Robotics

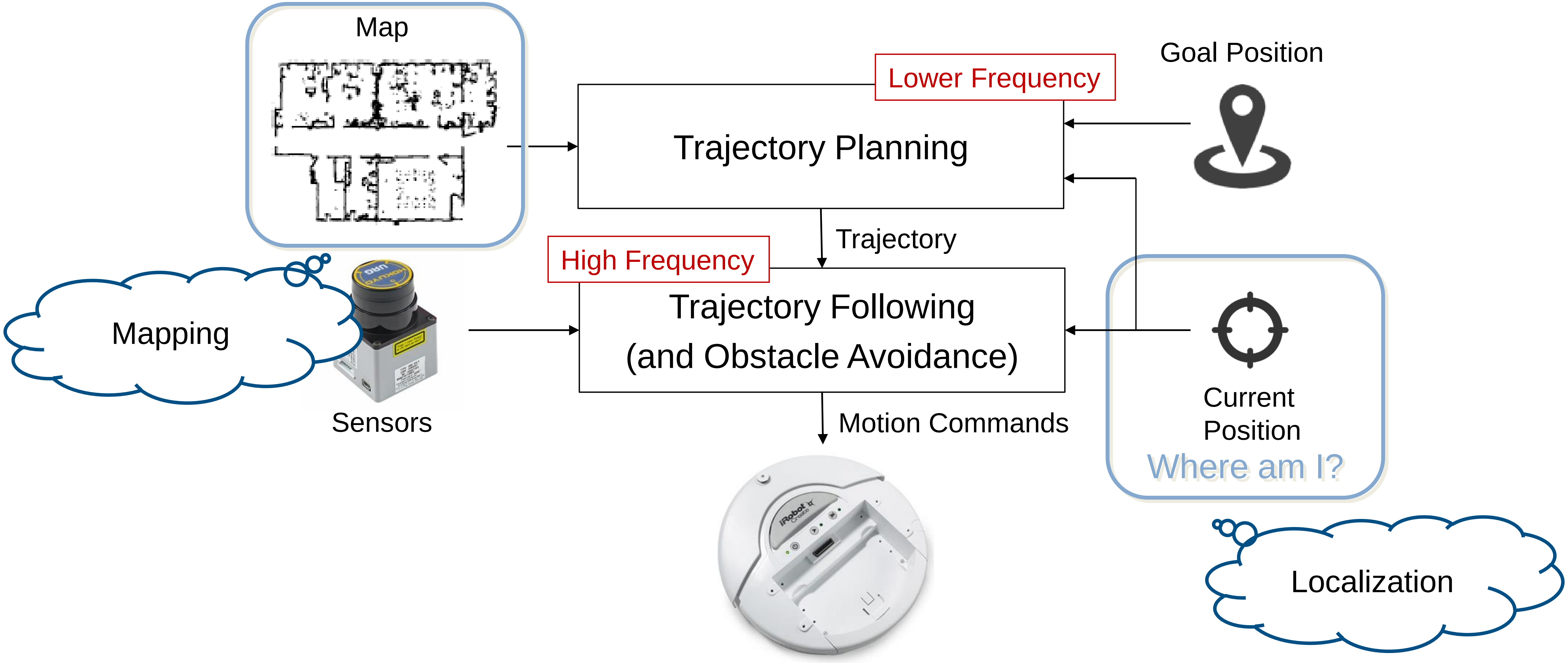
Simultaneous Localization and Mapping

Matteo Matteucci

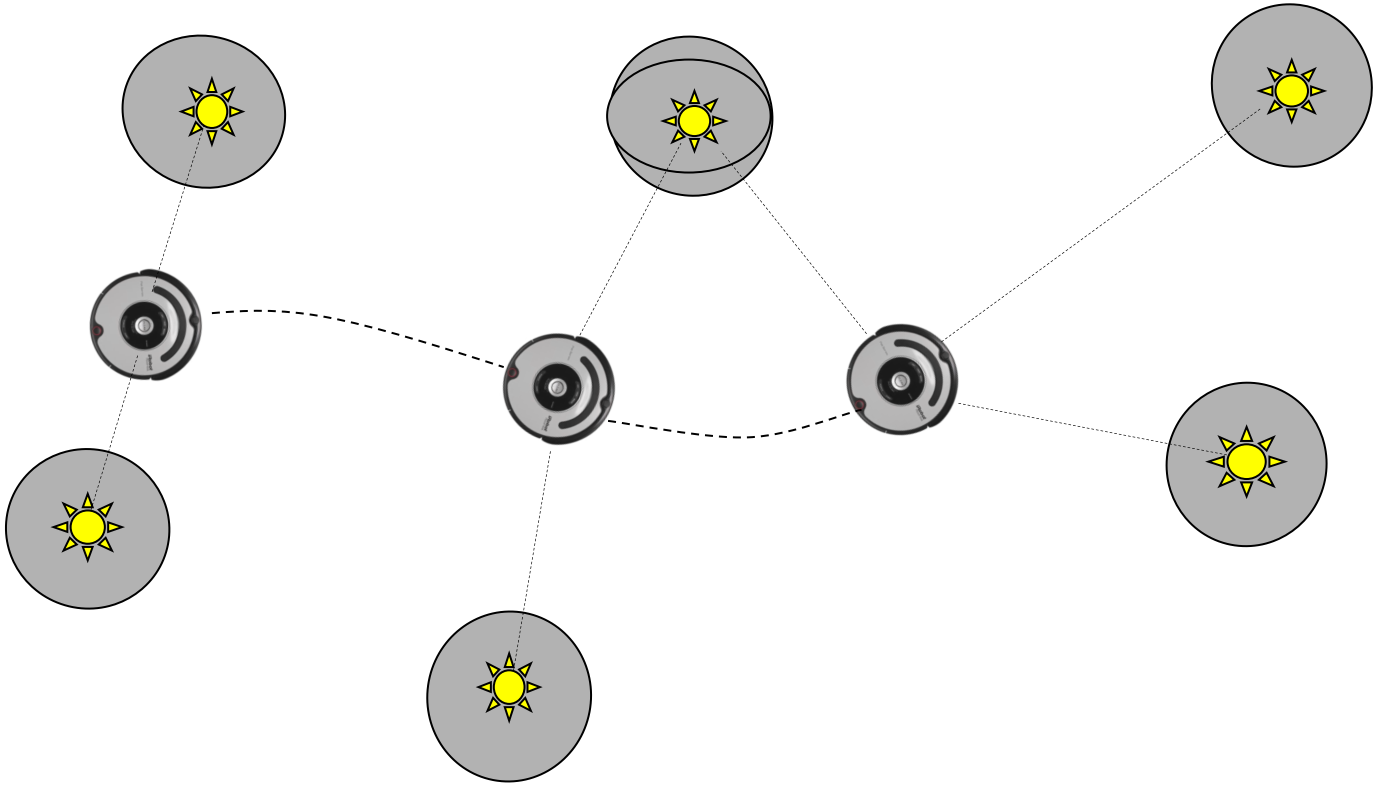
matteo.matteucci@polimi.it

Artificial Intelligence and Robotics Lab - Politecnico di Milano

A Simplified Sense-Plan-Act Architecture

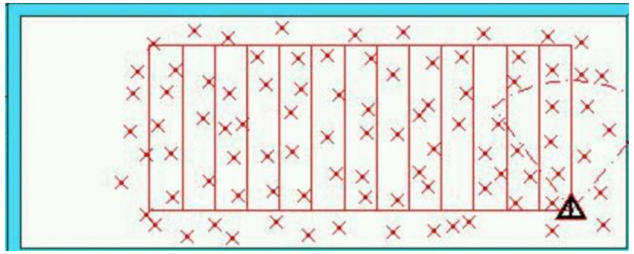
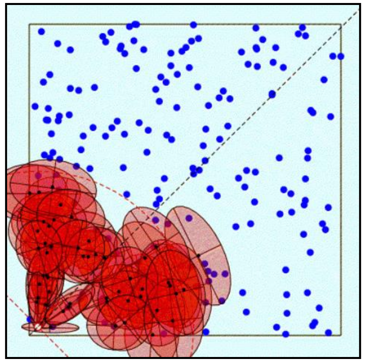
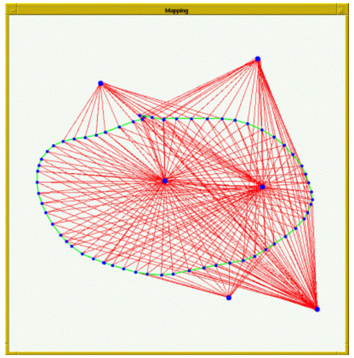


Mapping with Knwon Poses



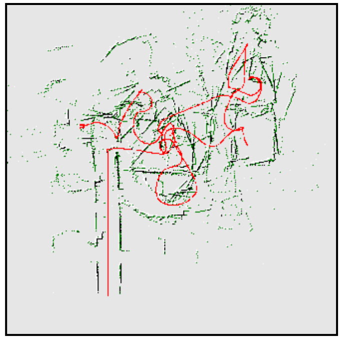
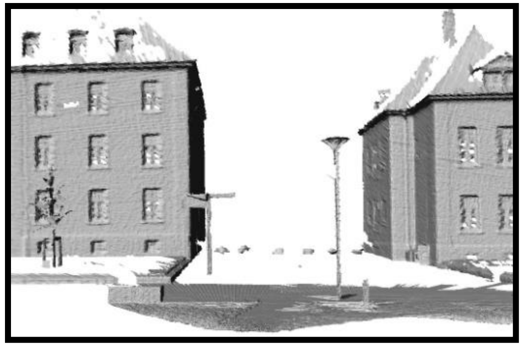
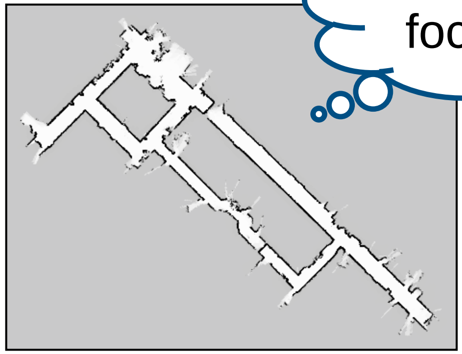
Representations

Landmark-based

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

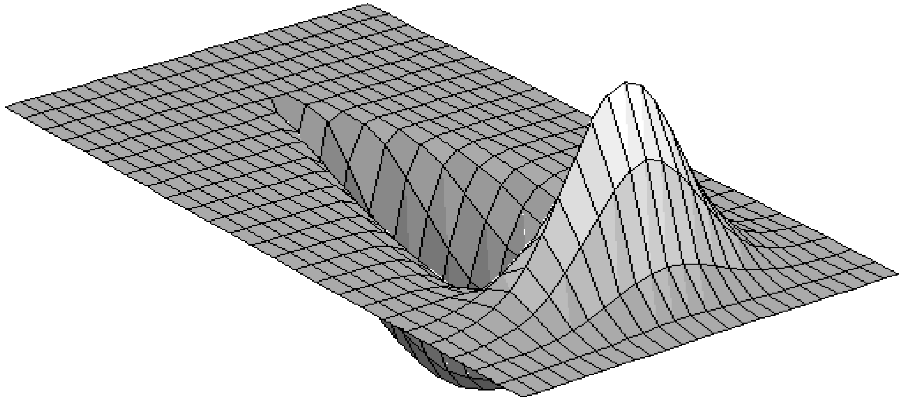
Grid maps or scans

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)

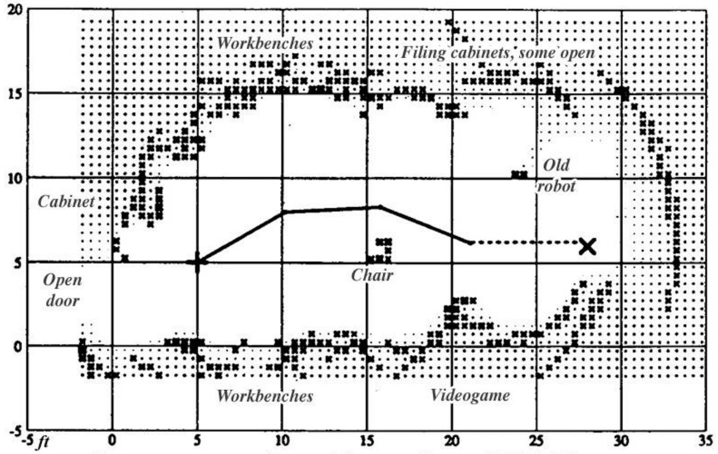


The most simple occupancy model used sonars

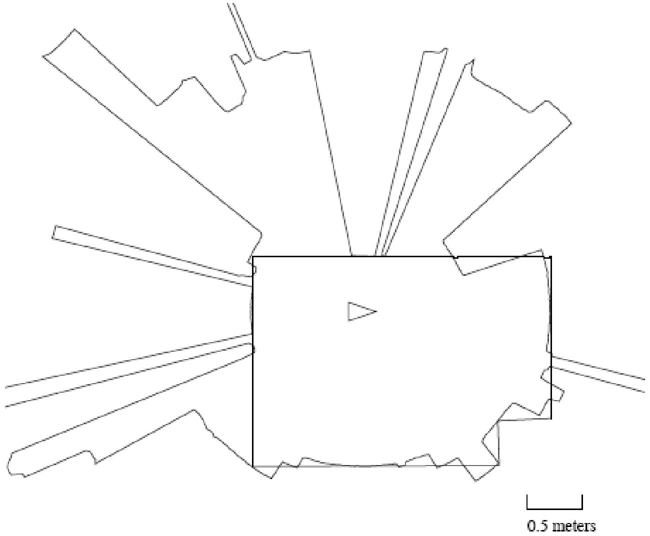
- A 2D Gaussian for information about occupancy
- Another 2D Gaussian for free space

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

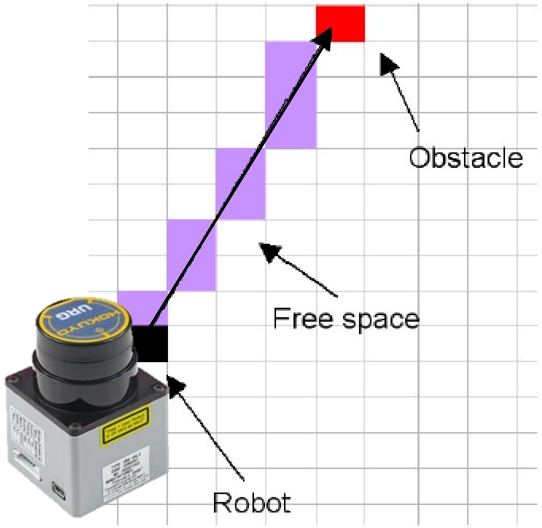
A simple 2D representation for maps

- 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



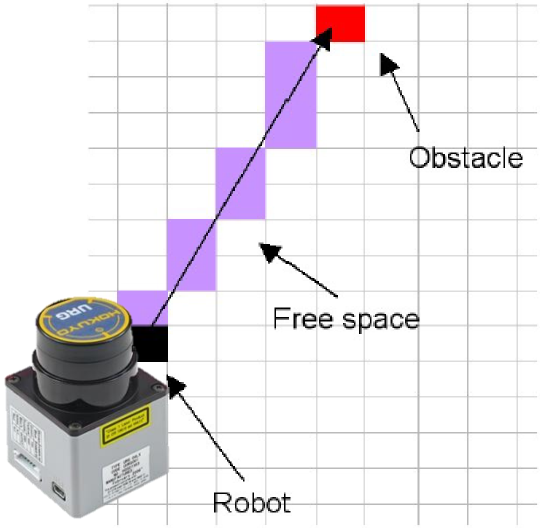
Occupancy Grid Cell Update

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 $[-\infty,\infty]$



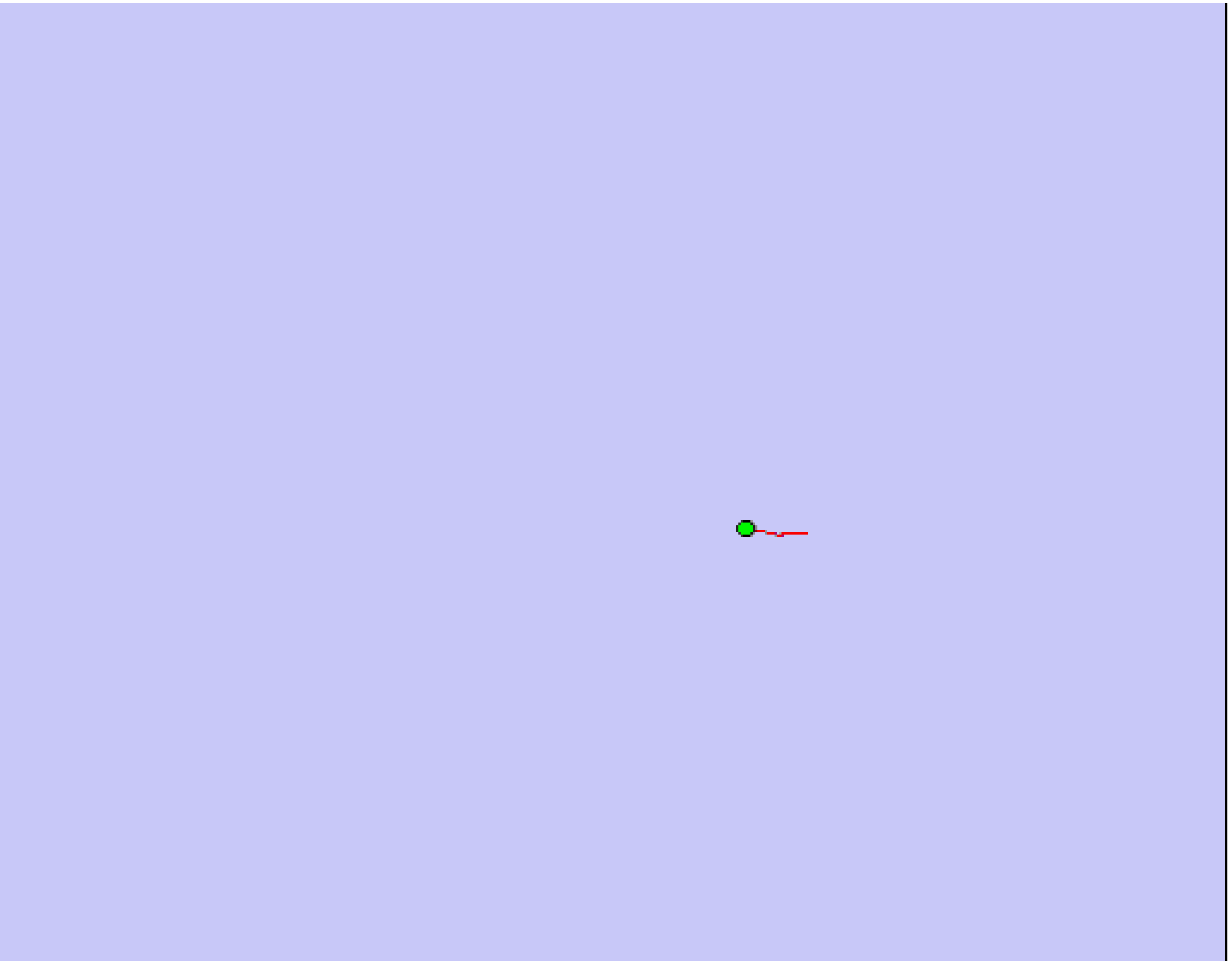
Each cell C_{ij} holds the value $\log \mathrm{o}(occ(i,j))$, $C_{ij} = 0$ corresponds to $\mathrm{P}\big(occ(i,j)\big) = 0.5$

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*.

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

current measurement

robot motion

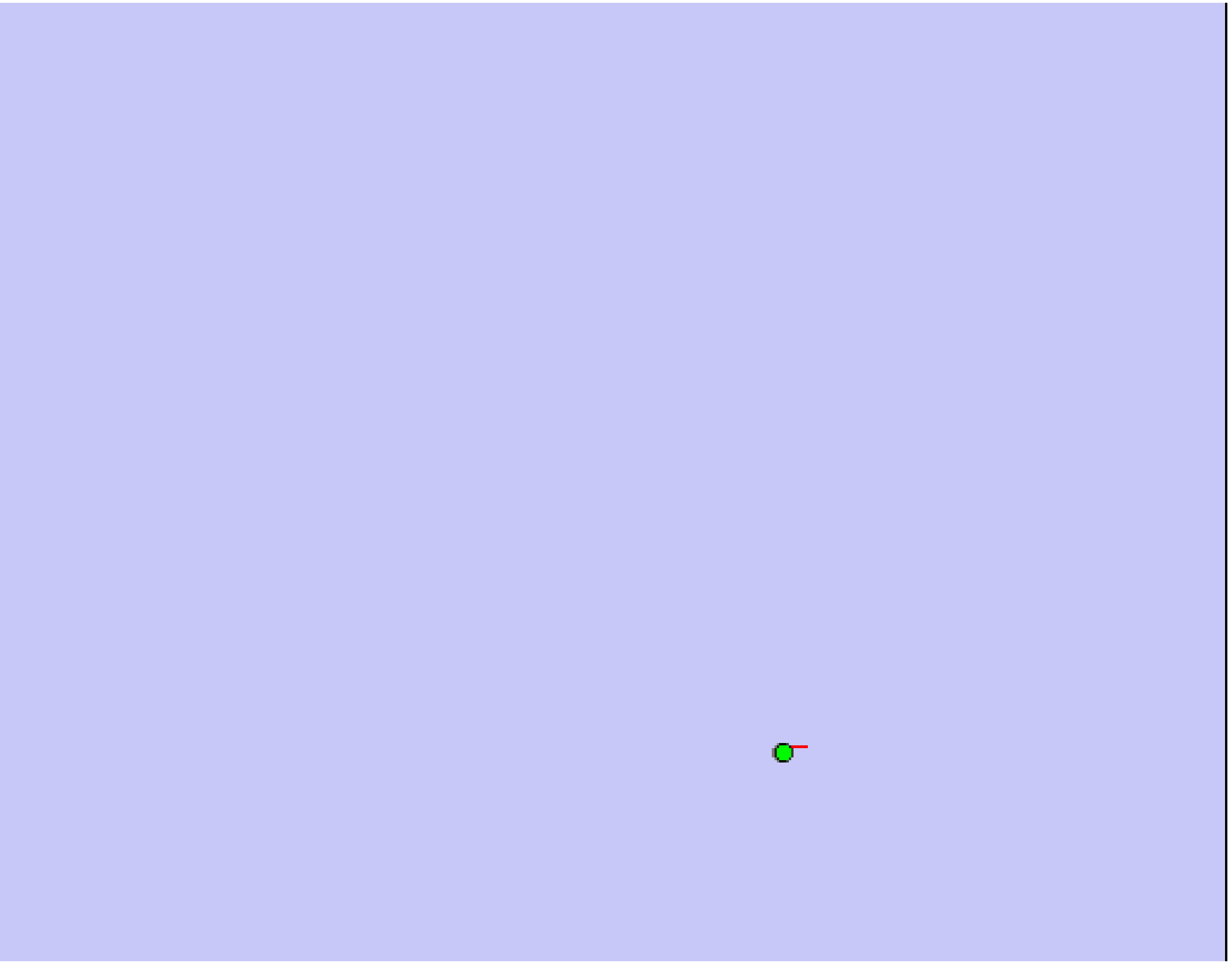
map constructed so far

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

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

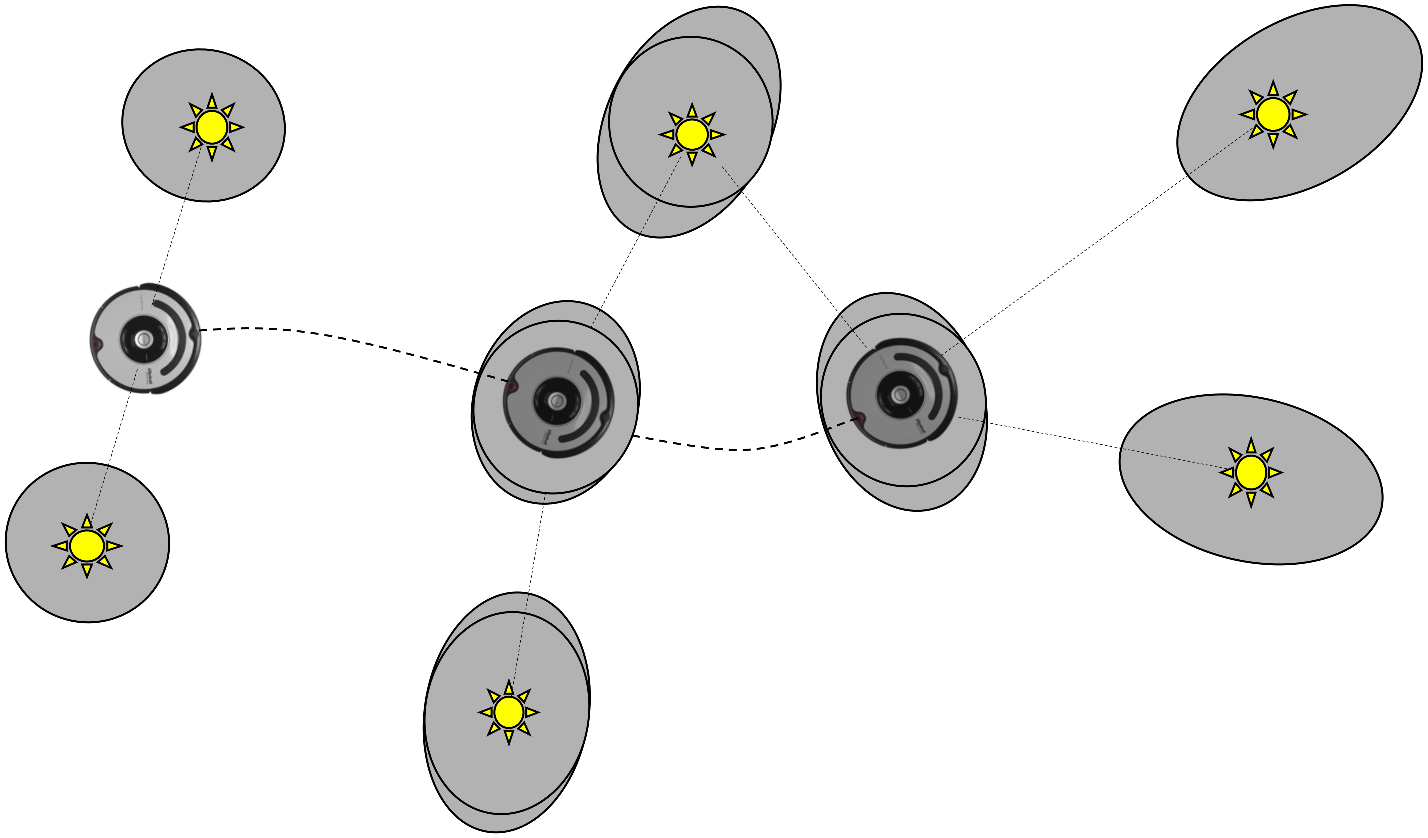
map co cted so far

Does not keep track of the uncertainty in the process

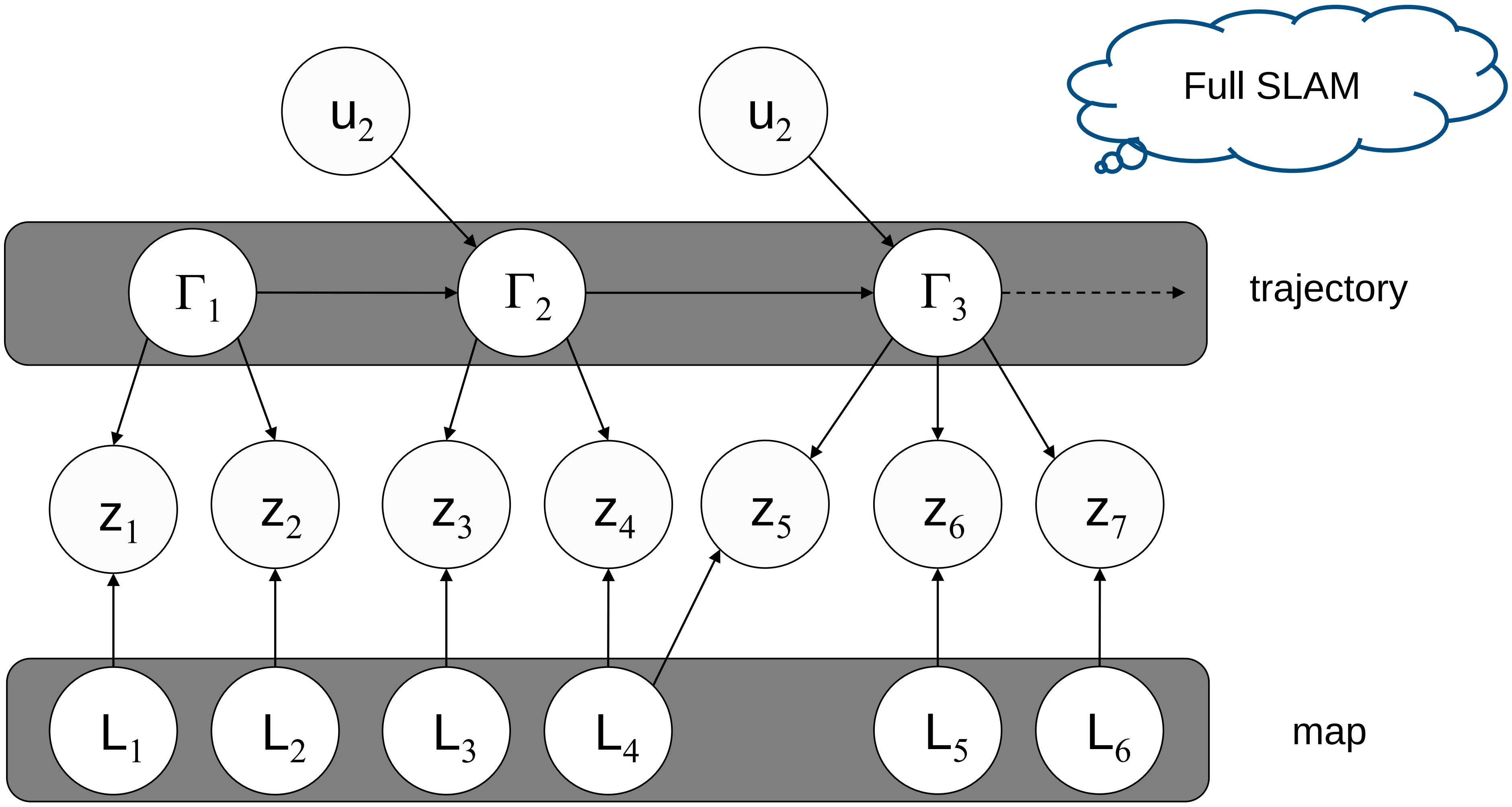
$\hat{m}^{[t]}$ The compute the map $\hat{m}^{[t]}$ according to “mapping with known poses” based on the new pose and current observations.

Iterate alternating the two steps of localization and mapping …

Simultaneous Localization and Mapping

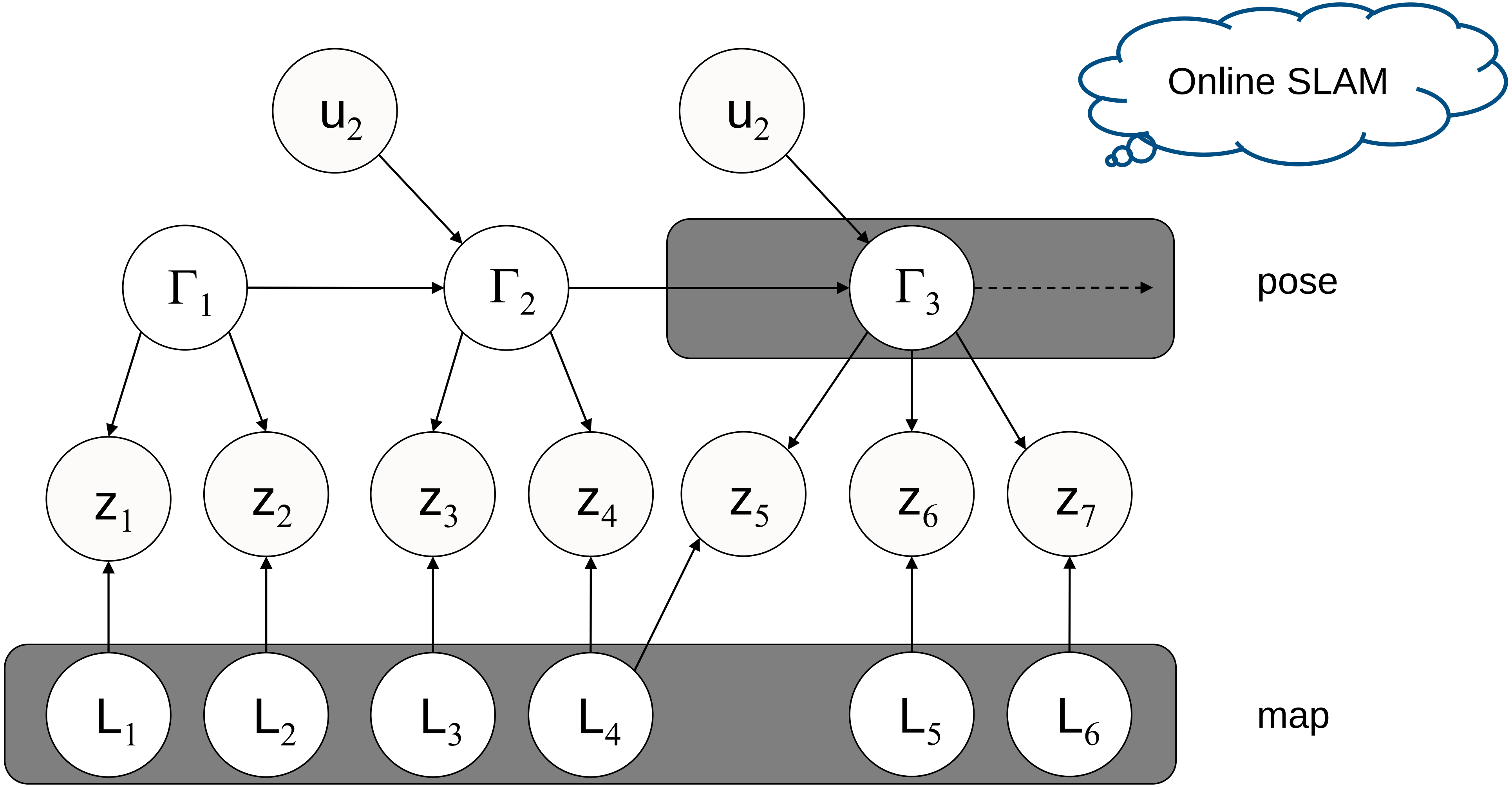


Dynamic Bayesian Networks and (Full) SLAM



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

Dynamic Bayesian Networks and (Online) SLAM



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

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

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

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

The map is part of the state

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

$$p(x_t \mid u_t, x_{t-1}) = N\left(x_t ; A_t x_{t-1} + B_t u_t, R_t\right)$$

(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

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

$$z_t = C_t x_t + \delta_t$$

$$p(z_t \mid x_t) = N\left(z_t; C_t x_t, Q_t\right)$$

Bayes Filter: The Algorithm

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

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

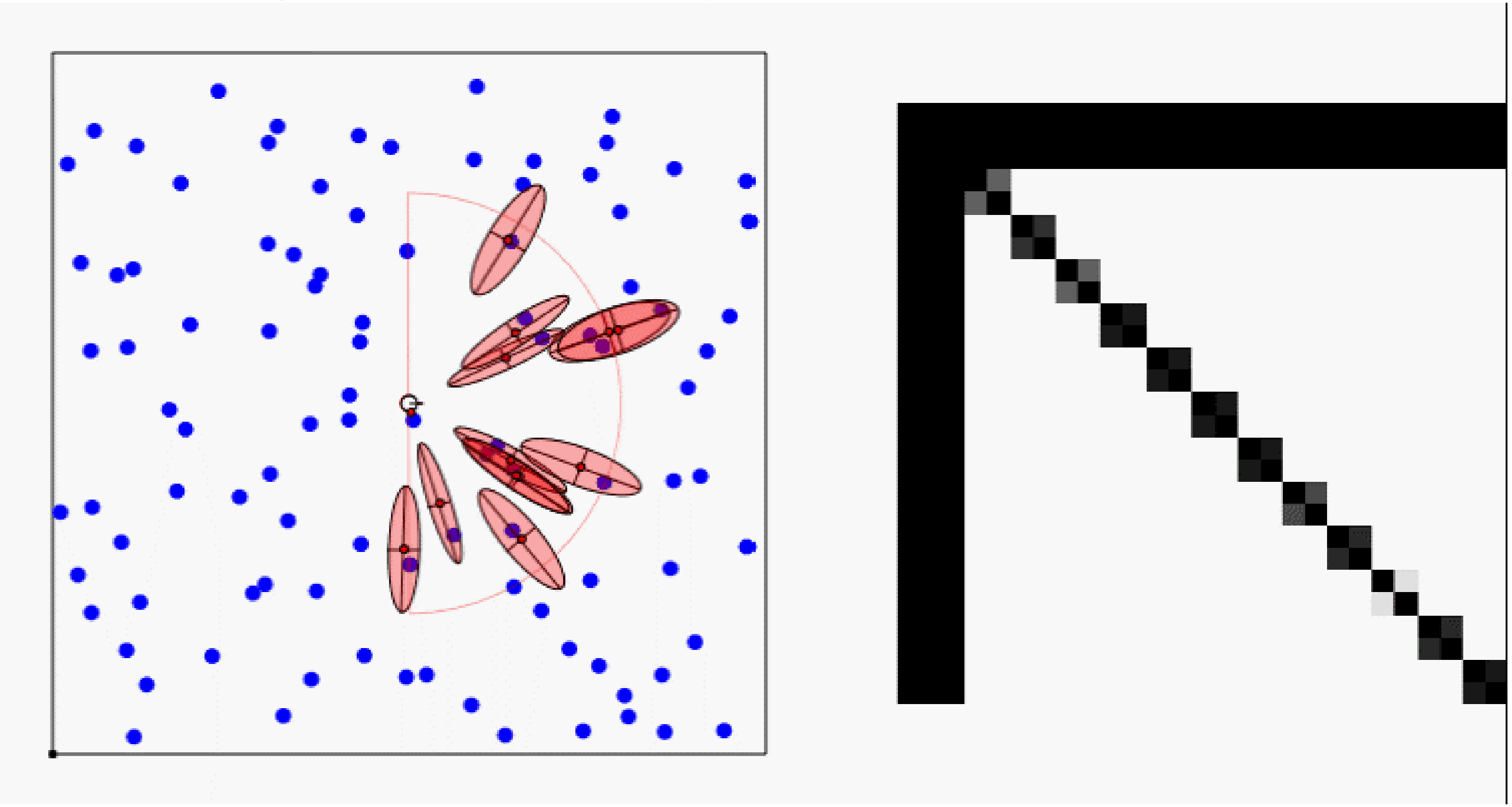
Return μ_t, Σ_t

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

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

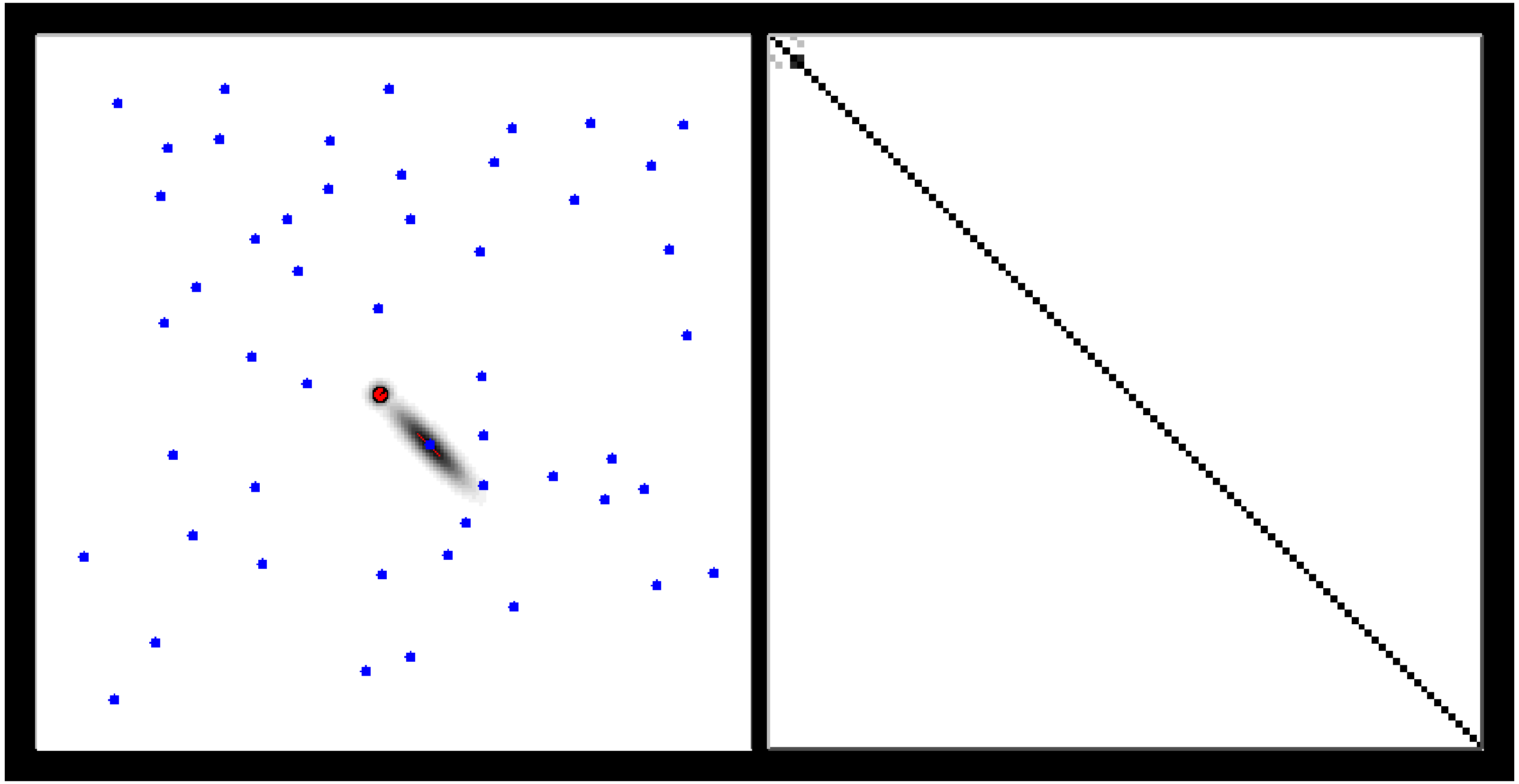
Classical Solution – The EKF

Approximate the SLAM posterior with a high-dimensional Gaussian



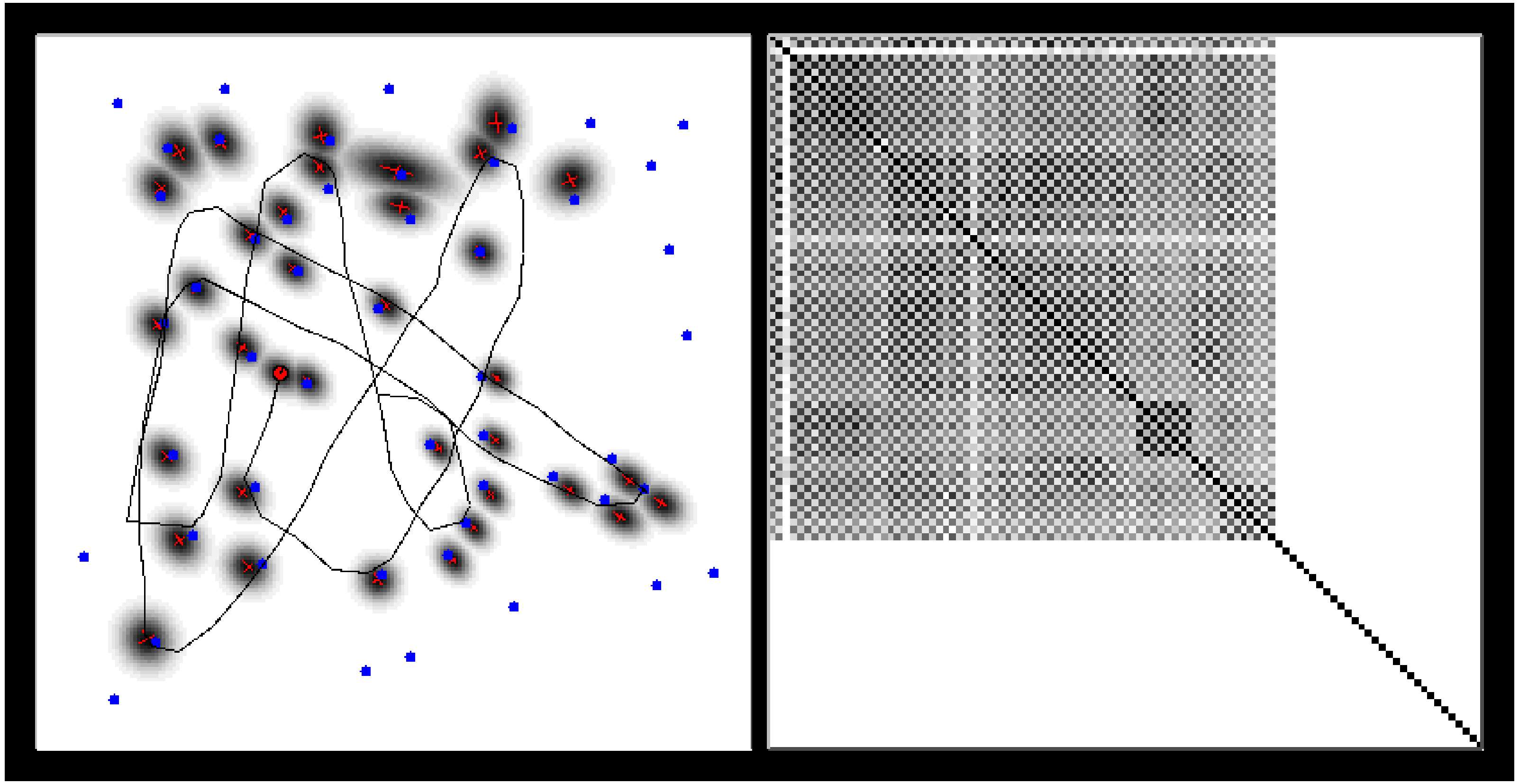
Blue path = true path **Red path** = estimated path **Black path** = odometry

EKF-SLAM



Map Correlation matrix

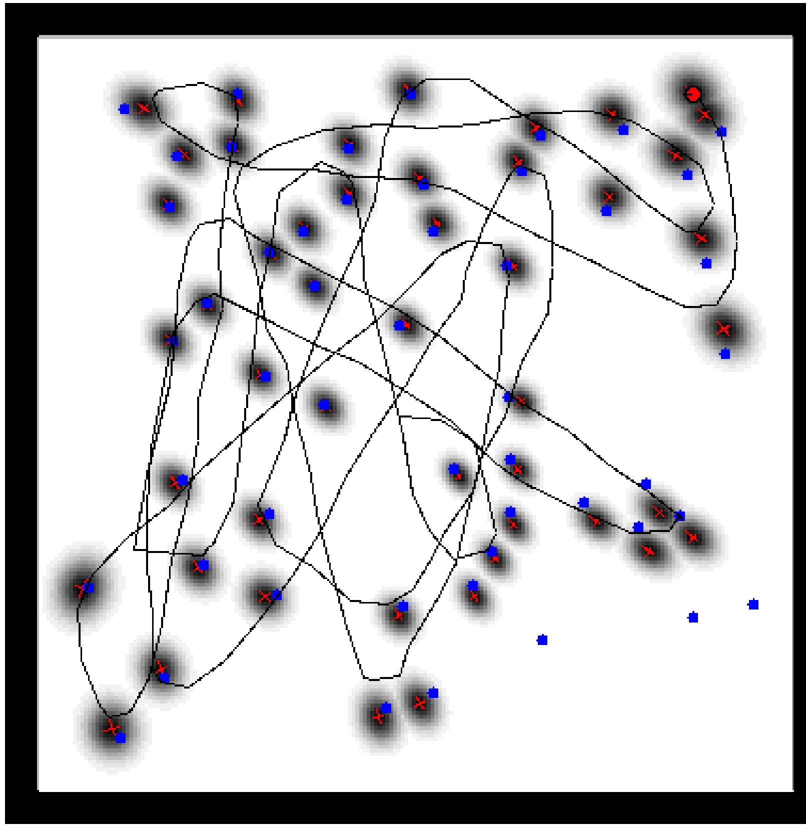
EKF-SLAM



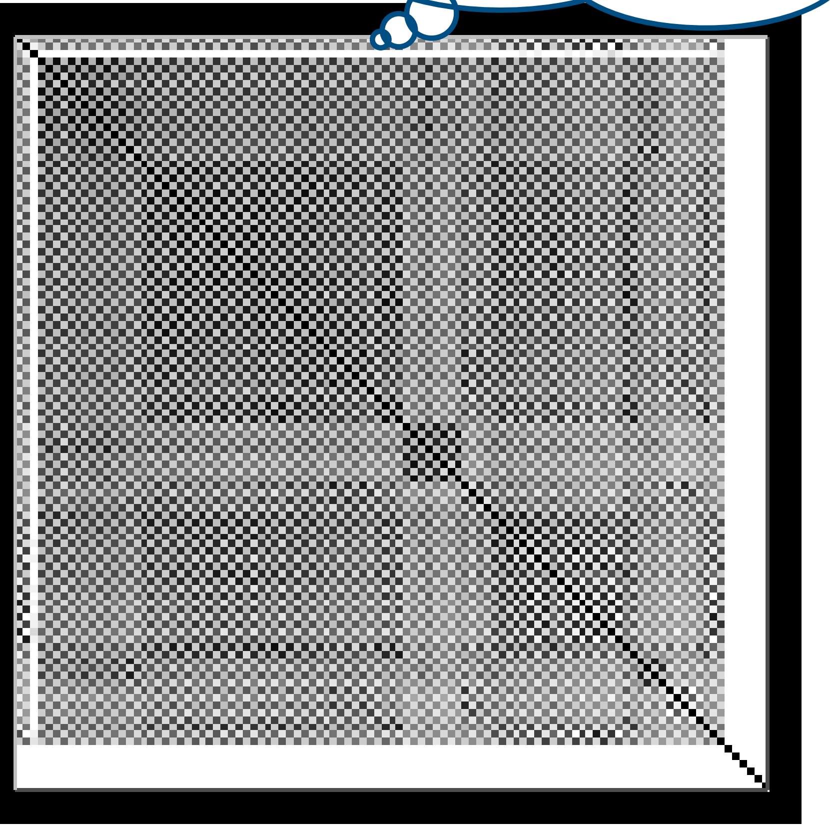
Map Correlation matrix

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.

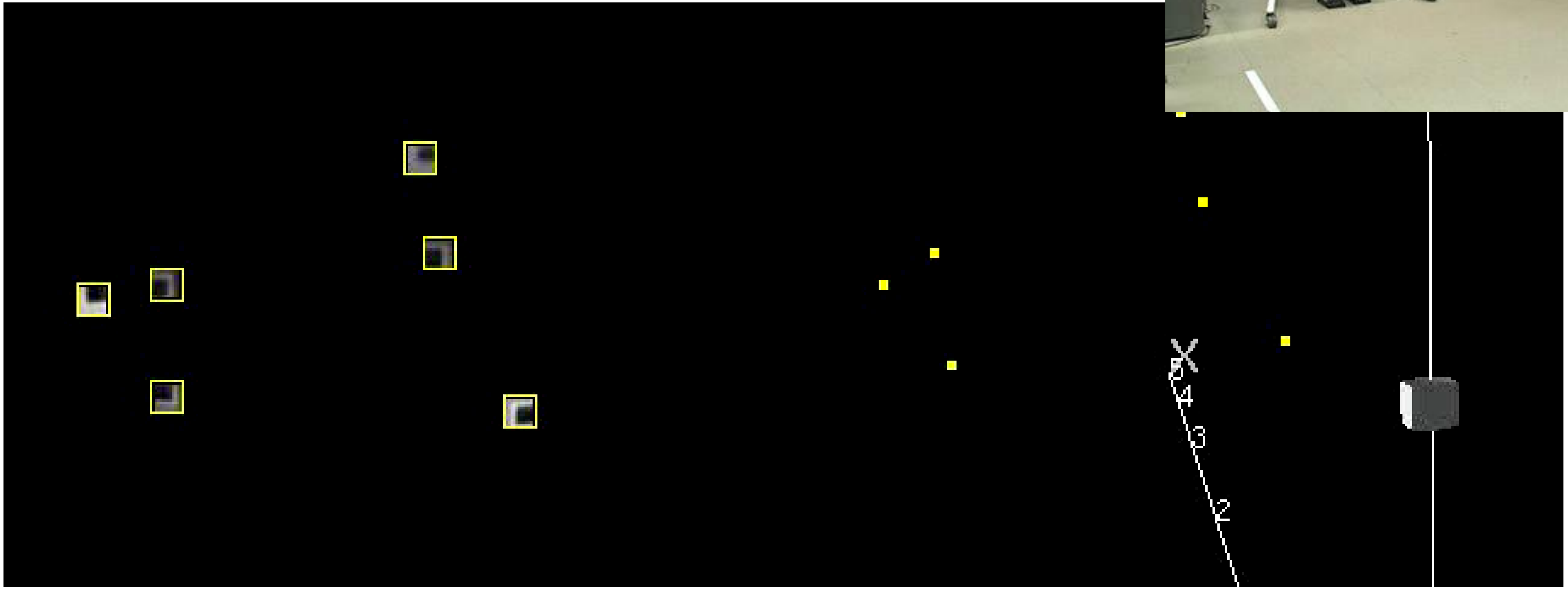
Theorem: In the limit the landmark estimates become fully correlated

[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

- $P(X_t \mid z_t)$ through particles
- $P(m \mid X_t, z_t)$ using an EKF

$$\underbrace{P(m}_{\text{map}} \mid \underbrace{X_t^{(i)}}_{\text{poses}}, z_t) = \prod_j^M P(m_j \mid X_t^{(i)}, z_t)$$

FastSLAM Formulation

Decouple map of features from pose ...

- 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

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

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

SLAM posterior

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

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

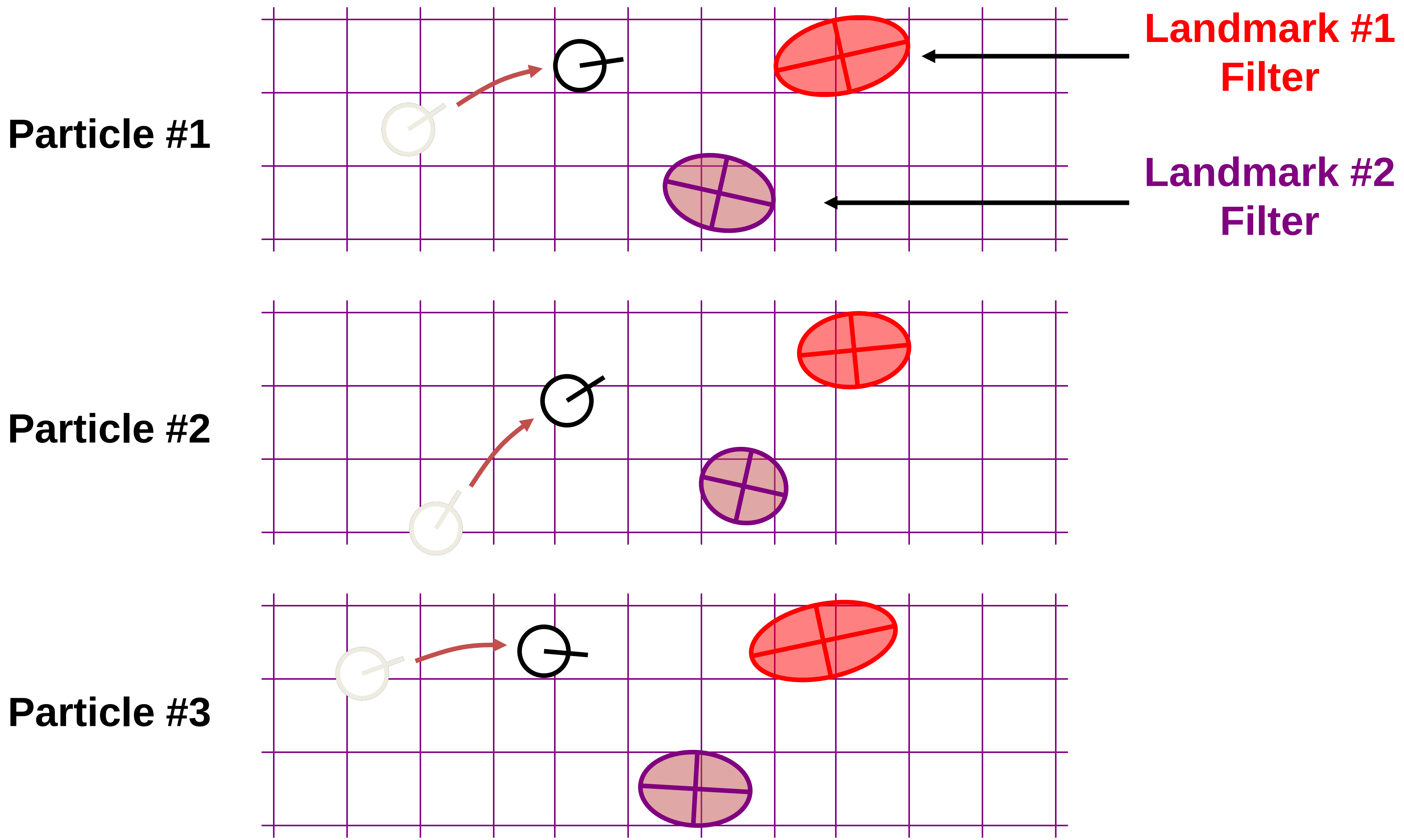
FastSLAM in Practice

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

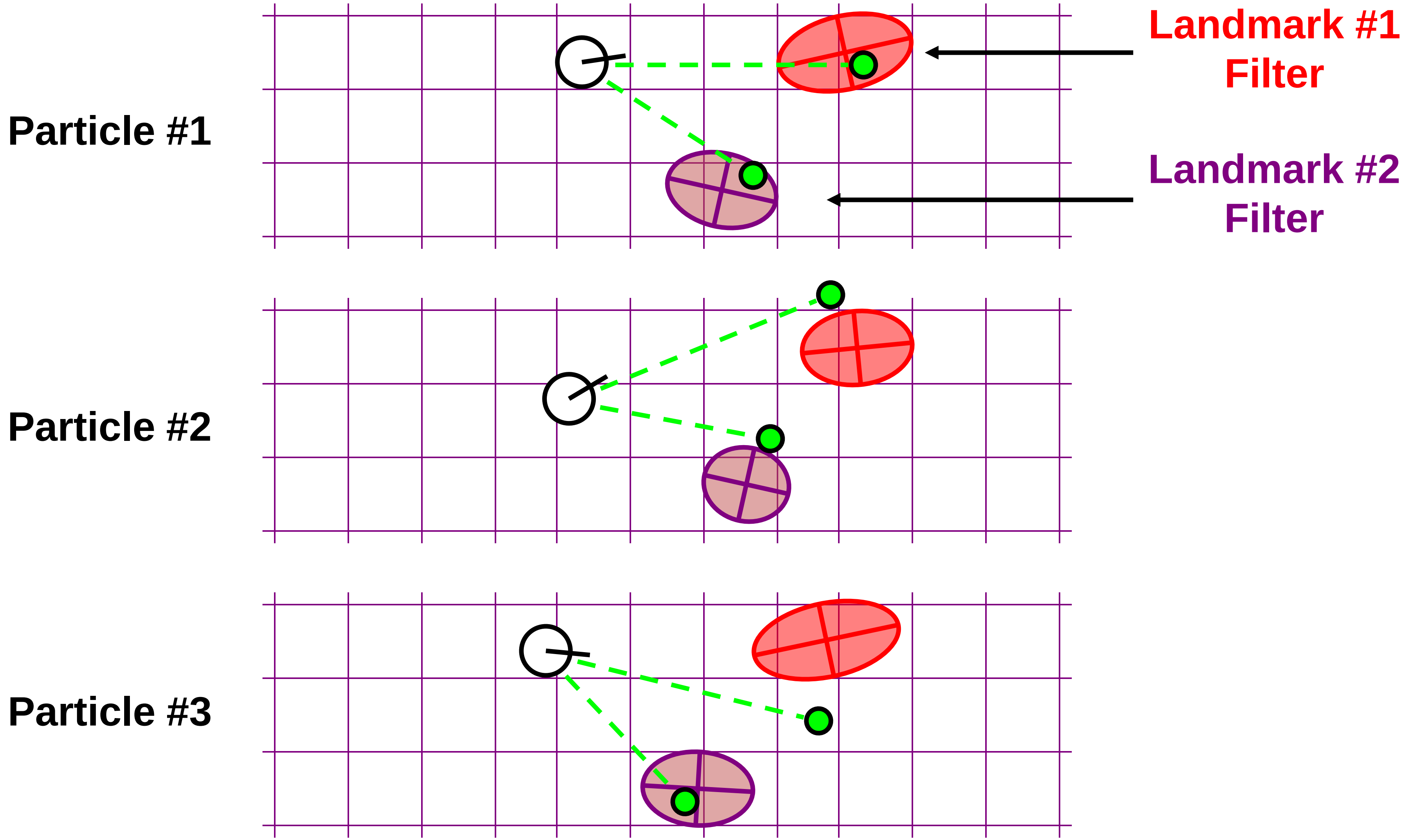
- Each particle is a trajectory (last pose + reference to previous)
- Each landmark is represented by a 2x2 Extended Kalman Filter (EKF)
- Each particle therefore has to maintain M EKFs

Particle #1	x, y, θ	**Landmark 1**	**Landmark 2**	**…**	**Landmark M**
Particle #2	x, y, θ	**Landmark 1**	**Landmark 2**	**…**	**Landmark M**
⋮					
Particle N	x, y, θ	**Landmark 1**	**Landmark 2**	**…**	**Landmark M**

FastSLAM – Action Update

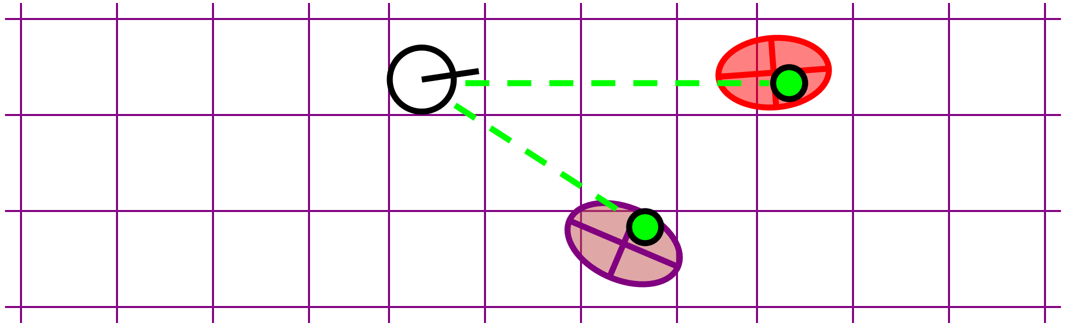


FastSLAM – Sensor Update



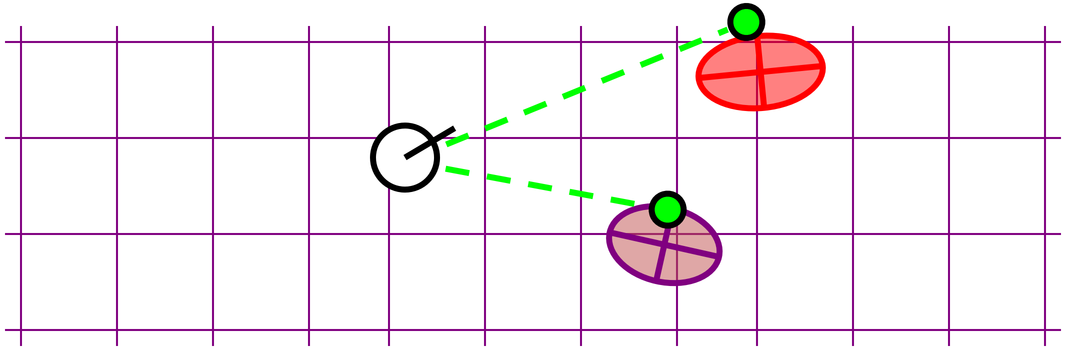
FastSLAM – Sensor Update

Particle #1



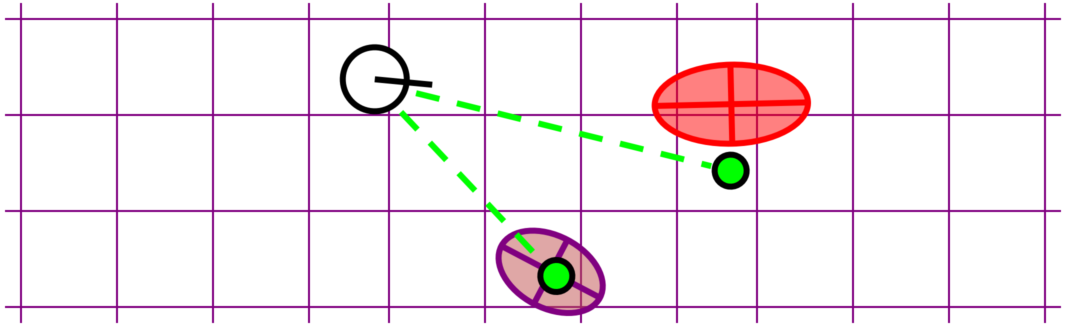
Weight = 0.8

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