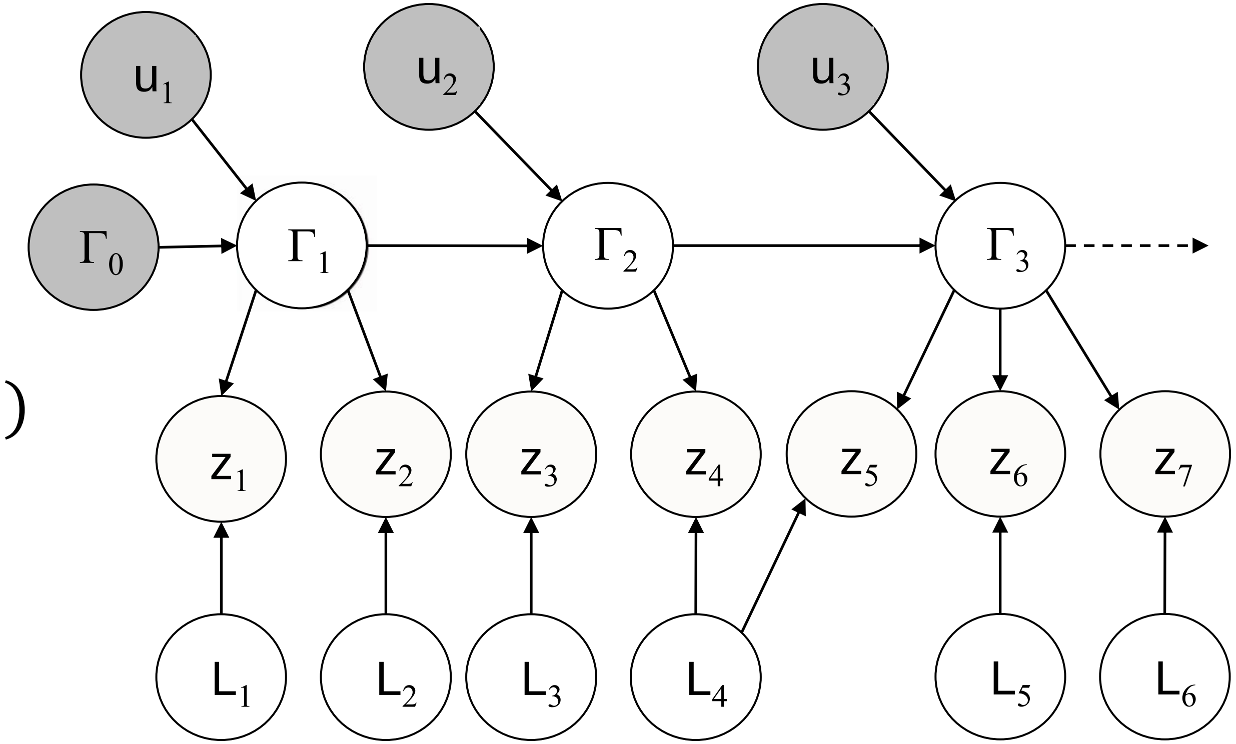
Full Joint Distribution



Smoothing: $p(\Gamma_{1:t}, L_1, \dots, L_N | Z_{1:t}, U_{1:t})$

Bayesian Networks and Joint Distribution

The full joint distribution of a Bayesian network is the product of the conditionals



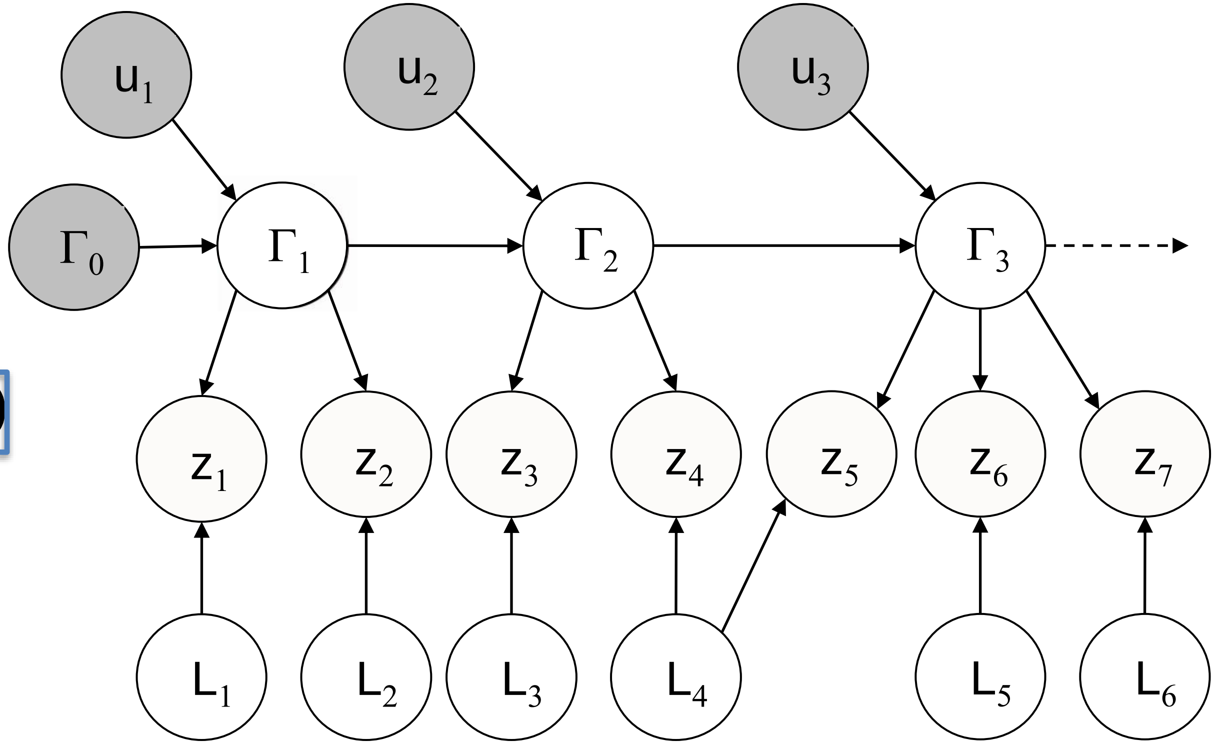
$$p(X,Z,U) = p(\Gamma_{0:3}, L_1, \dots, L_6 | z_{1:7}, u_{1:7})$$

$$= p(\Gamma_1|\Gamma_0,u_1)p(\Gamma_2|\Gamma_1,u_1)p(\Gamma_3|\Gamma_2,u_3)$$
$$p(Z_1|\Gamma_1,L_1)p(L_1)p(Z_2|\Gamma_1,L_2)p(L_2)$$
$$p(Z_3|\Gamma_2,L_3)p(L_3)p(Z_4|\Gamma_2,L_4)p(L_4)$$
$$p(Z_5|\Gamma_3,L_4)p(Z_6|\Gamma_3,L_5)p(L_5)$$
$$p(Z_7|\Gamma_3,L_6)p(L_6)$$

$$= \phi(\Gamma_1,\Gamma_0,u_1)\phi(\Gamma_2,\Gamma_1,u_1)\phi(\Gamma_3,\Gamma_2,u_3)\phi(Z_1,\Gamma_1,L_1)\phi(Z_2,\Gamma_1,L_2)$$
$$\phi(Z_3,\Gamma_2,L_3)\phi(Z_4,\Gamma_2,L_4)\phi(Z_5,\Gamma_3,L_4)\phi(Z_6,\Gamma_3,L_5)\phi(Z_7,\Gamma_3,L_6)$$

Bayesian Networks and Joint Distribution

The full joint distribution of a Bayesian network is the product of the conditionals



$$p(X,Z,U) = p(\Gamma_{0:3}, L_1, \ldots, L_6 | z_{1:7}, u_{1:7})$$

$$= \boxed{p(\Gamma_1|\Gamma_0,u_1)}\boxed{p(\Gamma_2|\Gamma_1,u_1)}\boxed{p(\Gamma_3|\Gamma_2,u_3)}$$
$$p(Z_1|\Gamma_1,L_1)p(L_1)p(Z_2|\Gamma_1,L_2)p(L_2)$$
$$p(Z_3|\Gamma_2,L_3)p(L_3)p(Z_4|\Gamma_2,L_4)p(L_4)$$
$$p(Z_5|\Gamma_3,L_4)p(Z_6|\Gamma_3,L_5)p(L_5)$$
$$p(Z_7|\Gamma_3,L_6)p(L_6)$$

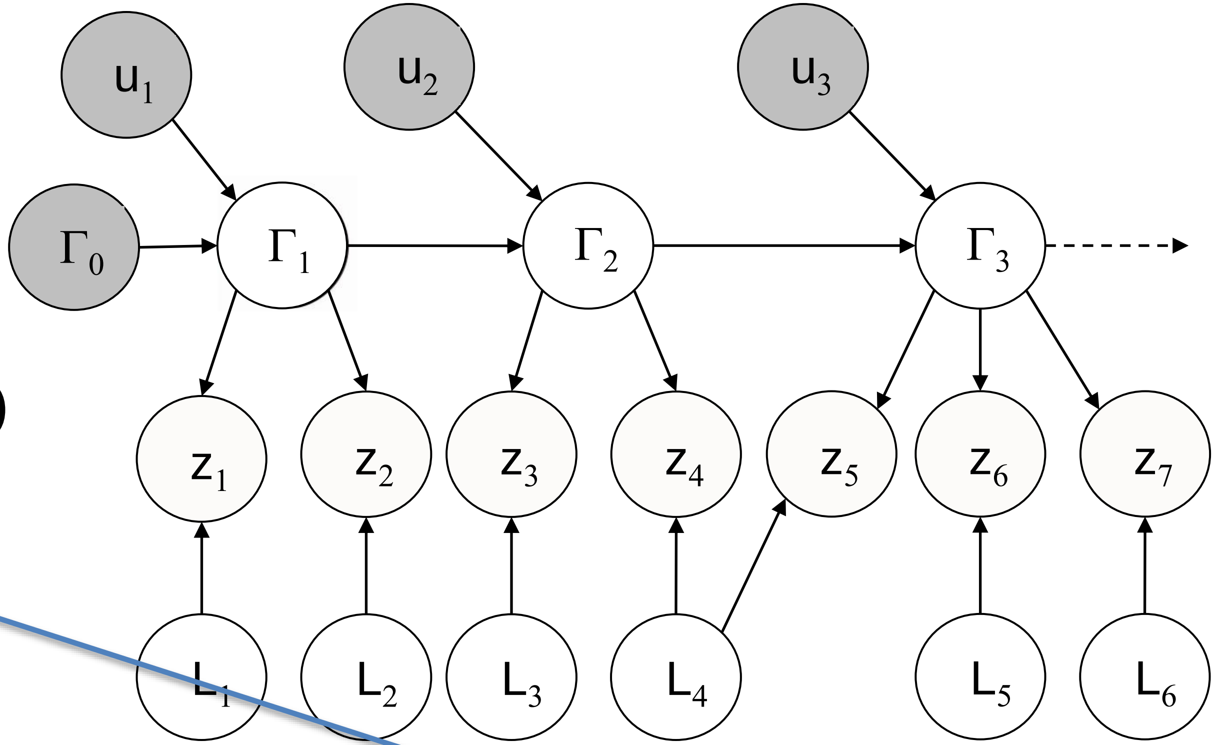
$$= \phi(\Gamma_1,\Gamma_0,u_1)\phi(\Gamma_2,\Gamma_1,u_1)\phi(\Gamma_3,\Gamma_2,u_3)\phi(Z_1,\Gamma_1,L_1)\phi(Z_2,\Gamma_1,L_2)$$
$$\phi(Z_3,\Gamma_2,L_3)\phi(Z_4,\Gamma_2,L_4)\phi(Z_5,\Gamma_3,L_4)\phi(Z_6,\Gamma_3,L_5)\phi(Z_7,\Gamma_3,L_6)$$

Bayesian Networks and Joint Distribution

The full joint distribution of a Bayesian network is the product of the conditionals

$$p(X,Z,U) = p(\Gamma_{0:3}, L_1, \dots, L_6 | z_{1:7}, u_{1:7})$$

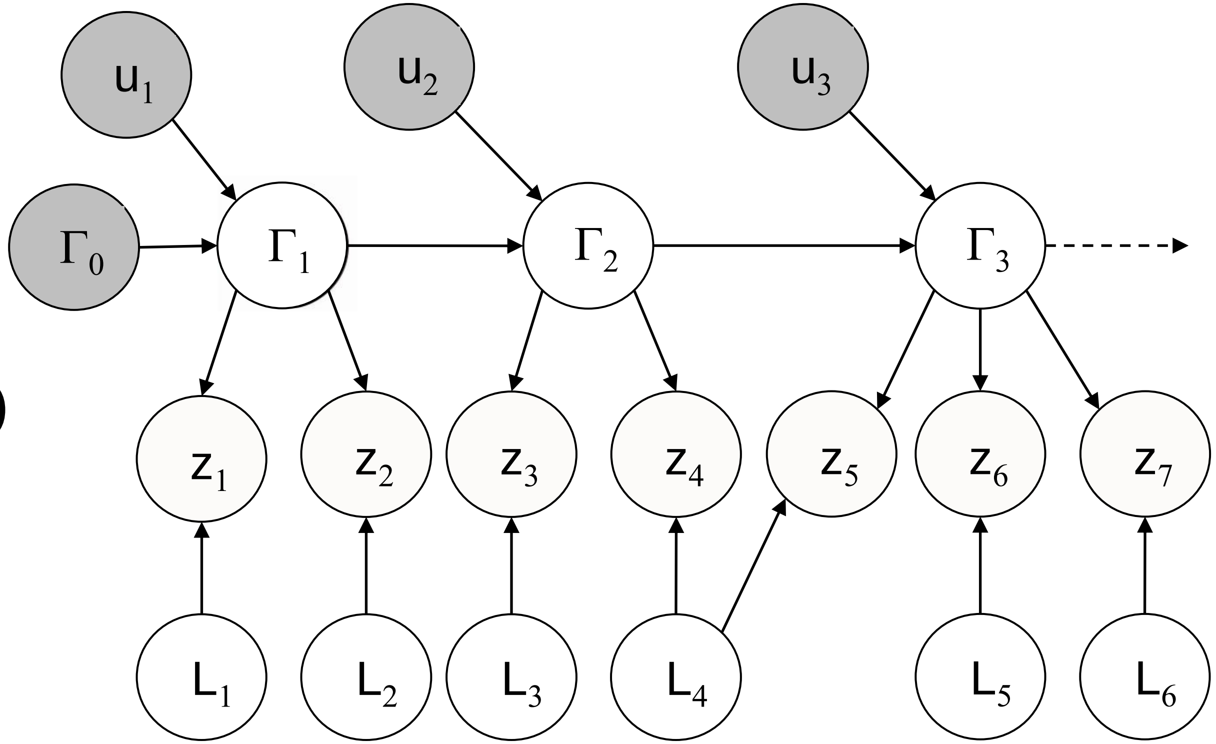
$$= p(\Gamma_1|\Gamma_0,u_1)p(\Gamma_2|\Gamma_1,u_1)p(\Gamma_3|\Gamma_2,u_3)$$
$$\boxed{p(Z_1|\Gamma_1,L_1)p(L_1)}\boxed{p(Z_2|\Gamma_1,L_2)p(L_2)}$$
$$p(Z_3|\Gamma_2,L_3)p(L_3)p(Z_4|\Gamma_2,L_4)p(L_4)$$
$$p(Z_5|\Gamma_3,L_4)p(Z_6|\Gamma_3,L_5)p(L_5)$$
$$p(Z_7|\Gamma_3,L_6)p(L_6)$$



$$= \phi(\Gamma_1,\Gamma_0,u_1)\phi(\Gamma_2,\Gamma_1,u_1)\phi(\Gamma_3,\Gamma_2,u_3)\phi(Z_1,\Gamma_1,L_1)\phi(Z_2,\Gamma_1,L_2)$$
$$\phi(Z_3,\Gamma_2,L_3)\phi(Z_4,\Gamma_2,L_4)\phi(Z_5,\Gamma_3,L_4)\phi(Z_6,\Gamma_3,L_5)\phi(Z_7,\Gamma_3,L_6)$$

Bayesian Networks and Joint Distribution

The full joint distribution of a Bayesian network is the product of the conditionals



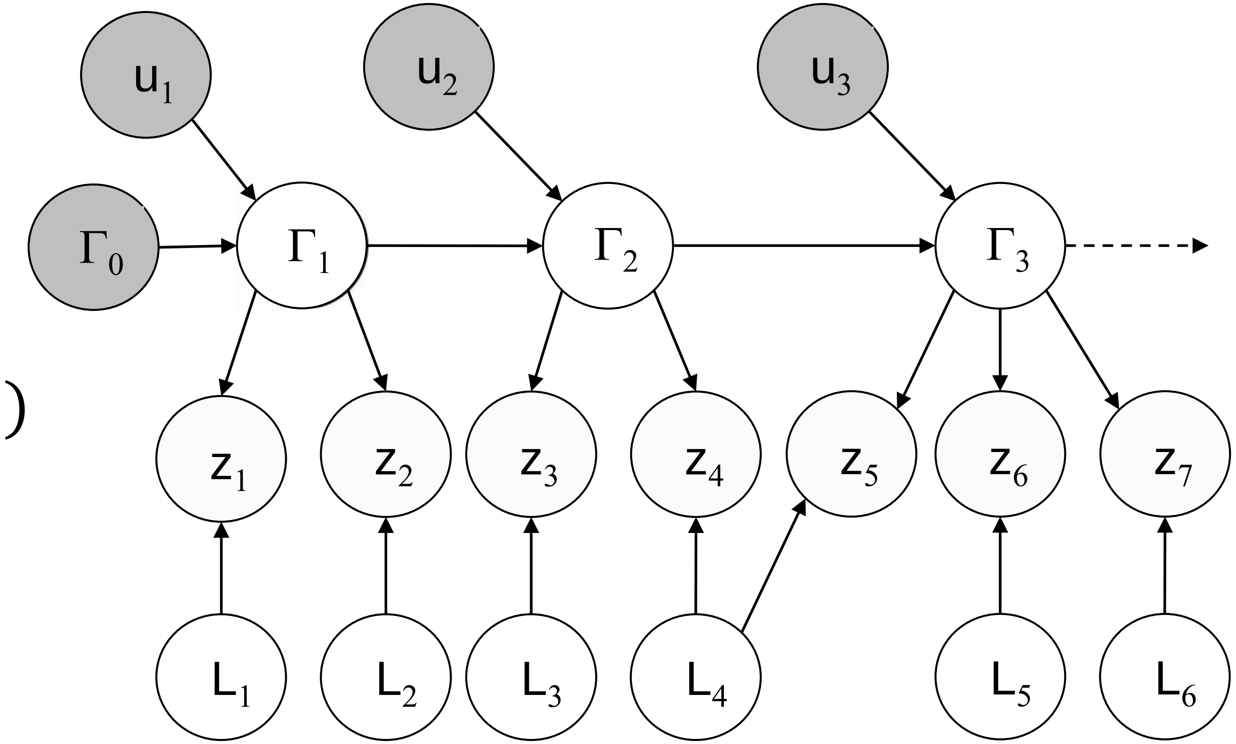
$$p(X,Z,U) = p(\Gamma_{0:3}, L_1, \ldots, L_6 | z_{1:7}, u_{1:7})$$

$$= p(\Gamma_1|\Gamma_0,u_1)p(\Gamma_2|\Gamma_1,u_1)p(\Gamma_3|\Gamma_2,u_3)$$
$$p(Z_1|\Gamma_1,L_1)p(L_1)p(Z_2|\Gamma_1,L_2)p(L_2)$$
$$\boxed{p(Z_3|\Gamma_2,L_3)p(L_3)}\boxed{p(Z_4|\Gamma_2,L_4)p(L_4)}$$
$$p(Z_5|\Gamma_3,L_4)p(Z_6|\Gamma_3,L_5)p(L_5)$$
$$p(Z_7|\Gamma_3,L_6)p(L_6)$$

$$= \phi(\Gamma_1,\Gamma_0,u_1)\phi(\Gamma_2,\Gamma_1,u_1)\phi(\Gamma_3,\Gamma_2,u_3)\phi(Z_1,\Gamma_1,L_1)\phi(Z_2,\Gamma_1,L_2)$$
$$\phi(Z_3,\Gamma_2,L_3)\phi(Z_4,\Gamma_2,L_4)\phi(Z_5,\Gamma_3,L_4)\phi(Z_6,\Gamma_3,L_5)\phi(Z_7,\Gamma_3,L_6)$$

Bayesian Networks and Joint Distribution

The full joint distribution of a Bayesian network is the product of the conditionals



$$p(X,Z,U) = p(\Gamma_{0:3}, L_1, \dots, L_6 | z_{1:7}, u_{1:7})$$

$$= p(\Gamma_1|\Gamma_0,u_1)p(\Gamma_2|\Gamma_1,u_1)p(\Gamma_3|\Gamma_2,u_3)$$
$$p(Z_1|\Gamma_1,L_1)p(L_1)p(Z_2|\Gamma_1,L_2)p(L_2)$$
$$p(Z_3|\Gamma_2,L_3)p(L_3)p(Z_4|\Gamma_2,L_4)p(L_4)$$
$$\boxed{p(Z_5|\Gamma_3,L_4)}\boxed{p(Z_6|\Gamma_3,L_5)p(L_5)}$$
$$\boxed{p(Z_7|\Gamma_3,L_6)p(L_6)}$$

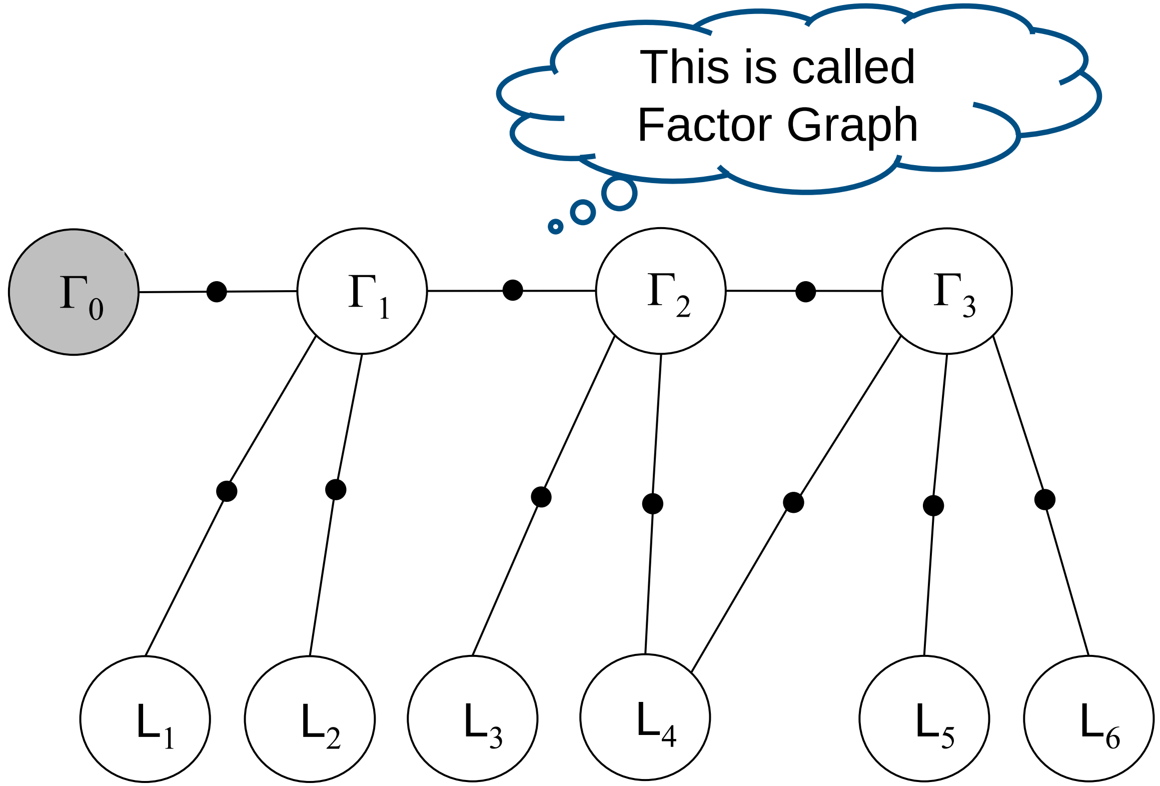
$$= \phi(\Gamma_1,\Gamma_0,u_1)\phi(\Gamma_2,\Gamma_1,u_1)\phi(\Gamma_3,\Gamma_2,u_3)\phi(Z_1,\Gamma_1,L_1)\phi(Z_2,\Gamma_1,L_2)$$
$$\phi(Z_3,\Gamma_2,L_3)\phi(Z_4,\Gamma_2,L_4)\phi(Z_5,\Gamma_3,L_4)\phi(Z_6,\Gamma_3,L_5)\phi(Z_7,\Gamma_3,L_6)$$

Bayesian Networks and Joint Distribution

The full joint distribution of a Bayesian network is the product of the conditionals

$$p(X,Z,U) = p(\Gamma_{0:3}, L_1, \dots, L_6 | z_{1:7}, u_{1:7})$$

$$= p(\Gamma_1|\Gamma_0,u_1)p(\Gamma_2|\Gamma_1,u_1)p(\Gamma_3|\Gamma_2,u_3) \\ p(Z_1|\Gamma_1,L_1)p(L_1)p(Z_2|\Gamma_1,L_2)p(L_2) \\ p(Z_3|\Gamma_2,L_3)p(L_3)p(Z_4|\Gamma_2,L_4)p(L_4) \\ p(Z_5|\Gamma_3,L_4)p(Z_6|\Gamma_3,L_5)p(L_5) \\ p(Z_7|\Gamma_3,L_6)p(L_6)$$



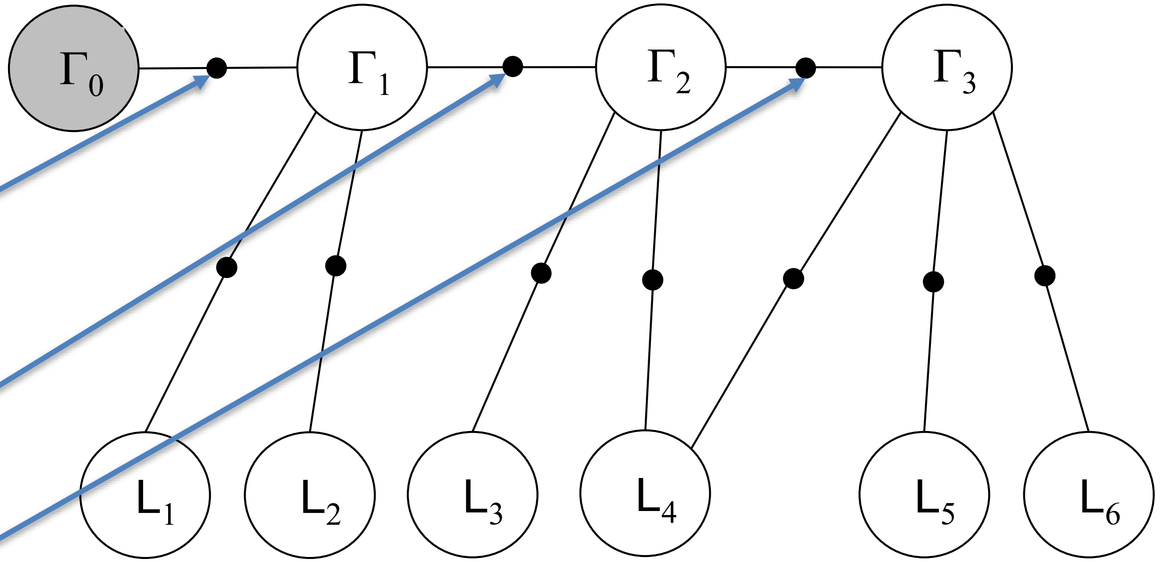
$$= \phi(\Gamma_1,\Gamma_0,u_1)\phi(\Gamma_2,\Gamma_1,u_1)\phi(\Gamma_3,\Gamma_2,u_3)\phi(Z_1,\Gamma_1,L_1)\phi(Z_2,\Gamma_1,L_2) \\ \phi(Z_3,\Gamma_2,L_3)\phi(Z_4,\Gamma_2,L_4)\phi(Z_5,\Gamma_3,L_4)\phi(Z_6,\Gamma_3,L_5)\phi(Z_7,\Gamma_3,L_6)$$

Bayesian Networks and Joint Distribution

The full joint distribution of a Bayesian network is the product of the conditionals

$$p(X,Z,U) = p(\Gamma_{0:3}, L_1, \dots, L_6 | z_{1:7}, u_{1:7})$$

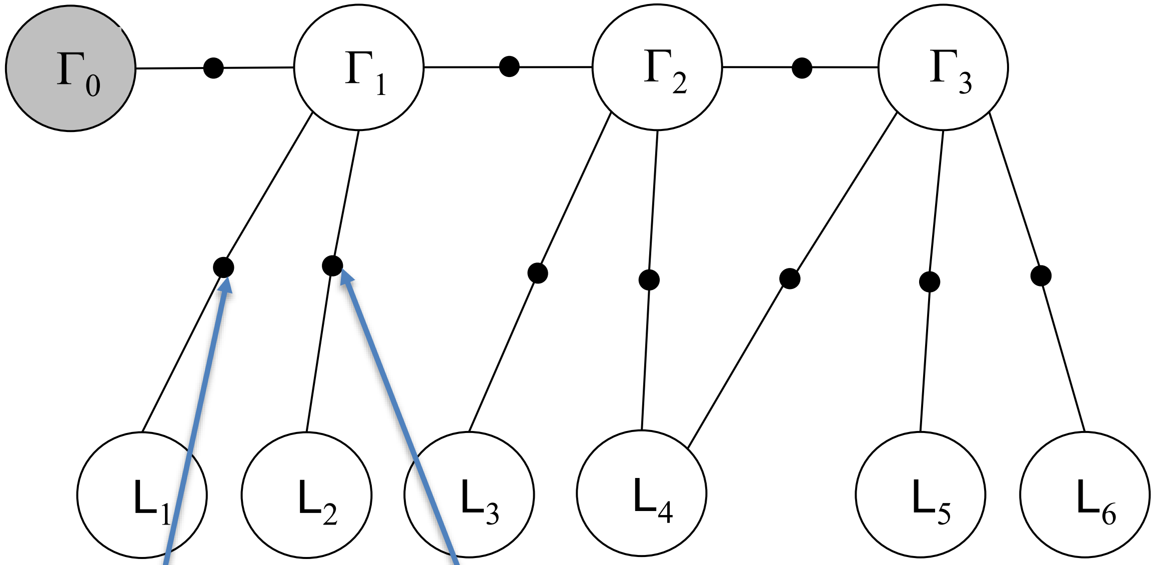
$$= p(\Gamma_1|\Gamma_0,u_1)p(\Gamma_2|\Gamma_1,u_1)p(\Gamma_3|\Gamma_2,u_3) \\ p(Z_1|\Gamma_1,L_1)p(L_1)p(Z_2|\Gamma_1,L_2)p(L_2) \\ p(Z_3|\Gamma_2,L_3)p(L_3)p(Z_4|\Gamma_2,L_4)p(L_4) \\ p(Z_5|\Gamma_3,L_4)p(Z_6|\Gamma_3,L_5)p(L_5) \\ p(Z_7|\Gamma_3,L_6)p(L_6)$$



$$= \boxed{\phi(\Gamma_1,\Gamma_0,u_1)}\boxed{\phi(\Gamma_2,\Gamma_1,u_1)}\boxed{\phi(\Gamma_3,\Gamma_2,u_3)}\phi(Z_1,\Gamma_1,L_1)\phi(Z_2,\Gamma_1,L_2) \\ \phi(Z_3,\Gamma_2,L_3)\phi(Z_4,\Gamma_2,L_4)\phi(Z_5,\Gamma_3,L_4)\phi(Z_6,\Gamma_3,L_5)\phi(Z_7,\Gamma_3,L_6)$$

Bayesian Networks and Joint Distribution

The full joint distribution of a Bayesian network is the product of the conditionals



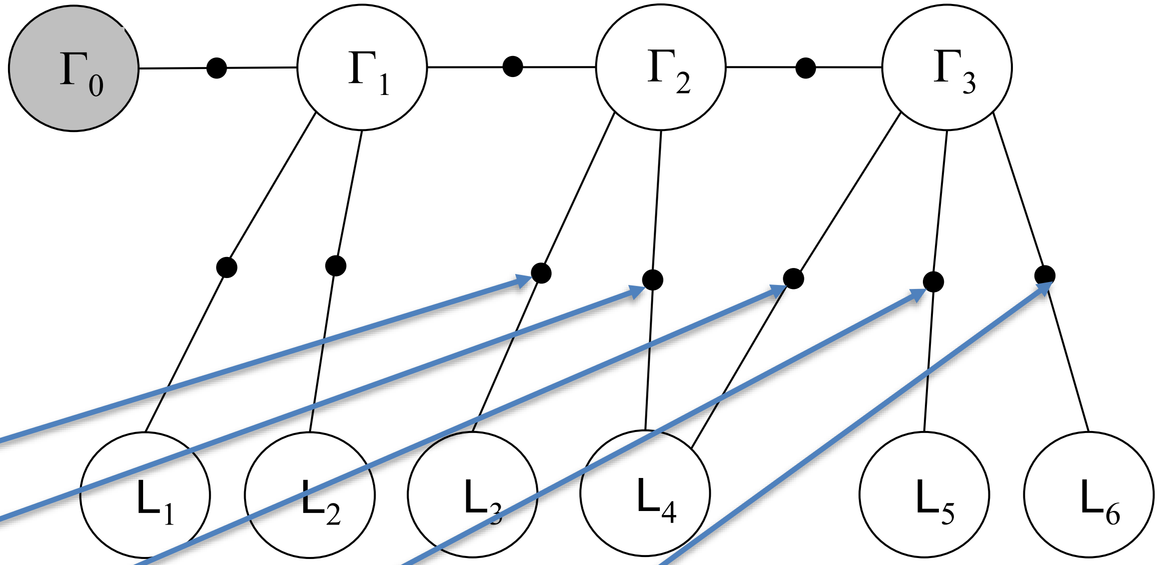
$$p(X, Z, U) = p(\Gamma_{0:3}, L_1, \dots, L_6 | z_{1:7}, u_{1:7})$$

$$= p(\Gamma_1|\Gamma_0, u_1)p(\Gamma_2|\Gamma_1, u_1)p(\Gamma_3|\Gamma_2, u_3) \\ p(Z_1|\Gamma_1, L_1)p(L_1)p(Z_2|\Gamma_1, L_2)p(L_2) \\ p(Z_3|\Gamma_2, L_3)p(L_3)p(Z_4|\Gamma_2, L_4)p(L_4) \\ p(Z_5|\Gamma_3, L_4)p(Z_6|\Gamma_3, L_5)p(L_5) \\ p(Z_7|\Gamma_3, L_6)p(L_6)$$

$$= \phi(\Gamma_1, \Gamma_0, u_1)\phi(\Gamma_2, \Gamma_1, u_1)\phi(\Gamma_3, \Gamma_2, u_3)\boxed{\phi(Z_1, \Gamma_1, L_1)}\boxed{\phi(Z_2, \Gamma_1, L_2)} \\ \phi(Z_3, \Gamma_2, L_3)\phi(Z_4, \Gamma_2, L_4)\phi(Z_5, \Gamma_3, L_4)\phi(Z_6, \Gamma_3, L_5)\phi(Z_7, \Gamma_3, L_6)$$

Bayesian Networks and Joint Distribution

The full joint distribution of a Bayesian network is the product of the conditionals

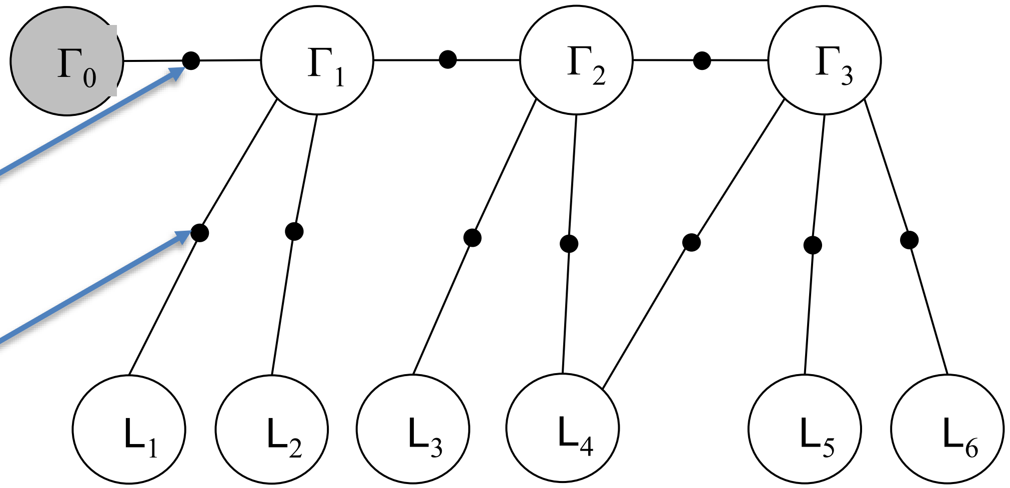


$$p(X,Z,U) = p(\Gamma_{0:3}, L_1, \ldots, L_6 | z_{1:7}, u_{1:7})$$

$$= p(\Gamma_1|\Gamma_0,u_1)p(\Gamma_2|\Gamma_1,u_1)p(\Gamma_3|\Gamma_2,u_3)$$
$$p(Z_1|\Gamma_1,L_1)p(L_1)p(Z_2|\Gamma_1,L_2)p(L_2)$$
$$p(Z_3|\Gamma_2,L_3)p(L_3)p(Z_4|\Gamma_2,L_4)p(L_4)$$
$$p(Z_5|\Gamma_3,L_4)p(Z_6|\Gamma_3,L_5)p(L_5)$$
$$p(Z_7|\Gamma_3,L_6)p(L_6)$$

$$= \phi(\Gamma_1,\Gamma_0,u_1)\phi(\Gamma_2,\Gamma_1,u_1)\phi(\Gamma_3,\Gamma_2,u_3)\phi(Z_1,\Gamma_1,L_1)\phi(Z_2,\Gamma_1,L_2)$$
$$\boxed{\phi(Z_3,\Gamma_2,L_3)}\boxed{\phi(Z_4,\Gamma_2,L_4)}\boxed{\phi(Z_5,\Gamma_3,L_4)}\boxed{\phi(Z_6,\Gamma_3,L_5)}\boxed{\phi(Z_7,\Gamma_3,L_6)}$$

Full SLAM as Graph Optimization



Given the Factor Graph Full Joint Distribution

$$p(X, Z, U) = \prod_i \phi_i(X_i)$$

The Full SLAM problem is reformulated as

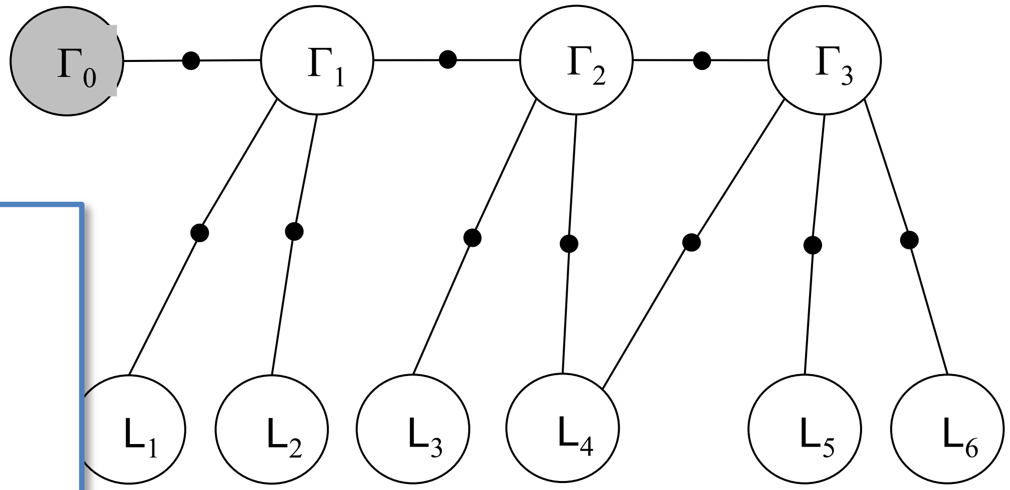
$$X^{MAP} = \underset{X}{argmax}\ p(X|Z, U) = \underset{X}{argmax}\ p(X, Z, U) = \underset{X}{argmax} \prod_i \phi_i(X_i)$$

Let's also assume to have Gaussian Factors (not mandatory but convenient)

$$\boxed{\phi(\Gamma_1, \Gamma_0, u_1)} = P(\Gamma_1|\Gamma_0, u_1) = N(g(\Gamma_0, u_1), R) = \frac{1}{\sqrt{|2\pi R|}} \cdot \exp\left(-\frac{1}{2}\|g(\Gamma_0, u_1) - \Gamma_1\|_R^2\right)$$

$$\boxed{\phi(Z_1, \Gamma_1, L_1)} = P(Z_1|\Gamma_1, L_1) = N(h(\Gamma_1, L_1), Q) = \frac{1}{\sqrt{|2\pi Q|}} \cdot \exp\left(-\frac{1}{2}\|h(\Gamma_1, L_1) - Z_1\|_Q^2\right)$$

Full SLAM as Graph Optimization



Given the Factor Graph Full Joint Distribution

$$x \sim N(\mu, \Sigma) = \frac{1}{\sqrt{|2\pi\Sigma|}} \cdot \exp\left(-\frac{1}{2}\|\mu - x\|_{\Sigma}^{2}\right)$$

$$\|\mu - x\|_{\Sigma}^{2} \equiv (\mu - x)^{T}\Sigma^{-1}(\mu - x)$$

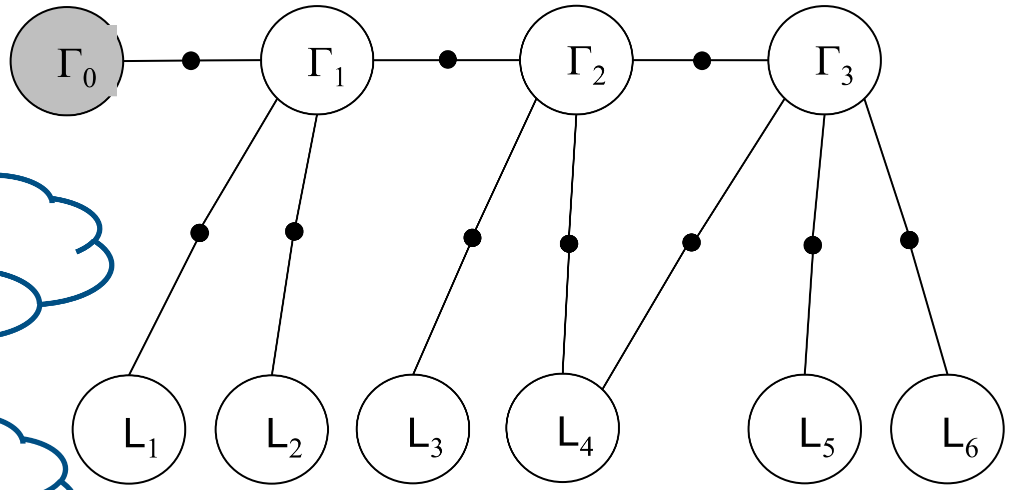
$$Z, U) = \underset{X}{argmax} \prod_{i} \phi_i(X_i)$$

Let's also assume to have Gaussian Factors (not mandatory but convenient)

$$\phi(\Gamma_1, \Gamma_0, u_1) = P(\Gamma_1|\Gamma_0, u_1) = N(g(\Gamma_0, u_1), R) = \boxed{\frac{1}{\sqrt{|2\pi R|}} \cdot \exp\left(-\frac{1}{2}\|g(\Gamma_0, u_1) - \Gamma_1\|_{R}^{2}\right)}$$

$$\phi(Z_1, \Gamma_1, L_1) = P(Z_1|\Gamma_1, L_1) = N(h(\Gamma_1, L_1), Q) = \frac{1}{\sqrt{|2\pi Q|}} \cdot \exp\left(-\frac{1}{2}\|h(\Gamma_1, L_1) - Z_1\|_{Q}^{2}\right)$$

Graph Optimization on Factor Graphs



The (Gaussian) Full SLAM problem becom

$$\phi_{i=u_i}(X_i) \propto \exp\left(-\frac{1}{2}\|g_i(X_i) - \Gamma_i\|_R^2\right)$$

Odometry factors

$$\phi_{i=z_i}(X_i) \propto \exp\left(-\frac{1}{2}\|h_i(X_i) - z_i\|_Q^2\right)$$

Measurement factors

$$X^{MAP} = \underset{X}{argmax} \prod_i \phi_i(X_i) = \underset{X}{argmax} \prod_{i=u_i} \exp\left(-\frac{1}{2}\|g_i(X_i) - \Gamma_i\|_R^2\right) \prod_{i=z_i} \exp\left(-\frac{1}{2}\|h_i(X_i) - z_i\|_Q^2\right)$$

If we solve for the logarithm we get a simpler optimization algorithm

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

$$= \underset{X}{argmax} \sum_{i=u_i} \log\exp\left(-\frac{1}{2}\|g_i(X_i) - \Gamma_i\|_R^2\right) + \sum_{i=z_i} \log\exp\left(-\frac{1}{2}\|h_i(X_i) - z_i\|_Q^2\right)$$

Graph Optimization on Factor Graphs

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

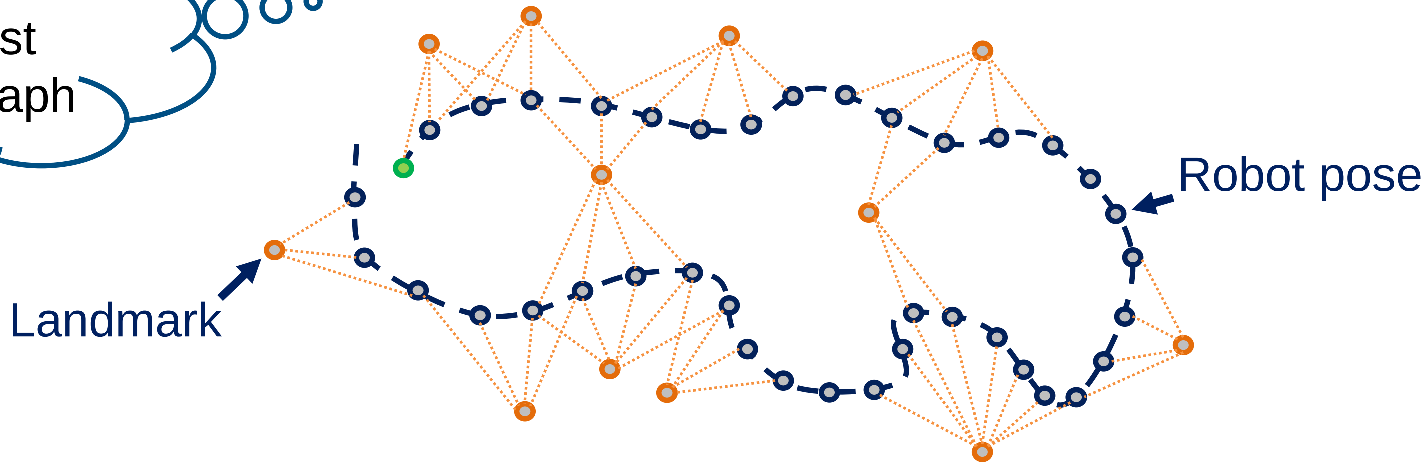
$$= \underset{X}{argmax} \sum_{i=u_i} \log \exp\left(-\frac{1}{2}\|g_i(X_i) - \Gamma_i\|_R^2\right) + \sum_{i=z_i} \log \exp\left(-\frac{1}{2}\|h_i(X_i) - \mathrm{z}_i\|_Q^2\right)$$

$$= \underset{X}{argmax} \sum_{i=u_i} -\frac{1}{2}\|g_i(X_i) - \Gamma_i\|_R^2 + \sum_{i=z_i} -\frac{1}{2}\|h_i(X_i) - \mathrm{z}_i\|_Q^2$$

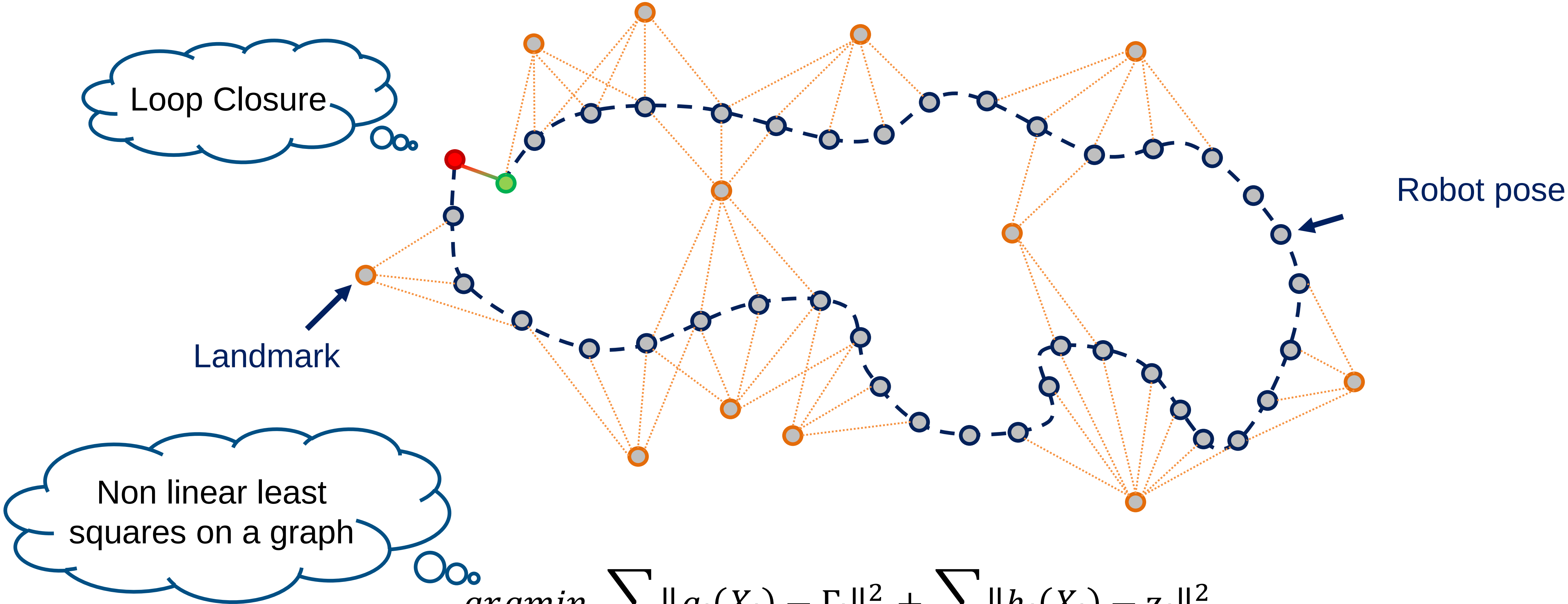
Multiply by -2 then max -> min

$$= \underset{X}{argmin} \sum_{i=u_i} \|g_i(X_i) - \Gamma_i\|_R^2 + \sum_{i=z_i} \|h_i(X_i) - \mathrm{z}_i\|_Q^2$$

Non linear least squares on a graph

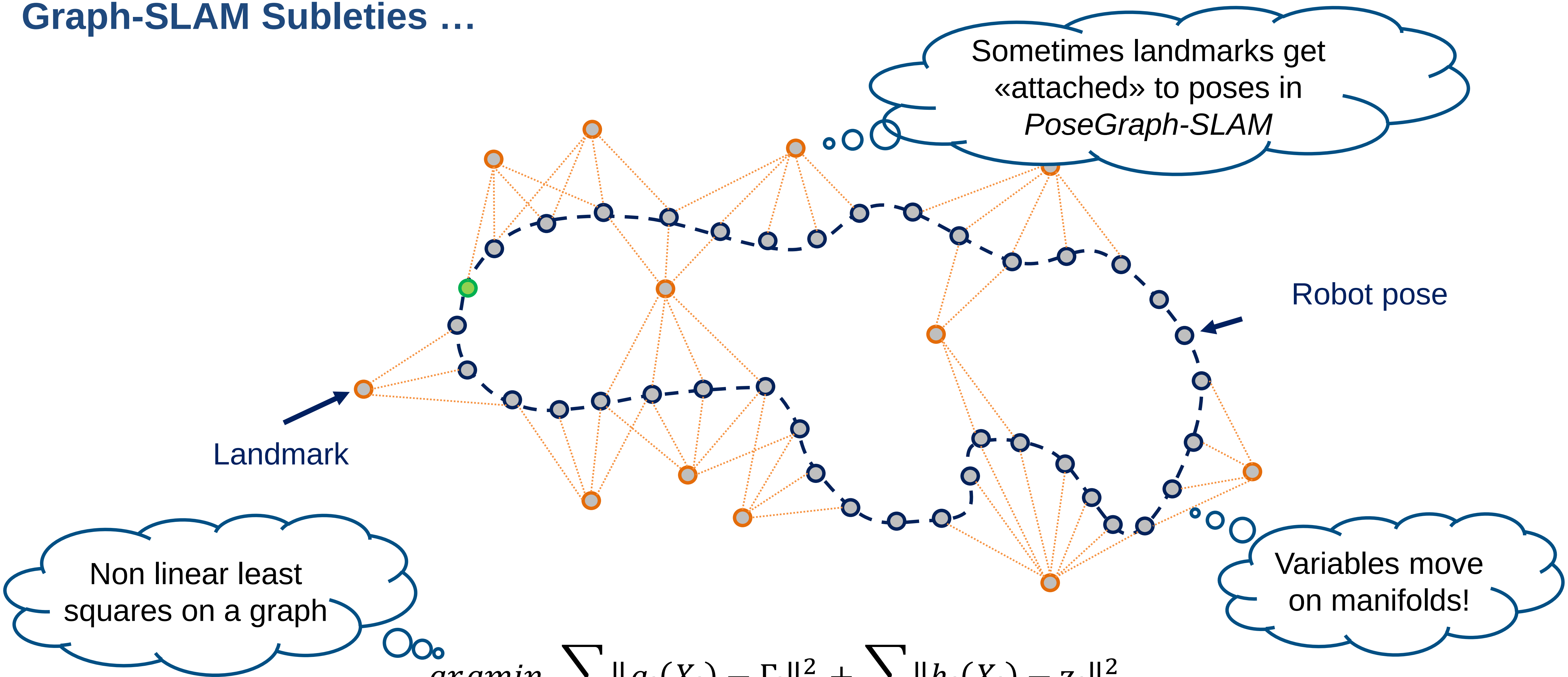


Graph-SLAM



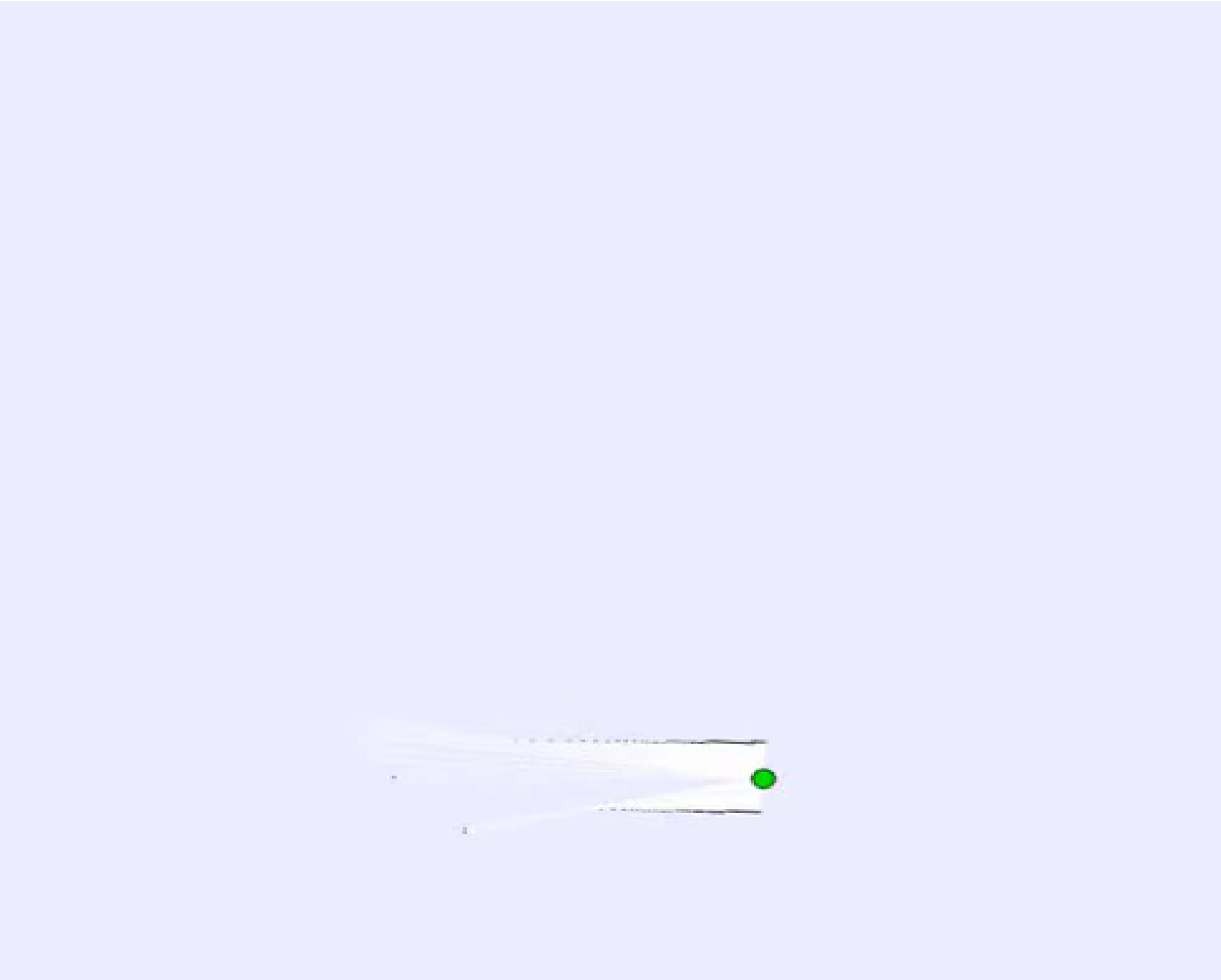
$$\underset{X}{argmin} \sum_{i=u_i} \|g_i(X_i) - \Gamma_i\|_R^2 + \sum_{i=z_i} \|h_i(X_i) - z_i\|_Q^2$$

Graph-SLAM Subleties …



$$\underset{X}{argmin} \sum_{i=u_i} \|g_i(X_i) - \Gamma_i\|_R^2 + \sum_{i=z_i} \|h_i(X_i) - z_i\|_Q^2$$

Graph-SLAM Example



What did happen to the quadratic complexity?

Solving non-linear least squares needs iterative adjustments (gradient descend)

$$\underset{X}{argmin} \sum_{i=u_i} \|g_i(X_i) - \Gamma_i\|_R^2 + \sum_{i=z_i} \|h_i(X_i) - \mathrm{z}_i\|_Q^2$$

Let's focus on measurement factors, then the following extends to all factors

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

This is the usual Taylor expansion

$$\Delta_i \equiv X_i - X_i^0, \quad H_i \equiv \frac{\partial h_i(X_i)}{\partial X_i}\Big|_{X_i^0}$$

We look for the single adjustment step which minimizes all measurement factors

$$\Delta^* = \underset{\Delta}{argmin} \sum_{i=z_i} \|h_i(X_i) - \mathrm{z}_i\|_Q^2 = \underset{\Delta}{argmin} \sum_{i=z_i} \left\| H_i \Delta_i - \left(z_i - h_i\left(X_i^0\right)\right) \right\|_Q^2$$

What did happen to the quadratic complexity?

$$\Delta^* = \underset{\Delta}{argmin} \sum_{i=z_i} \|h_i(X_i) - z_i\|_Q^2 = \underset{\Delta}{argmin} \sum_{i=z_i} \Big\| \underbrace{H_i\Delta_i - \left(z_i - h_i\left(X_i^0\right)\right)}_{e_i} \Big\|_Q^2$$

We can rewrite the Mahalanobis norm as it follows turning it into quadratic

$$\|e_i\|_Q^2 \equiv e_i^T Q^{-1} e_i = \left(Q^{-1/2} e_i\right)^T \left(Q^{-1/2} e_i\right) = \left\|Q^{-1/2} e\right\|_2^2$$

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

From this we can get to

$$\Delta^* = \underset{\Delta}{argmin} \sum_i \|A_i\Delta_i - b_i\|_2^2 = \underset{\Delta}{argmin} \|A\Delta - b\|_2^2$$

$$A_i = Q_i^{-1/2} H_i, \quad b_i = Q_i^{-1}\left(z_i - h_i\left(X_i^0\right)\right)$$

Linear least squares problem

Lets' assume Odometry is included too from now on ...

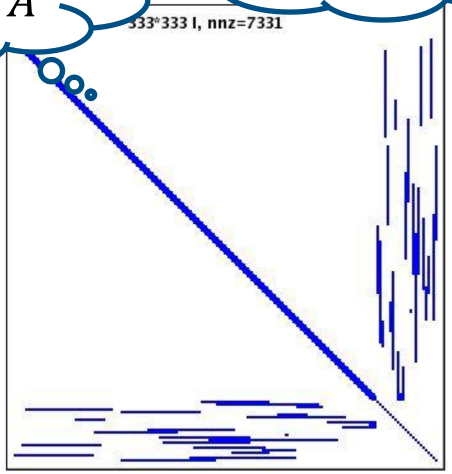
What did happen to the quadratic complexity?

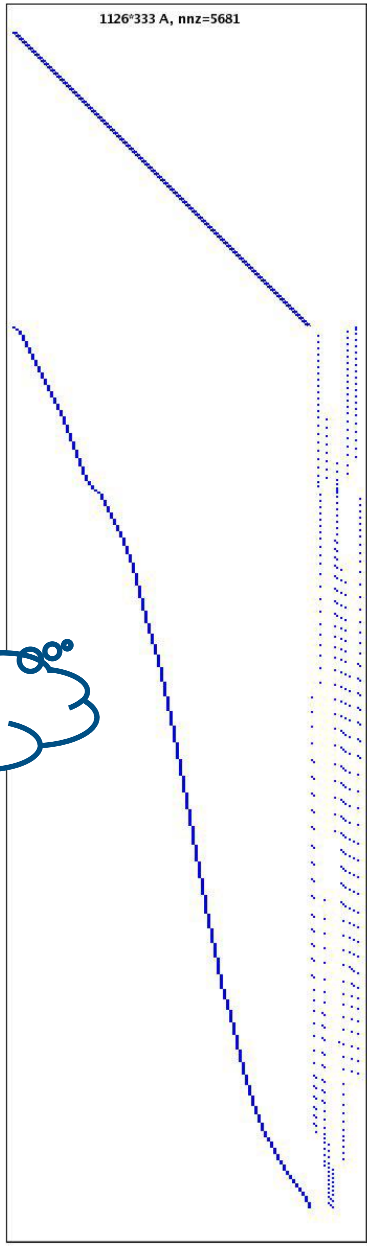
$$\Delta^* = \underset{\Delta}{argmin} \sum_i \|A_i \Delta_i - b_i\|_2^2 = \underset{\Delta}{argmin} \|A\Delta - b\|_2^2$$

Let's solve the least squares problem

$$\|A\Delta - b\|_2^2 = (A\Delta - b)^T(A\Delta - b) = \Delta^{\mathrm{T}} A^{\mathrm{T}} A \Delta - 2\Delta^{\mathrm{T}} A^T b + bb^T$$

$$\frac{\partial \|A\Delta - b\|_2^2}{\partial \Delta} = 0 \quad \Rightarrow \quad A^T A \Delta = A^T b$$





Matrix A from Odometry and Measurement Jacobians

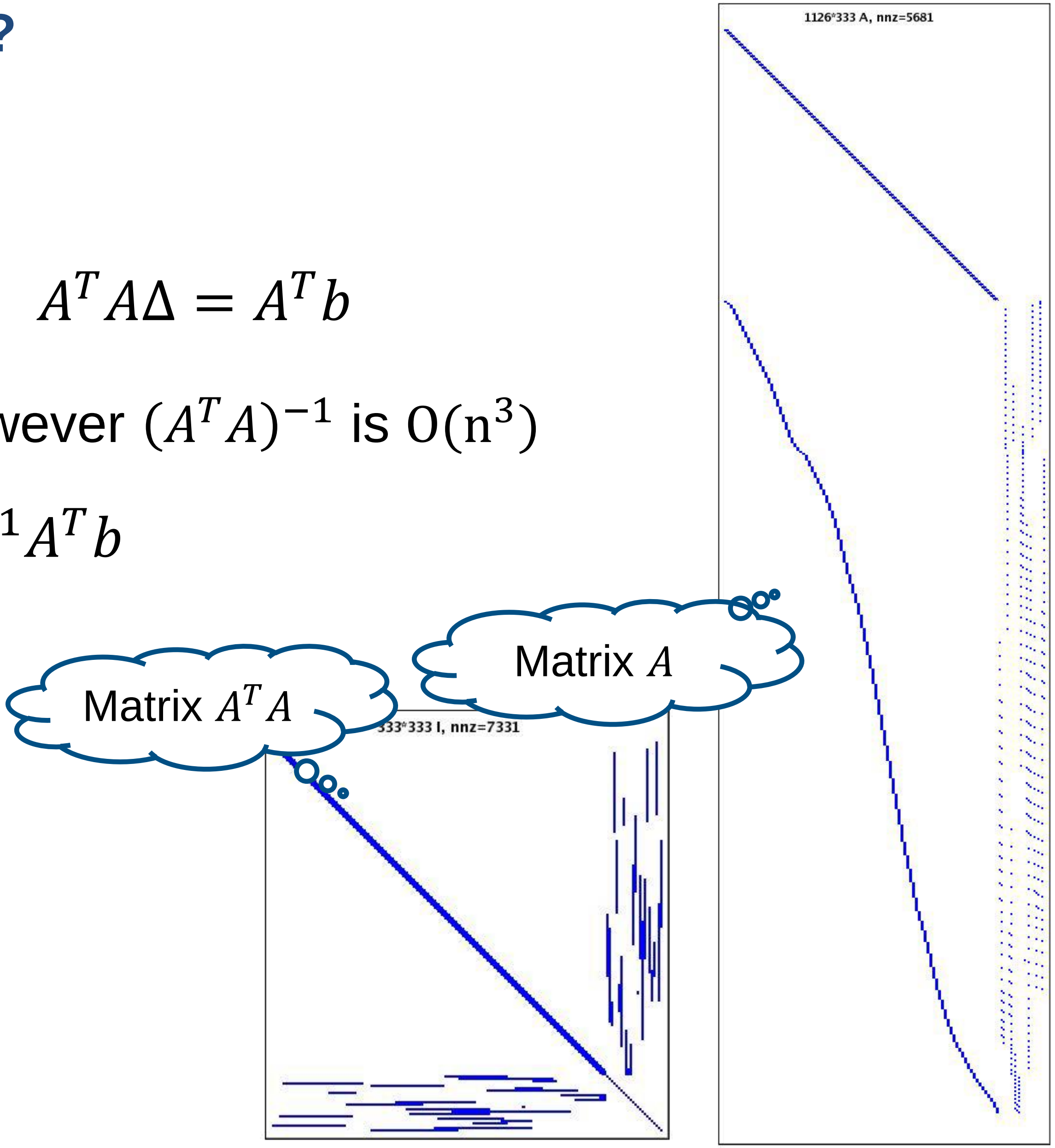
- Factors are constraints between 2 variables
- Matrix A is sparse and matrix $A^T A$ too
- We can use sparse methods which are fast !!!

What did happen to the quadratic complexity?

$$\frac{\partial \|A\Delta - b\|_2^2}{\partial \Delta} = 0 \quad \Rightarrow \quad A^T A \Delta = A^T b$$

Naïve least squares uses pseudo inverse, however $(A^T A)^{-1}$ is $O(n^3)$

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



What did happen to the quadratic complexity?

$$\frac{\partial \|A\Delta - b\|_2^2}{\partial \Delta} = 0 \quad \Rightarrow \quad A^T A\Delta = A^T b$$

Naïve least squares uses pseudo inverse, however $(A^T A)^{-1}$ is $\mathrm{O}(\mathrm{n}^3)$

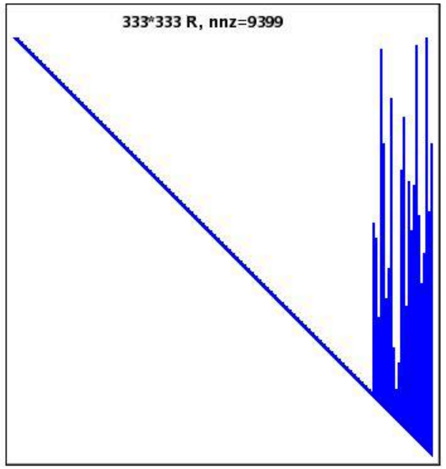
$$\Delta = (A^T A)^{-1} A^T b$$

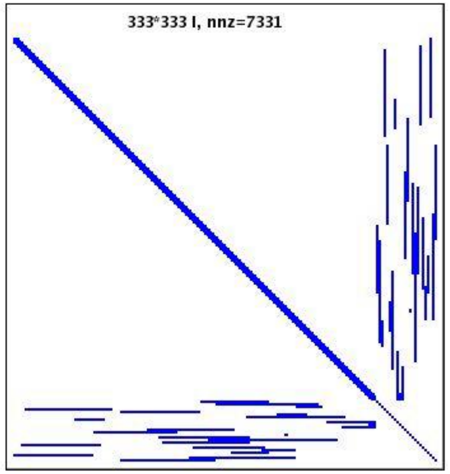
Cholesky decomposition $A^T A = R^T R$ (R upper triangular) is $\mathrm{O}(\mathrm{n}^{1.5})$ to $\mathrm{O}(\mathrm{n}^2)$

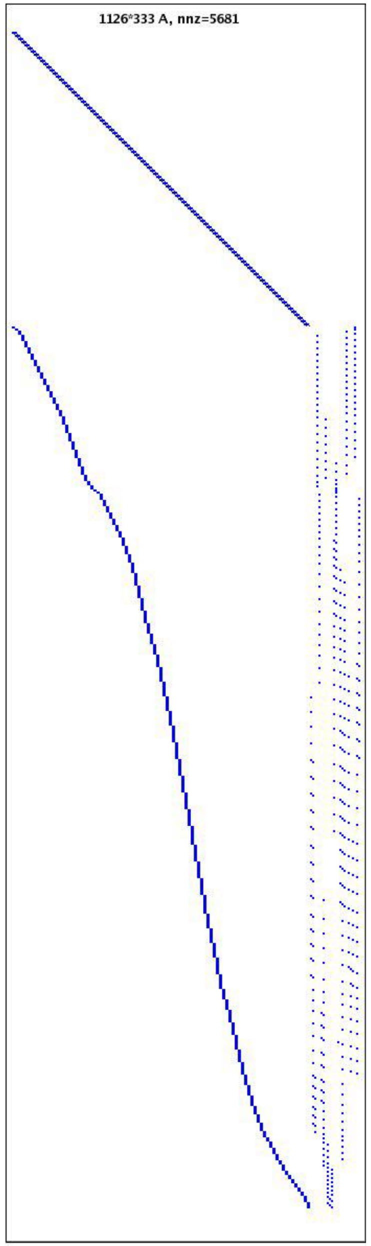
$$R^T R\Delta = A^T b$$

$$R^T y = A^T b$$

$$R^T \Delta = y$$







What did happen to the quadratic complexity?

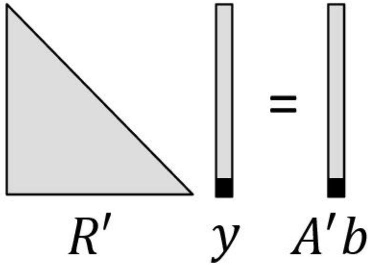
$$\frac{\partial \|A\Delta - b\|_2^2}{\partial \Delta} = 0 \quad \Rightarrow \quad A^T A \Delta = A^T b$$

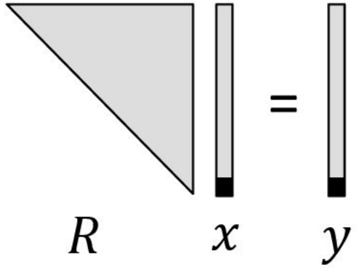
Naïve least squares uses pseudo inverse, however $(A^T A)^{-1}$ is $O(n^3)$

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

Cholesky decomposition $A^T A = R^T R$ (R upper triangular) is $O(n^{1.5})$ to $O(n^2)$

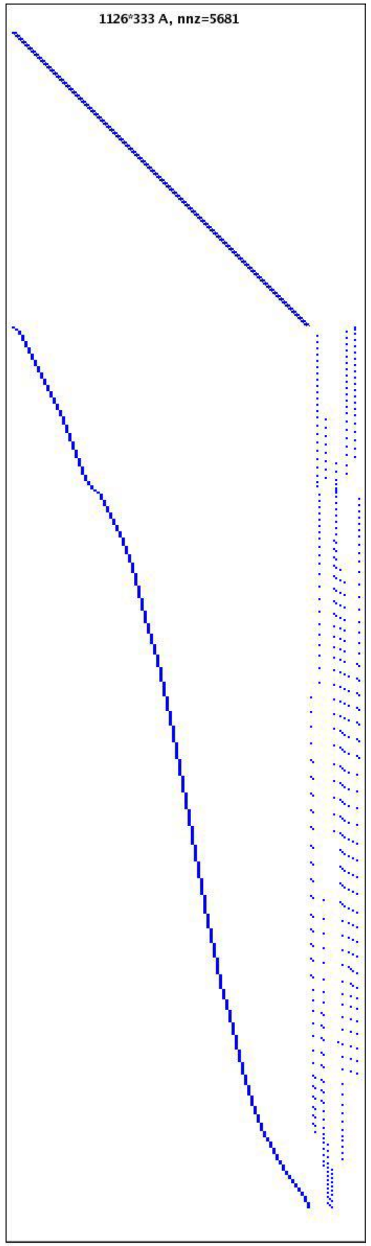
$$R^T R \Delta = A^T b$$
$$R^T y = A^T b$$
$$R^T \Delta = y$$



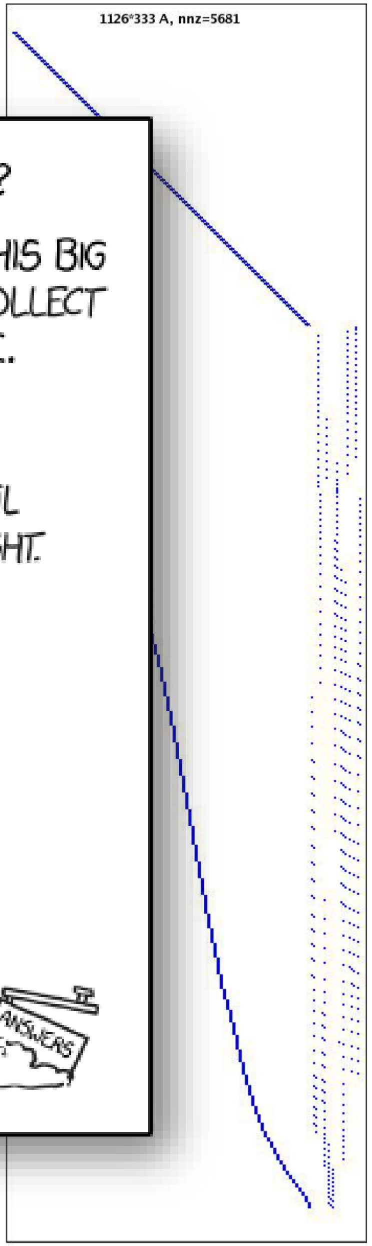


Solve by forward / backward substitution …

… and via LDL^T decomposition is even faster !!!



What did happen to the quadratic complexity?



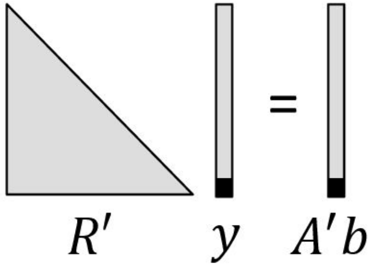
$$\frac{\partial \|A\Delta - b\|_2^2}{\partial \Delta} = 0 \Rightarrow A$$

Naïve least squares uses pseudo inverse, howe

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

Cholesky decomposition $A^T A = R^T R$ (R upper tri

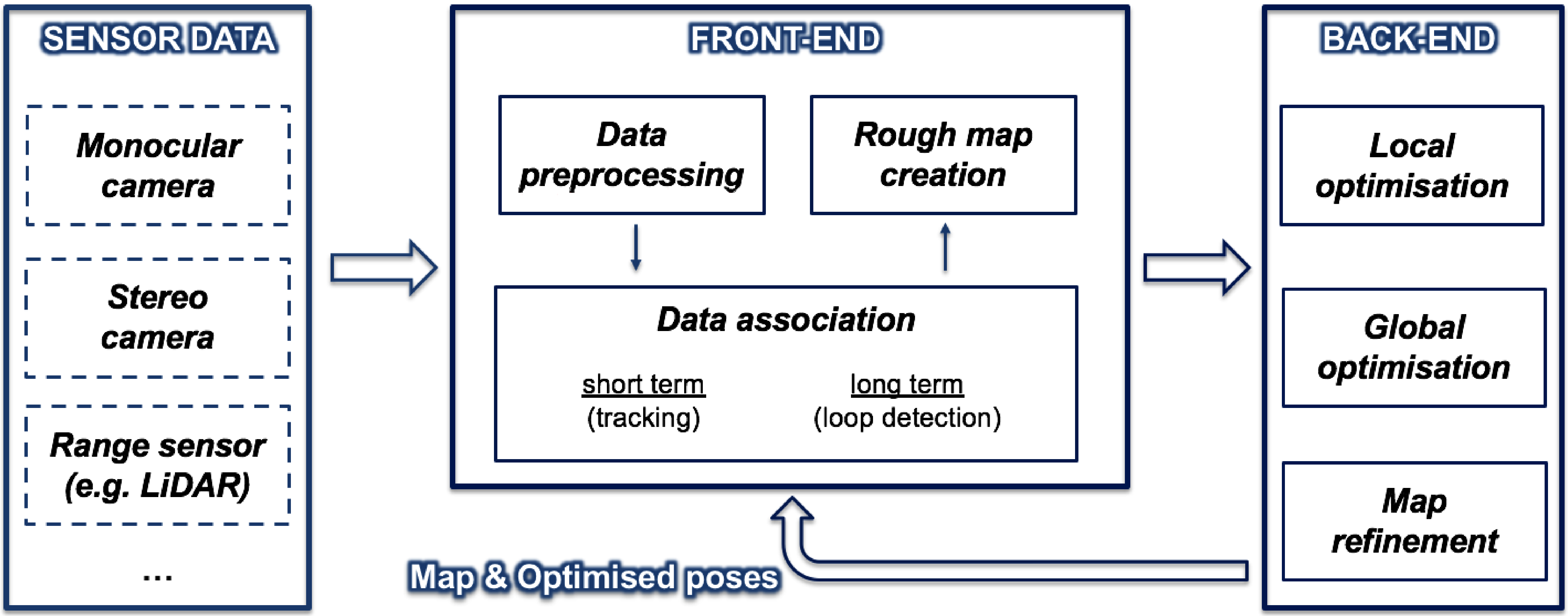
$$R^T R \Delta = A^T b$$
$$R^T y = A^T b$$
$$R^T \Delta = y$$



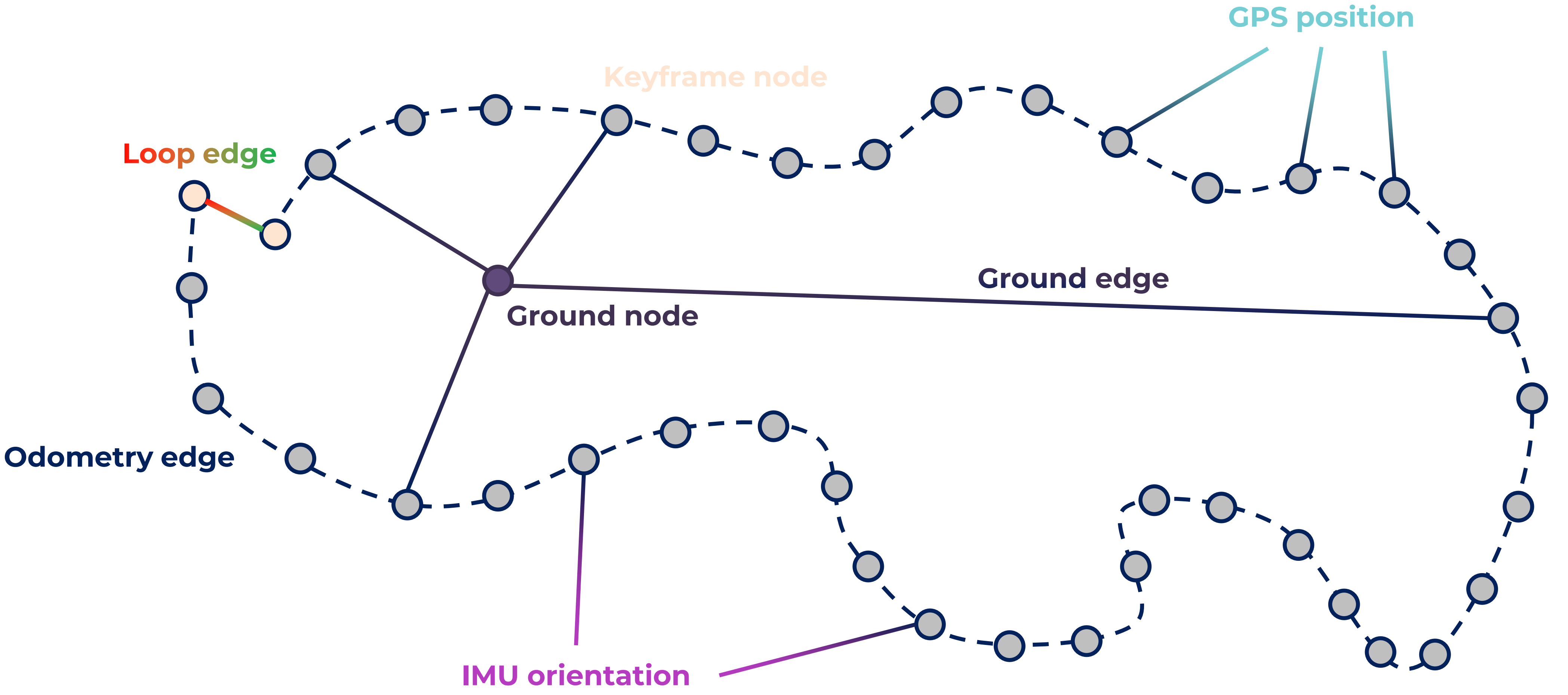
Solve by forward / backward substitution …

… and via LDL^T decomposition is even faster !!!

General Architecture of a Modern SLAM System



Pose-Graph SLAM



Pose-Graph SLAM