

ECE 484: Principles of Safe Autonomy (Fall 2026)
Lecture 14: Review of Bayes Filters; Simultaneous Localization and Mapping (SLAM)

Professor: Huan Zhang
huanz@illinois.edu

Slides adapted from Prof. Sayan Mitra's slides for Spring 2025
Reference: Probabilistic Robotics — Thrun, Burgard and Fox

Review: Beliefs

Belief: a probability distribution over possible states.

The state is not directly measured; infer it from sensor readings.

$$bel(x_t) = p(x_t | z_{1:t}, u_{0:t-1})$$

Posterior from the available history; update in two steps:

Initially: $bel(x_0) = \pi_0$ (e.g., uniform distribution over all locations)

Prediction: $\overline{bel}(x_t) = p(x_t | z_{1:t-1}, u_{0:t-1})$ apply motion model

Correction: update $\overline{bel}(x_t)$ using the new reading z_t to get $bel(x_t)$ apply measurement model; roughly speaking, multiple/reweight $\overline{bel}(x_t)$ with $p_M(z_t | x_t)$

Discrete Bayes filter (Grid localization), particle filter (Monte-Carlo localization) , and Kalman filter are all based on the same principle

Review: three ways to represent a belief

Grid

One probability per cell

Flexible shape: several peaks can remain.

Cost: many cells in a large state space.

Resolution: coarse cells lose detail.

Particle filter

A collection of sampled states

Flexible shape: samples can keep several peaks.

Cost and accuracy: depend on particle count.

Risk: resampling can lose useful alternatives.

Kalman filter

One Gaussian: mean + covariance

Exact with linear models, a Gaussian prior and independent Gaussian noises.

Efficient in a small state space.

Restricted to **linear models**; local **approximation** for nonlinear models may be used in Extended Kalman filter (EKF)

Discrete Bayes Filter

Define: $Q = \{x_1, \dots, x_K\}$; $p_{k,t} = bel(X_t = x_k)$

Discrete_Bayes_filter$(\{p_{k,t-1}\}, u_{t-1}, z_t)$:

For each candidate k:

$\overline{p}_{k,t} = \sum_i p_D(x_k|x_i, u_{t-1}) p_{i,t-1}$

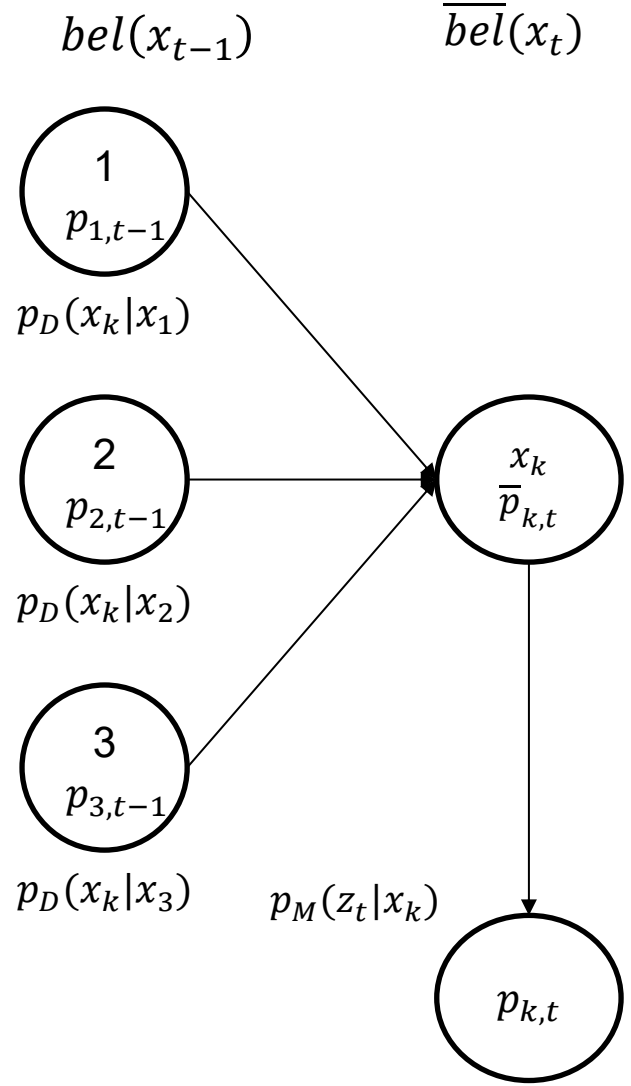
$w_{k,t} = p_M(z_t|x_k)\overline{p}_{k,t}$ (a multiplication, producing unnormalized weights)

End for

Compute $S = \sum_j w_{j,t}$; if $S > 0$, set $\eta = 1/S$. (Normalization, shared by all)

Return $p_{k,t} = \eta w_{k,t}$ for every k.

Fixed control u_{t-1} in this diagram.



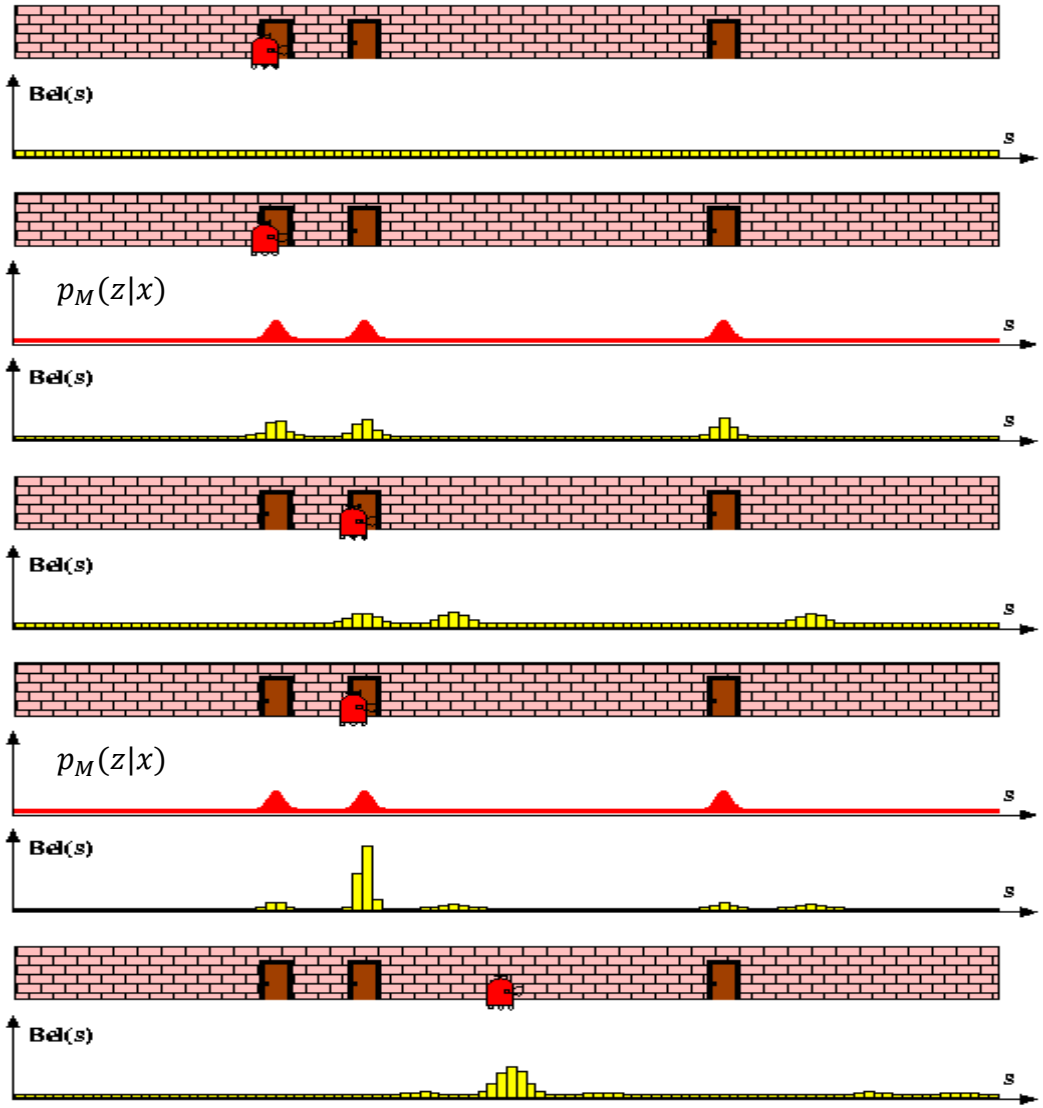
Grid localization Example: one-dimensional corridor

Correction: compare with the sensed doors.

Prediction: move all plausible positions.

A second reading can distinguish the earlier alternatives.

Here s is position along the corridor.



Initial

Correction

Prediction

Correction

Prediction

Try it yourself: [interactive grid-localization notebook]

Particle filtering algorithm

Input: M particles $\mathcal{X}_{t-1} = \{x_{t-1}^{[1]}, \dots, x_{t-1}^{[M]}\}$.

Particle_filter$(\mathcal{X}_{t-1}, u_{t-1}, z_t)$:

$\overline{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$

For $i = 1, \dots, M$:

Draw $x_t^{[i]} \sim p_D(x_t | x_{t-1}^{[i]}, u_{t-1})$. **Move particles with motion model**

$w_t^{[i]} = p_M(z_t | x_t^{[i]}, m)$ **Reweight with measurement model**

Add $(x_t^{[i]}, w_t^{[i]})$ to $\overline{\mathcal{X}}_t$.

Draw M indices **with replacement**:

sample particle j with prob. $P(j) = w_t^{[j]} / \sum_k w_t^{[k]}$

Copy the sampled $x_t^{[j]}$ into $\mathcal{X}_t$

Return $\mathcal{X}_t$.

Predict: sample candidates using the motion model p_D.

Weight: evaluate how well each candidate explains z_t.

$\overline{\mathcal{X}}_t$: temporary candidates and raw weights.

Before weighting, candidates approximate $\overline{bel}(x_t)$.

After weighting/resampling, candidates approximate $bel(x_t)$.

Resample: likely candidates can get multiple copies.

Resampling is essentially Importance Sampling*

Before resampling, particles are from $g = \overline{bel}(x_t)$; After resampling, particles are from $f = bel(x_t)$

If our particles $x^{[i]}$ are sampled from f, we should have:

$$P_f(x \in A) = E_f[I_A(x)] \approx \frac{1}{\mathrm{M}} \sum_{\mathrm{i}=1}^{\mathrm{M}} I_A(x^{[i]}) = \frac{\text{Number of particles in A}}{\mathrm{M}}$$

Number of particles in A reflects the probability that a sample drawn from f is in region A

Where $I_A(\mathrm{x})$ is 1 when x is in region A, 0 outside (indicator function)

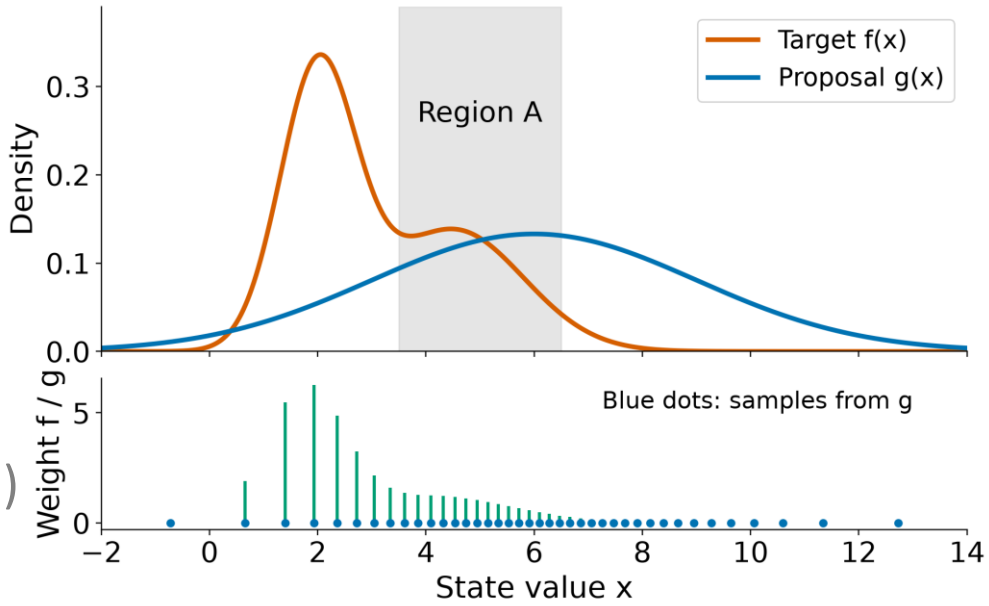
However, we have particles sampled from density g only. How to get $P_f(x \in A)$?

$$E_f[I_A] = \int f(x) I_A(x) dx$$

$$= \int \frac{f(x)}{g(x)} g(x) I_A(x) dx$$

$$= \int w(x) g(x) I_A(x) dx$$

$$= E_g[w(x) I_A(x)]$$

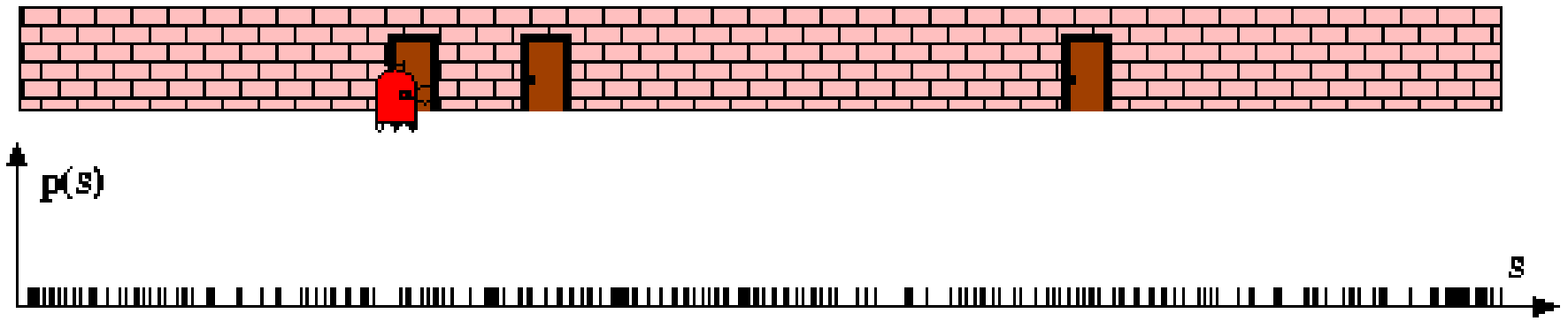


Weight: $w(x) = f(x)/g(x)$ (Need $g(x) > 0$ wherever $f(x) > 0$)

For particle filters: $w(x_t) = bel(x_t)/\overline{bel}(x_t) \propto p_M(z_t|x_t, m)$

$E_g[w(x)I_A(x)] \approx \frac{\sum_i w_i(x^{[i]})I_A(x^{[i]})}{\sum_i w_i(x^{[i]})}$, where $x^{[i]}$ are sampled from g. Normalization needed ensure weights add up to 1

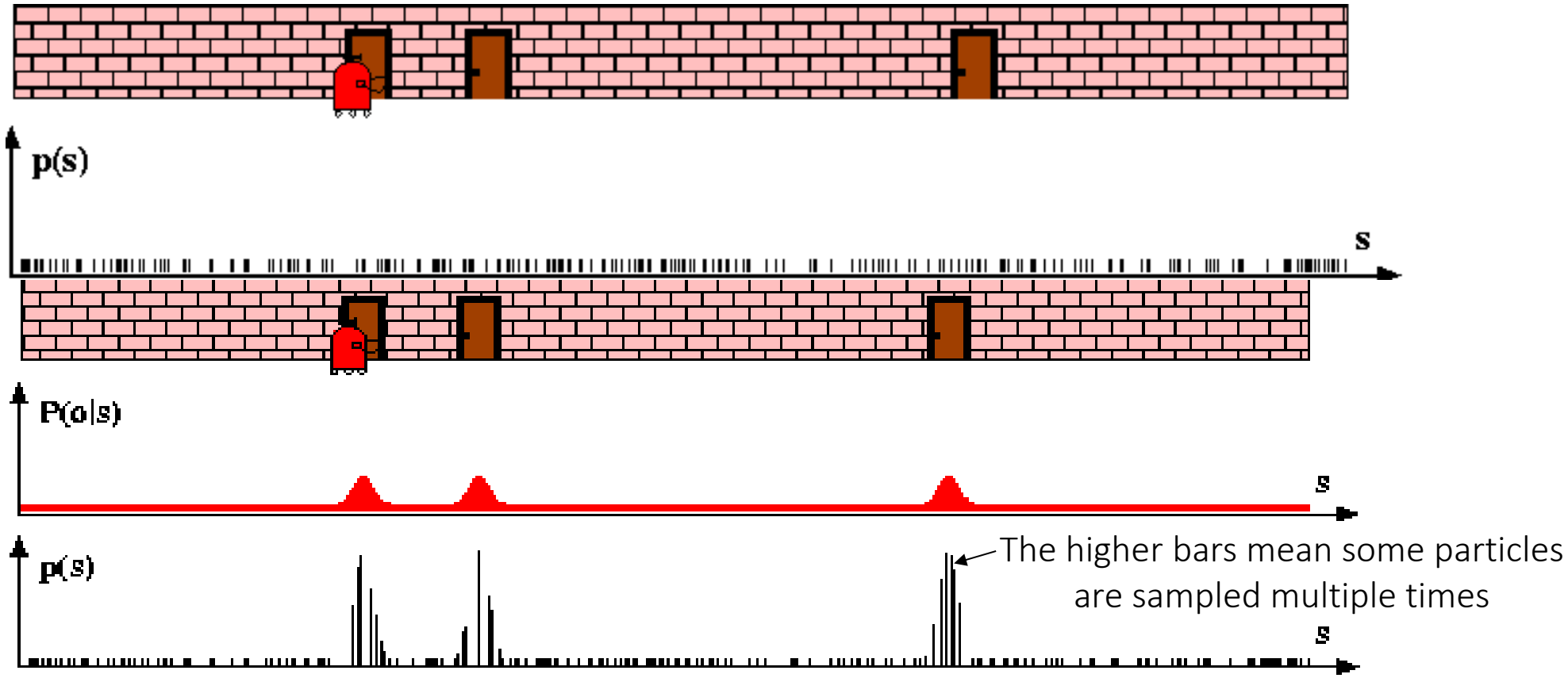
Particle Filters



Initially, particles are randomly distributed over all locations

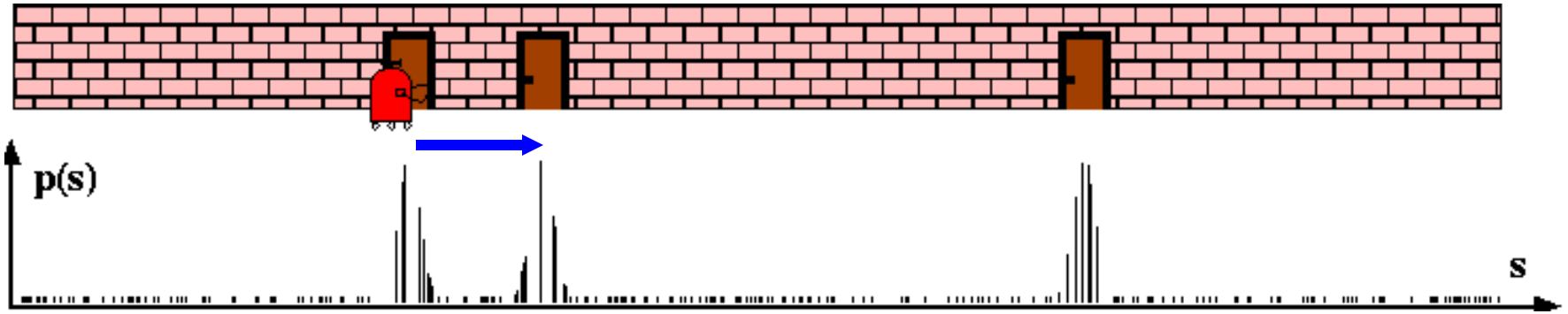
Sensor Information

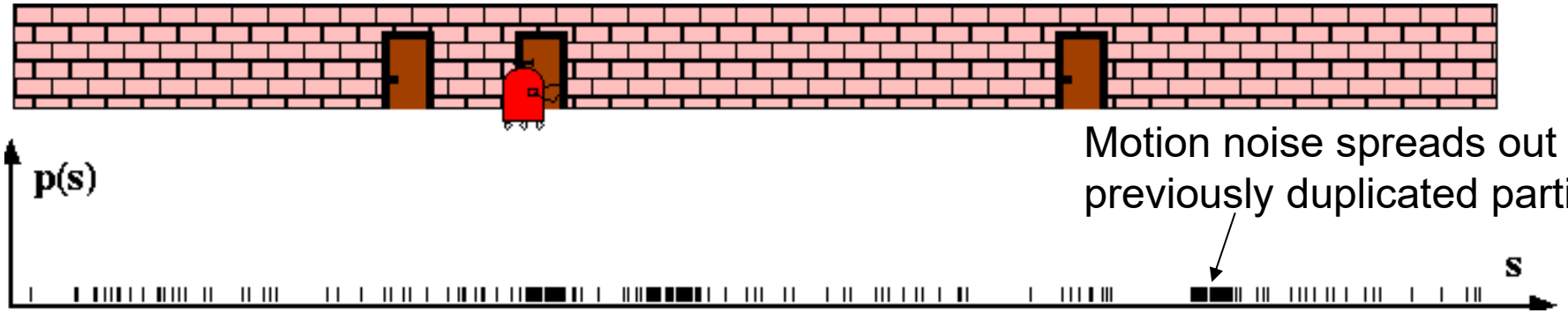
Resample the particles based on the measurement model:
particles near the doors have a higher chance of being sampled



Robot Motion

Move every particle by one step using the motion model





Motion noise spreads out previously duplicated particles.

Discrete Kalman Filter

The Kalman filter estimates a **discrete-time linear system** with Gaussian noise.

Time is discrete; states and measurements can take continuous values.

Assume Gaussian initial belief and independent, zero-mean Gaussian noises.

motion model

$$x_t = A_{t-1}x_{t-1} + B_{t-1}u_{t-1} + \varepsilon_{t-1}$$

measurement model

$$z_t = C_t x_t + \delta_t$$

observation matrix

x_t: state vector; u_{t-1}: applied input; z_t: sensor reading.

$\varepsilon_{t-1} \sim N(0, Q_{t-1})$: **process** noise.

$\delta_t \sim N(0, R_t)$: **measurement** noise.

$$p_D(x_t|x_{t-1}, u_{t-1}) = N(x_t; A_{t-1}x_{t-1} + B_{t-1}u_{t-1}, Q_{t-1})$$

$$p_M(z_t|x_t) = N(z_t; C_t x_t, R_t)$$

Kalman Filter Algorithm

Kalman_Filter($\mu_{t-1}, \Sigma_{t-1}, u_{t-1}, z_t$):

Prediction:

$$\overline{\mu}_t = A_{t-1}\mu_{t-1} + B_{t-1}u_{t-1}$$
$$\overline{\Sigma}_t = A_{t-1}\Sigma_{t-1}A_{t-1}^T + Q_{t-1}$$

Correction:

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

Return μ_t, Σ_t.

$bel(x_{t-1}) = N(x_{t-1}; \mu_{t-1}, \Sigma_{t-1})$

Prediction: affine motion plus independent noise.

Mean $\overline{\mu}_t$; covariance $\overline{\Sigma}_t$ includes Q_{t-1}.

Correction: a multiplication: prediction × measurement likelihood

$p_M(z_t|x_t) = N(z_t; C_t x_t, R_t)$

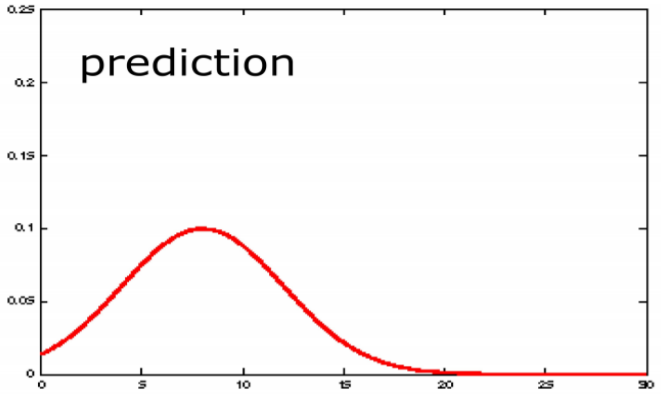
The posterior is Gaussian: both exponents are quadratic in the state.

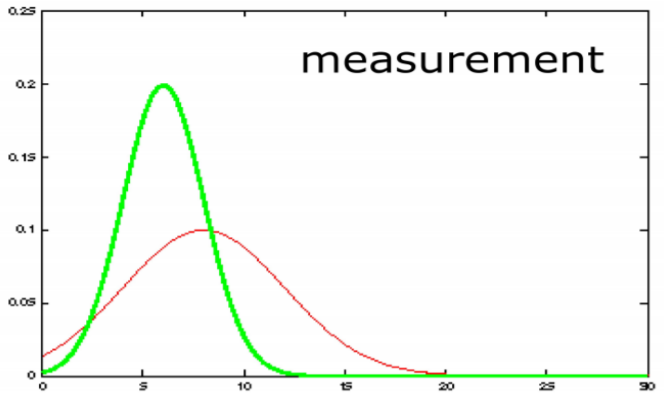
$bel(x_t) = N(x_t; \mu_t, \Sigma_t)$

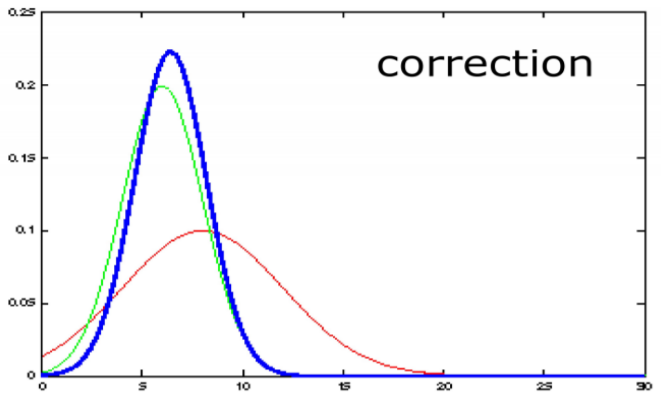
I: identity; R_t: sensor-noise covariance; C_t: observation matrix

Kalman Filter Example

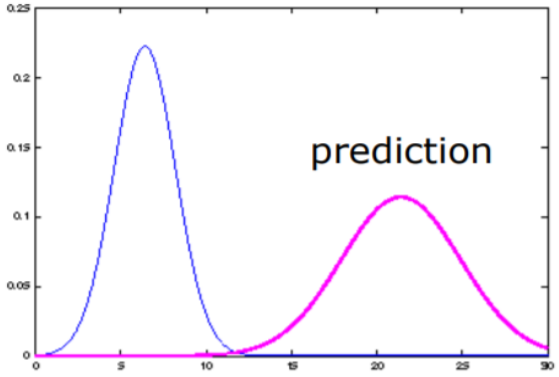
time = 1

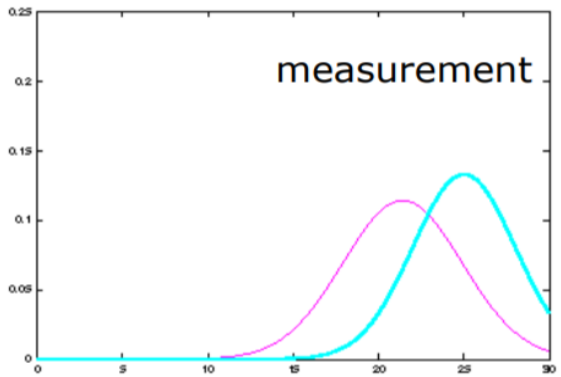






time = 2







demo: change sensor noise and observe the correction.

New this class: The SLAM Problem

SLAM: simultaneous localization and mapping

Build a map while estimating the robot's **pose relative to that map**.

Unlike known-map localization, the map is also **uncertain**.

Why is SLAM hard?

A map helps estimate the pose; a pose helps place observations in the map.



The SLAM task

A robot moving **through** an unknown, **static** environment

Given:

Controls or odometry

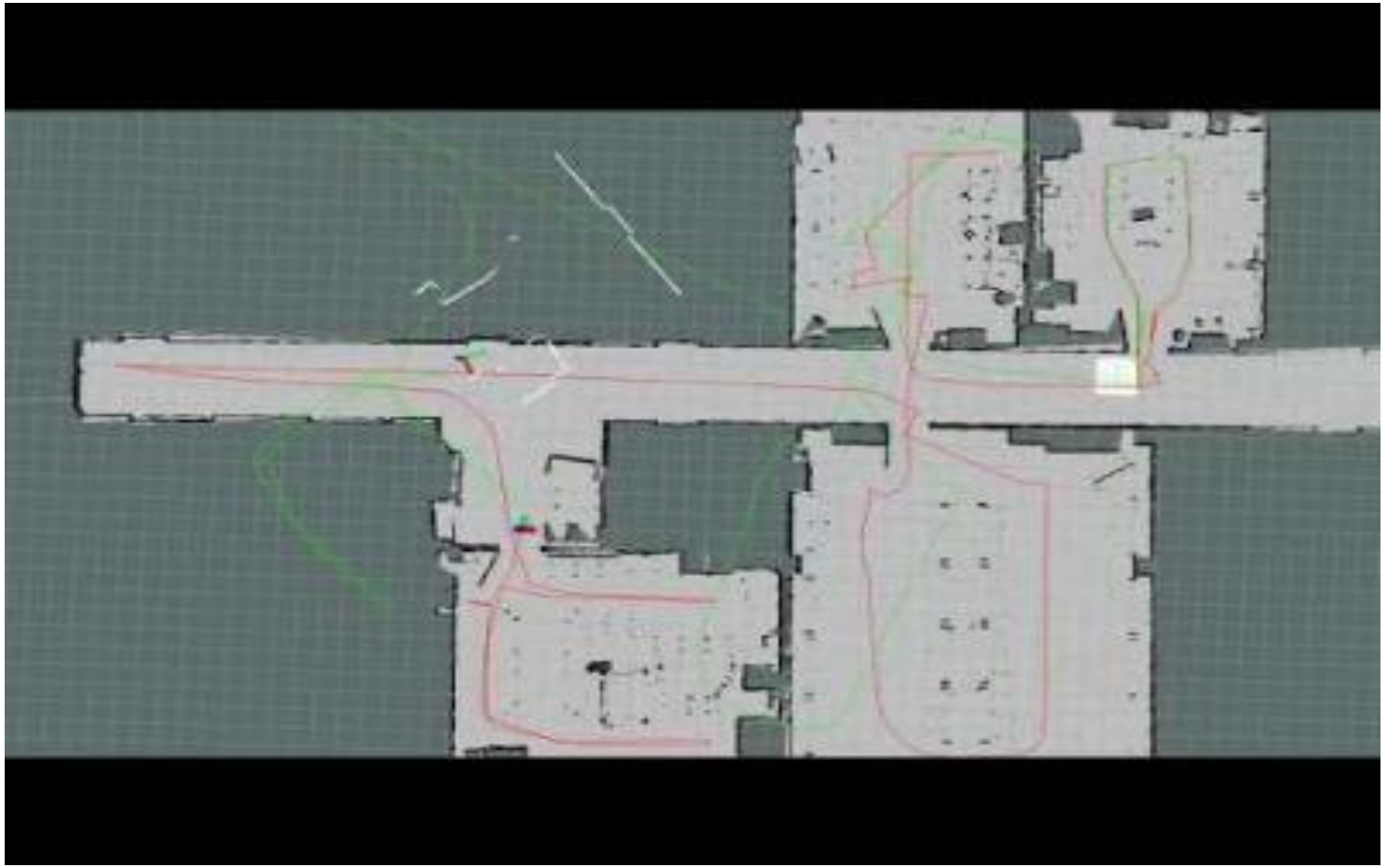
Sensor observations

Estimate:

Map

Robot path

Ask: What can revisiting a corridor tell us?



Video from Miklós Tóth https://www.youtube.com/watch?v=v4flz0AtENk

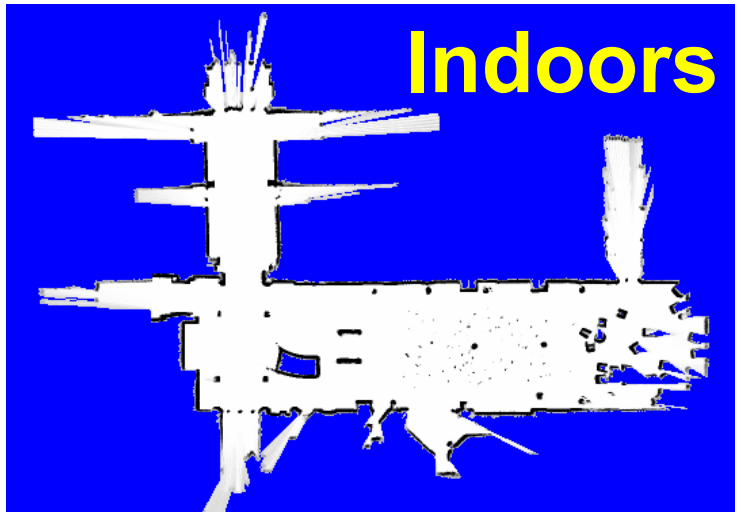
SLAM Applications

Why these settings?

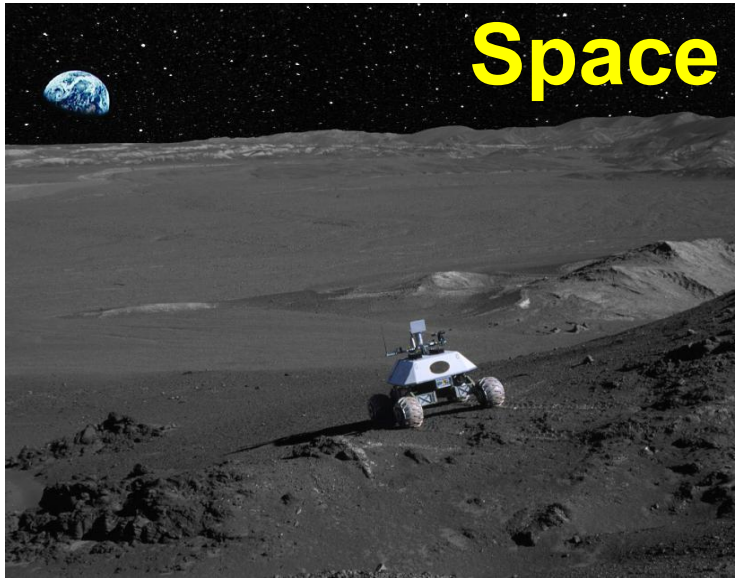
A precise prior map may be unavailable.

Motion estimates accumulate drift.

Use observations to constrain both pose and map.









Forms of SLAM

Online: current pose and map

$$p(x_t, m | z_{1:t}, u_{0:t-1})$$

Full: entire path and map

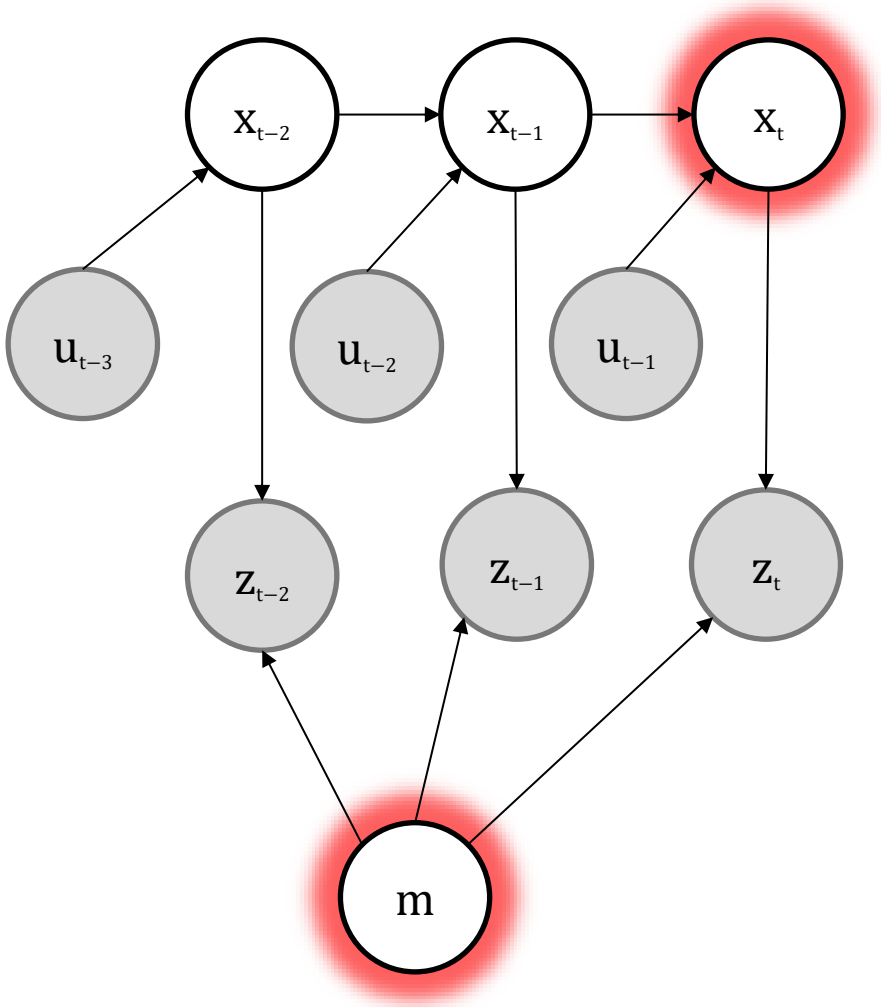
$$p(x_{0:t}, m | z_{1:t}, u_{0:t-1})$$

Continuous unknowns: poses and landmark coordinates.

Discrete unknowns: which landmark produced each observation.

x_t: pose; m: map; $z_{1:t}$: observations; $u_{0:t-1}$: controls.
$x_{0:t}$: all poses from time 0 through t.

Online SLAM



Gray = observed: controls and measurements.

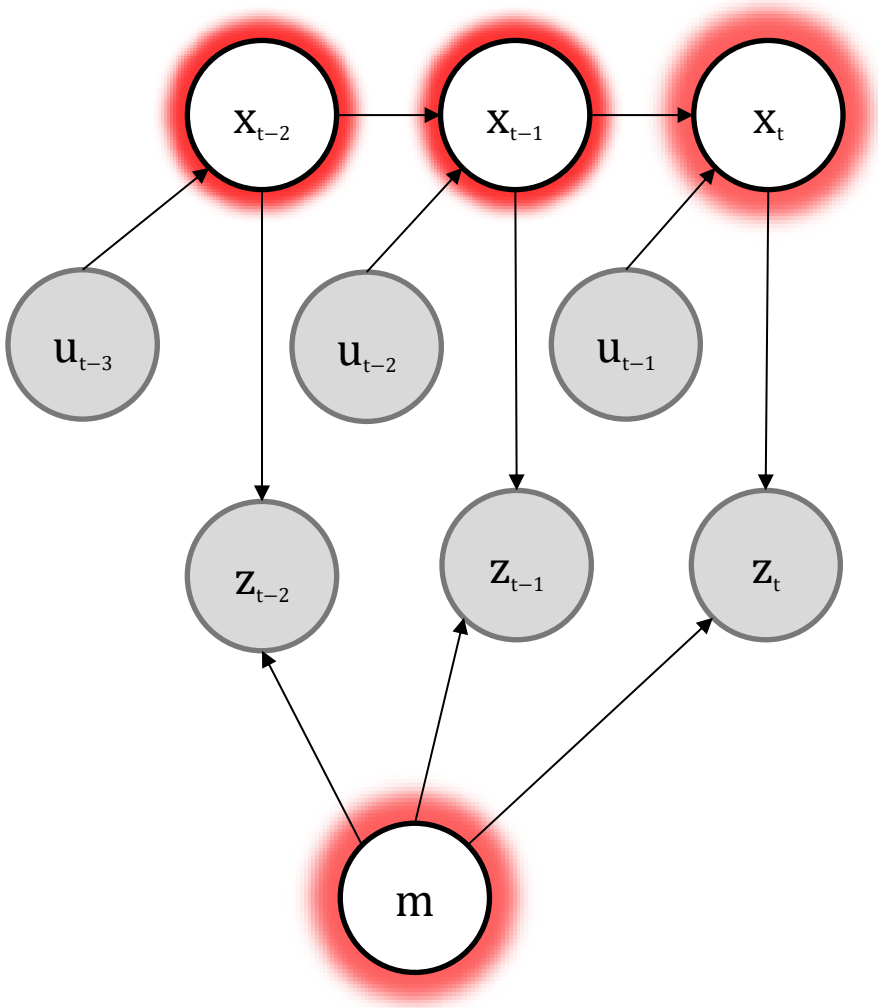
White = unknown: robot poses and map.

Keep the current pose and map:

$$p(x_t, m | z_{1:t}, u_{0:t-1})$$

Earlier poses are **marginalized out**.

Full SLAM



Gray = observed: controls and measurements.

White = unknown: robot poses and map.

Keep the entire path and map:

$$p(x_{0:t}, m | z_{1:t}, u_{0:t-1})$$

If associations are unknown:

$c_{1:t}$ identifies the observed landmarks.

Estimate $p(x_{0:t}, m, c_{1:t} | z_{1:t}, u_{0:t-1})$.

SLAM:

Simultaneous Localization and Mapping

Full SLAM: Estimates entire path and map!

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

Online SLAM:

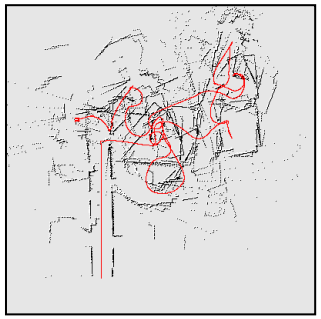
$$p(x_t, m \mid z_{1:t}, u_{1:t}) = \int\int \ldots \int p(x_{1:t}, m \mid z_{1:t}, u_{1:t})\, dx_1 dx_2 \ldots dx_{t-1}$$

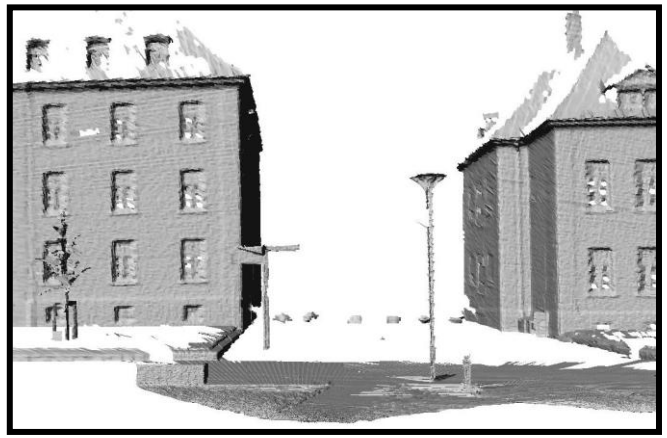
Integration of the marginals typically done one at a time

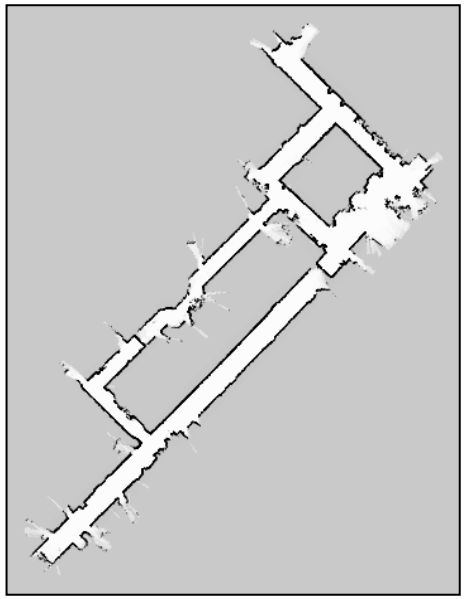
Estimates most recent pose and map!

Representations of maps

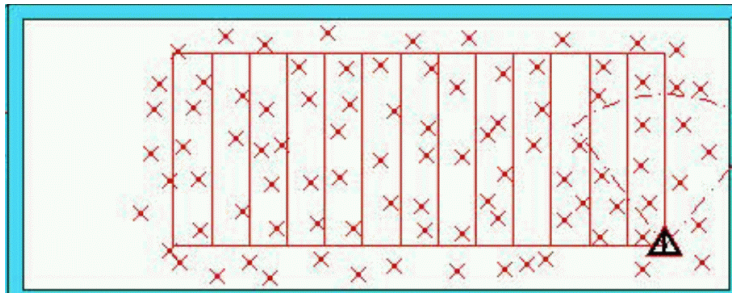
Grid maps, point clouds, or scans

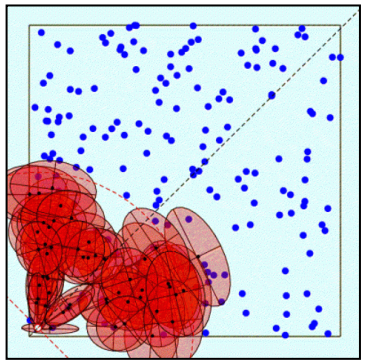


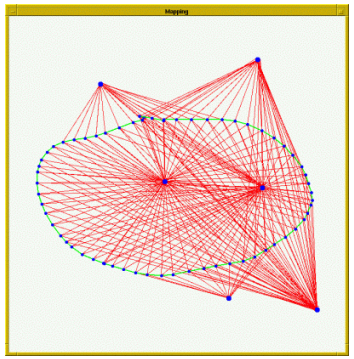




Landmark maps



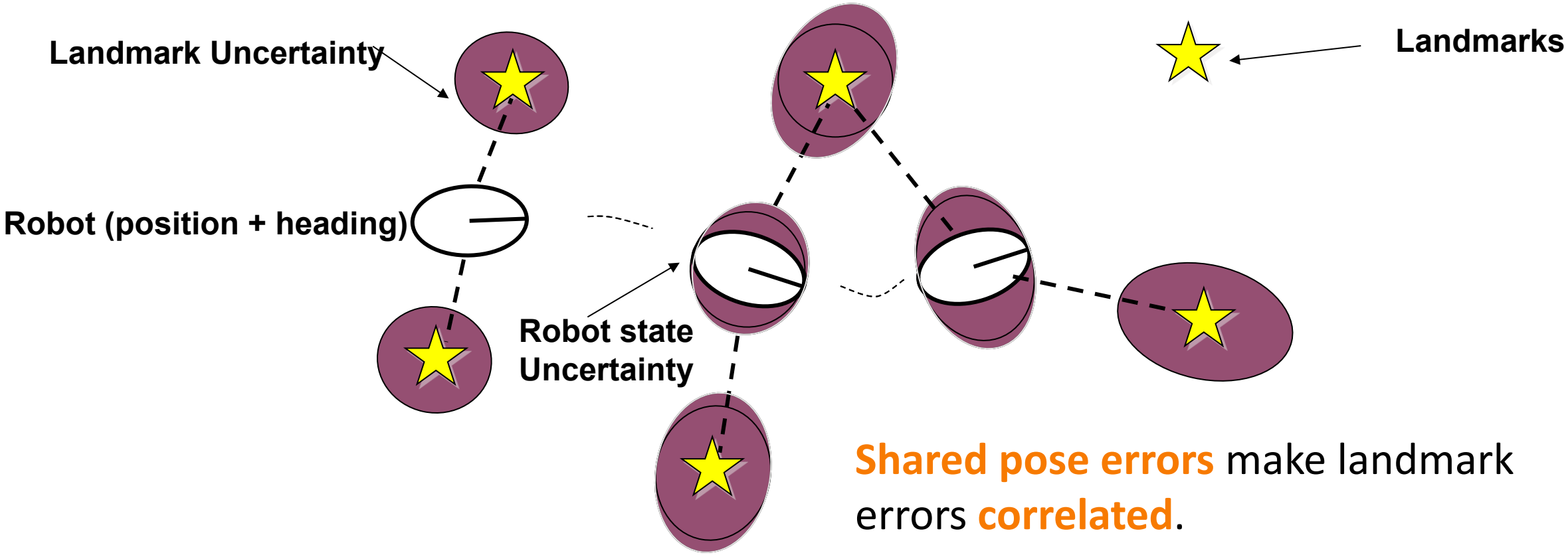




Historical examples: Lu & Milios; Gutmann; Thrun; Burgard; Konolige; Arras; Hähnel; Leonard; Castellanos; Dissanayake; Montemerlo and colleagues.

Why is SLAM a hard problem?

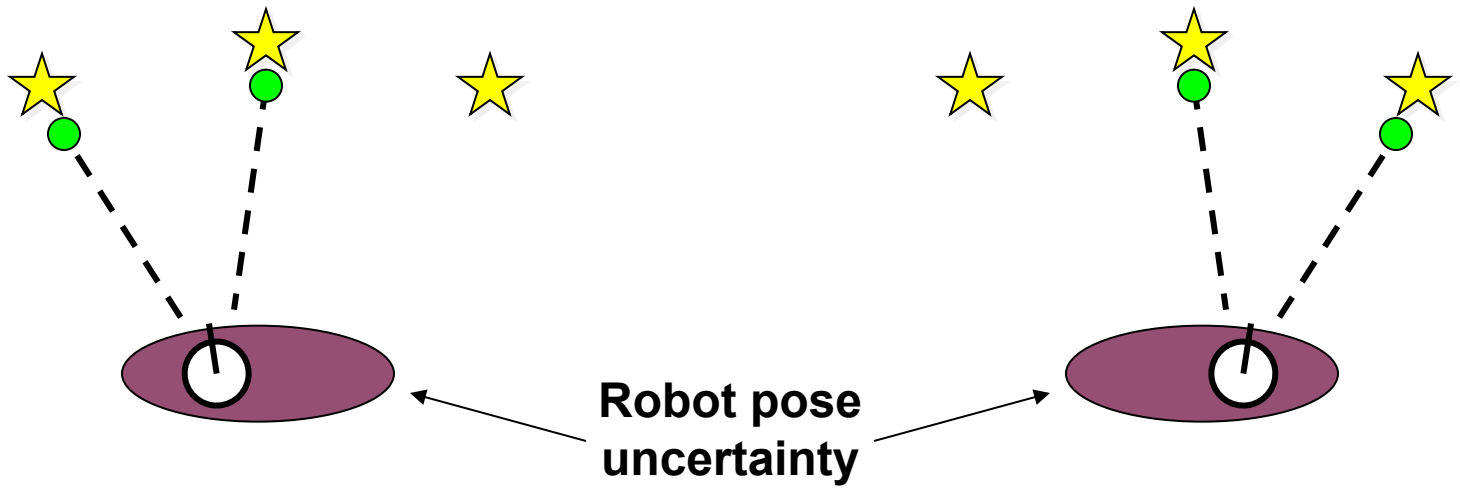
SLAM: robot path and map are both **unknown**



Shared pose errors make landmark errors **correlated**.

If the robot-position estimate shifts right by an error, every landmark inferred from that pose shifts right by that same error

Why is SLAM a hard problem?

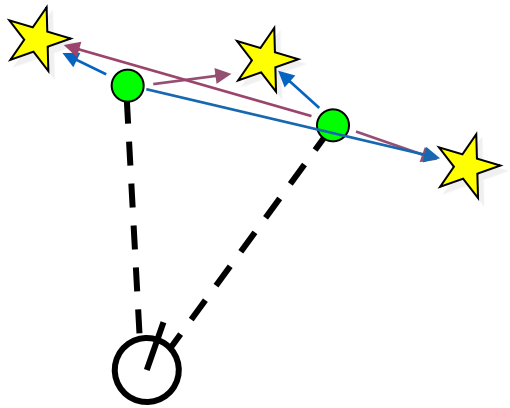


Data association: which landmark produced each observation?

A wrong match can distort **both pose and map**.

Changing the pose can change **several plausible matches together**.

Data Association Problem



Counting assumption:
Either green observation may match **any of the three stars**.

Data association assigns observations to mapped landmarks.

If we have n observations; m candidate landmarks, total $\binom{\mathrm{n}}{\mathrm{m}}$ possible assignments

Also called the **assignment problem**; new landmarks/outliers add choices.

Localization vs. SLAM

A particle filter can estimate either state:

Localization: pose $x_t = (x_B, y_B, \theta_B)$ — 3 coordinates.

SLAM: joint state $s_t = (x_t, m)$.

If map is represented as Landmark coordinates:
$m = (l_1, \dots, l_N)$, $l_j \in \mathbb{R}^2$, total: $3 + 2N$ coordinates.

If map is represented as occupancy grid:
$r \times c$ cells $q_{a,b}$; total: $3 + rc$ joint variables.

Naive joint-state particle filter

Start with M equally weighted states.

For each particle $i = 1, \dots, M$:

$$s_t^{[i]} \sim p_D(s_t | s_{t-1}^{[i]}, u_{t-1})$$
$$w_t^{[i]} = p_M(z_t | s_t^{[i]})$$

Normalize the weights.

Draw M indices **with replacement**.

Copy the selected joint states.

Reset each output weight to $1/M$.

Return the new particle collection.

Challenge: sampling pose and every map variable can spread particles too thinly.
High-dimensional importance weights can concentrate on very few particles.

Dependencies

Can the structure of SLAM reduce the amount we sample?

Map estimates share uncertainty about the robot's **path**.

If the **whole path** and **data associations** were known:

- Each reading could be placed in a known sensor frame.
- Each landmark could be estimated from its own readings.

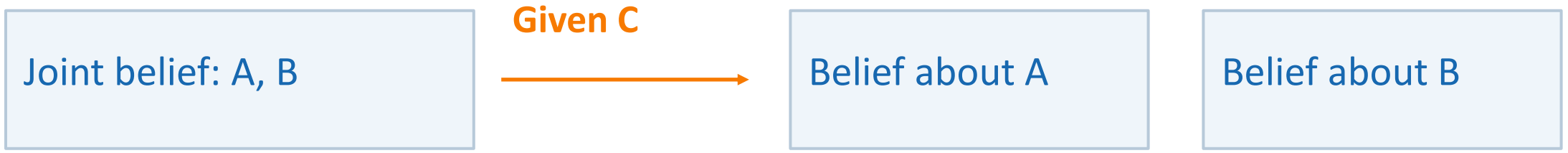
Next: when does this make landmarks conditionally independent?

Conditional Independence

A and B are **conditionally independent given** C when

$p(A, B|C) = p(A|C)\, p(B|C)$.

Once C is known, learning B gives no extra information about A.



Two smaller distributions can replace one large joint distribution.

Conditional Independence

x: robot pose; m: fixed map; z_1, z_2: sonar and laser readings.

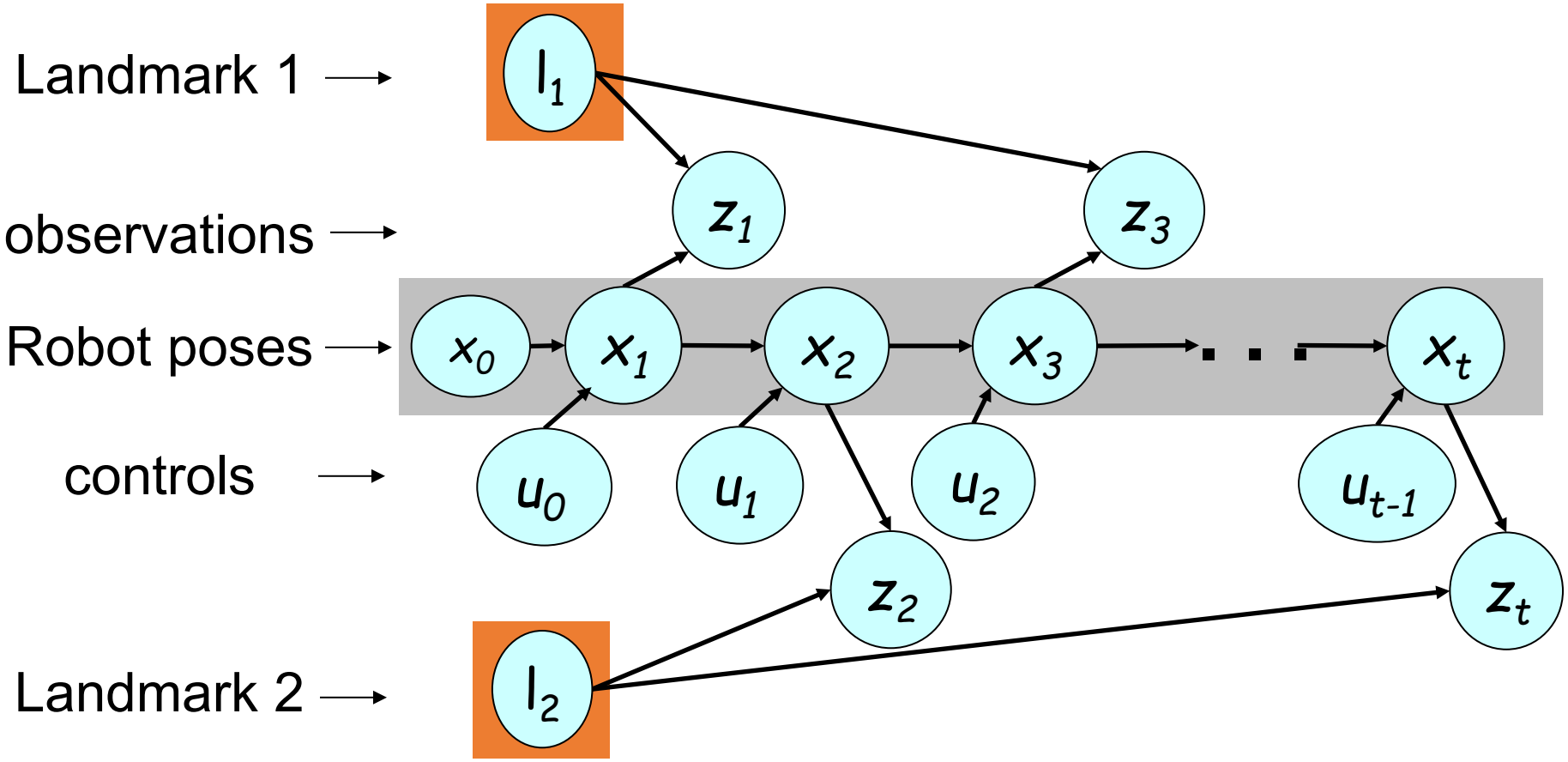
Assumption: the two sensors **have independent errors** once pose and map are fixed.

$$p(z_1, z_2 | x, m) = p(z_1 | x, m)\, p(z_2 | x, m).$$

Given the robot's true position x, each sensor's reading depends only on that position, not on what the other sensor reads.

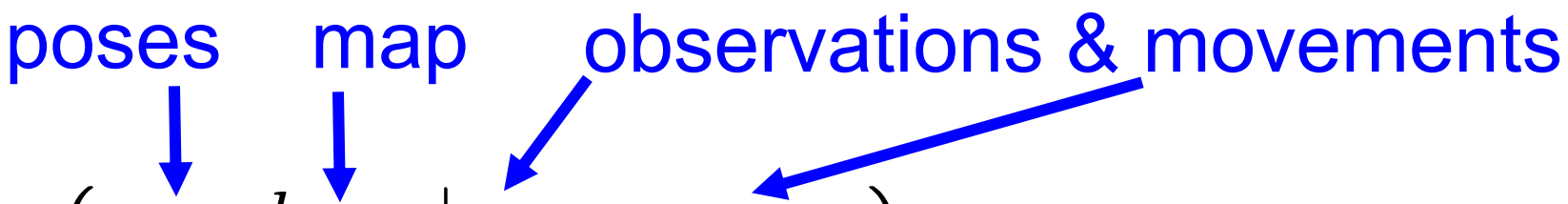


Mapping using Landmarks



Knowledge of the robot's true path renders landmark positions conditionally independent

Factored Posterior (Landmarks)



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

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

P(A , B | C, D) = P(B | C, D) P(A|B, C, D)

Factorization first introduced by Murphy in 1999

Factored Posterior (Landmarks)

poses map observations & movements

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

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

SLAM posterior

Robot path posterior

landmark positions

Does this help to solve the problem?

Factorization first introduced by Murphy in 1999

Factored Posterior: using conditionally independence

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

Robot path posterior (localization problem)

Conditionally independent landmark positions

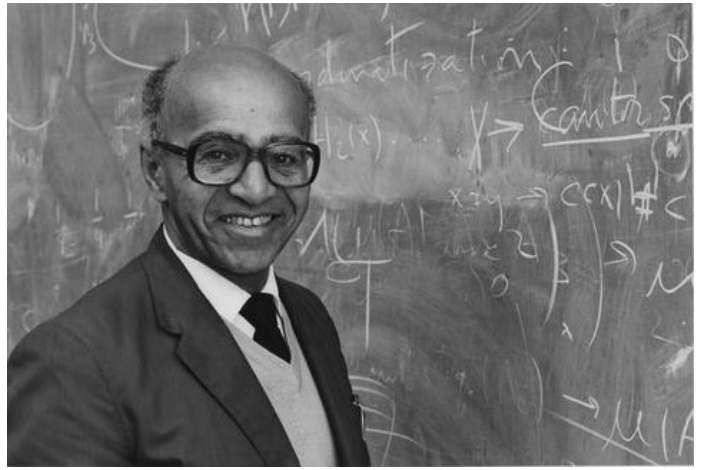
Rao-Blackwellization

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

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

- This factorization is also called **Rao-Blackwellization**
- No giant distributions, can be modeled using M separate distributions (e.g., Gaussian)
- Given that the second term can be computed efficiently, particle filtering becomes possible!

David H. Blackwell (1919-2010)



Foundational contributions to **dynamic programming**, renewal theory and statistics.

The **Rao-Blackwell theorem** motivates computing conditional averages instead of sampling everything.

University of Illinois: **BA 1938, MA 1939, PhD 1941**.

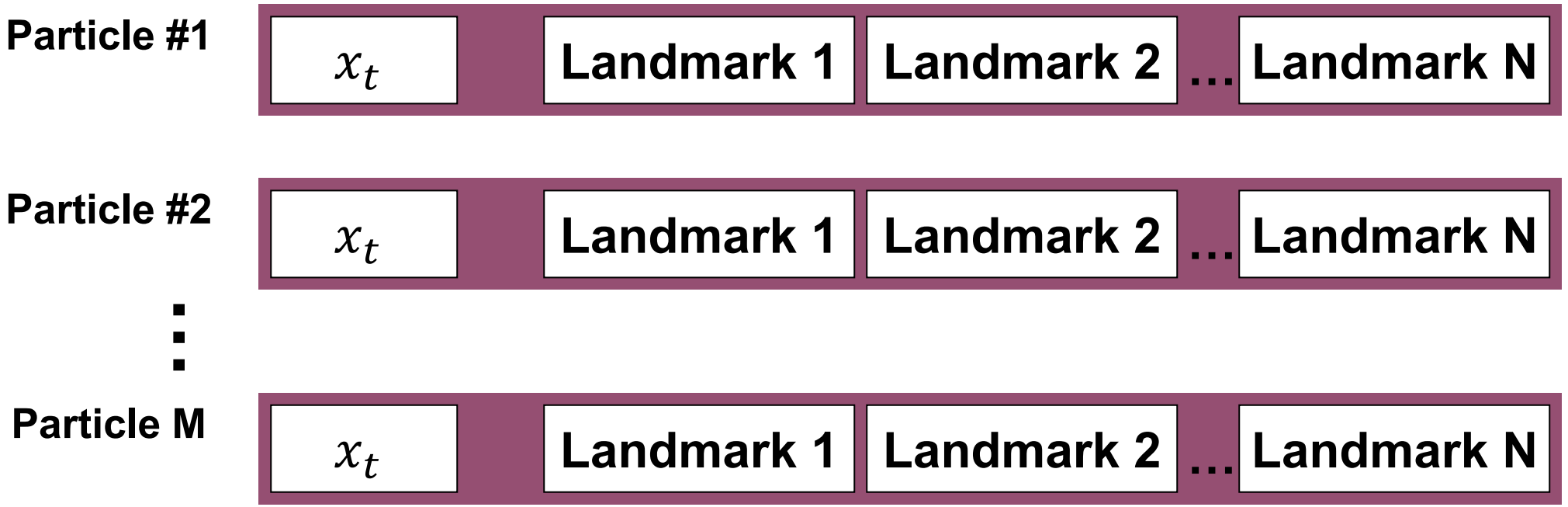
photo from stat.illinois

FastSLAM

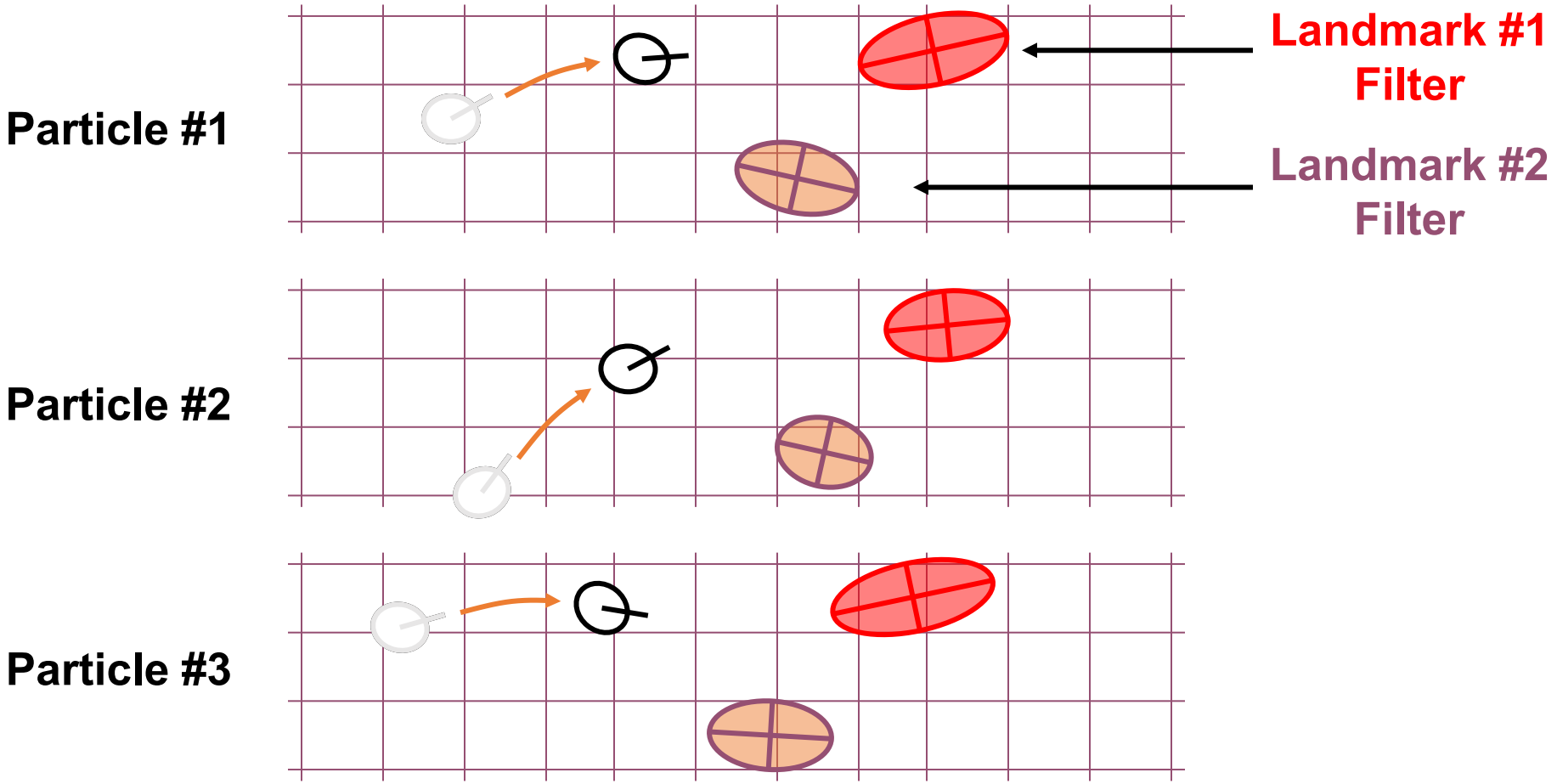
FastSLAM: a **particle filter** over robot paths, with **EKFs for the landmarks in each particle's** map
Each particle has a path and map; x_t is its current pose.

Each 2-D landmark has a 2-vector mean and a 2 × 2 covariance, updated by an EKF.

M particles; N landmarks: each particle maintains N landmark filters.

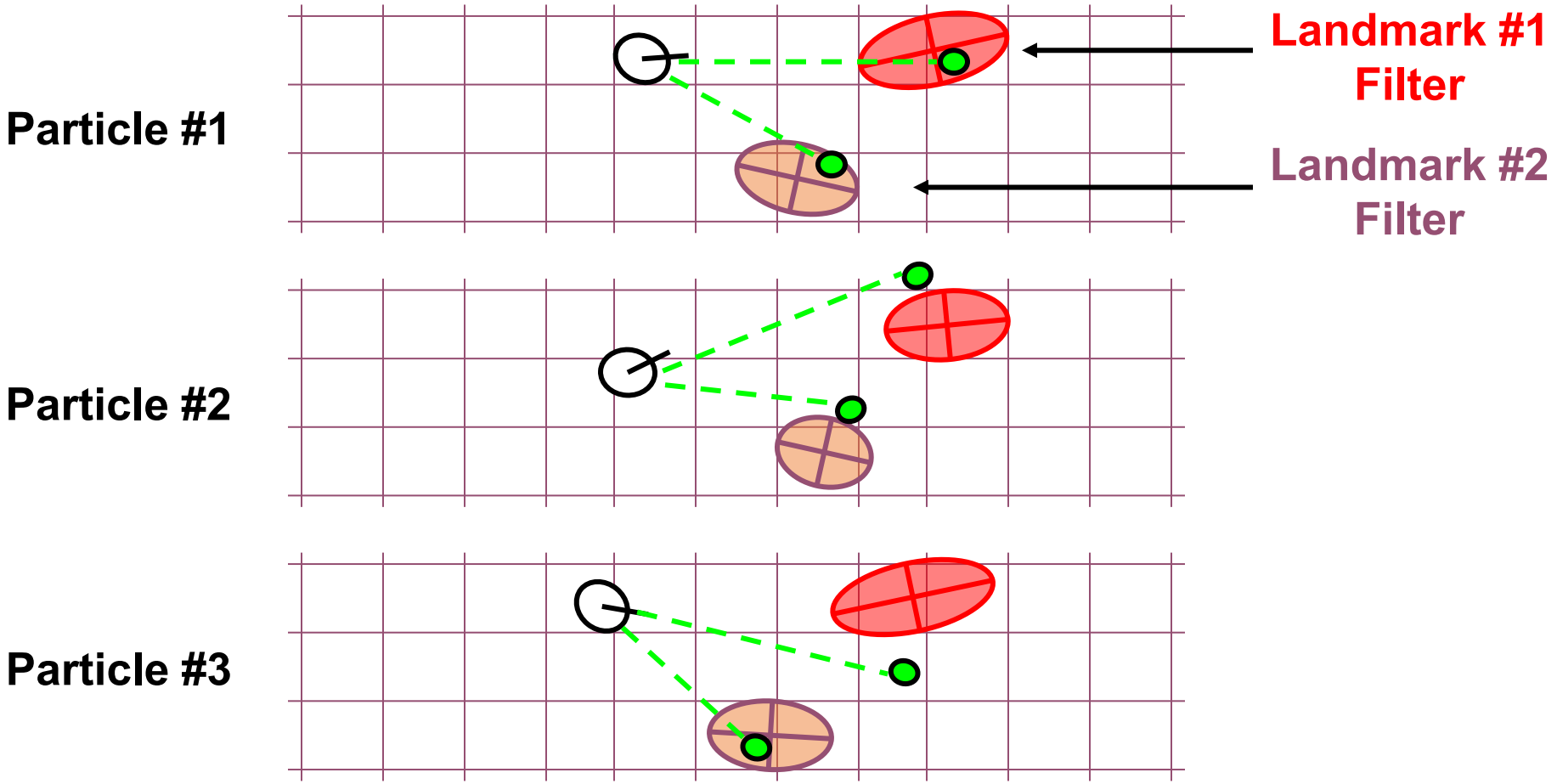


FastSLAM – Prediction (Motion model)



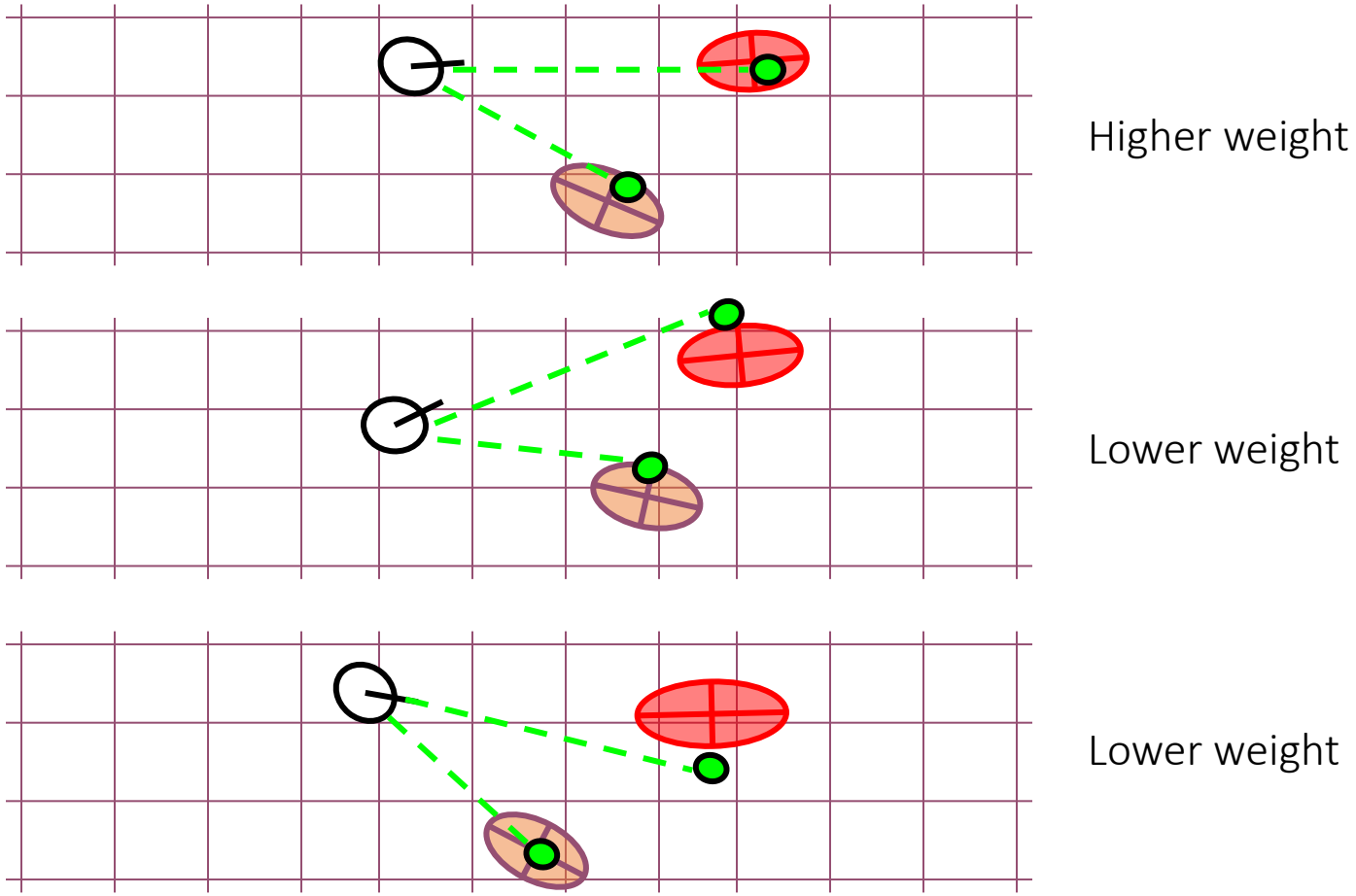
Prediction: sample a new pose for each particle. Landmark estimates stay fixed.

FastSLAM – Apply measurement model



Compare: the green observations are checked against each particle's own pose and map.

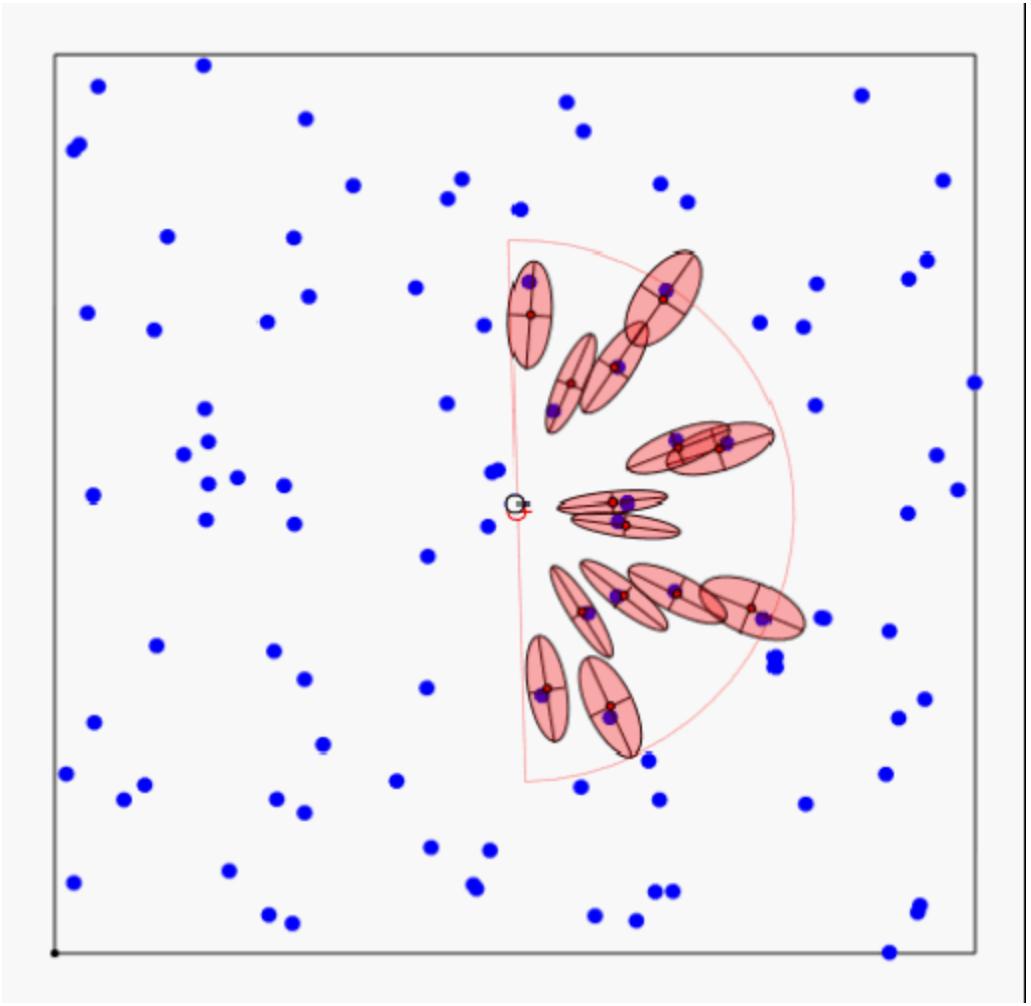
FastSLAM – landmark updates and particle weights



Update: adjust observed landmark filters; favor particles that predicted the readings well.

FastSLAM - Video

External demonstrations: open either YouTube link.



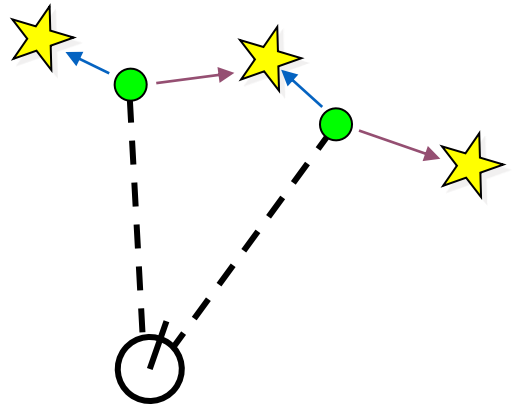
[Video 1 (YouTube)]

[Video 2 (YouTube)]

Historical AVI file is not included; the image is a still preview.

Data Association Problem

- Which observation belongs to which landmark?



Several landmark matches may be plausible.

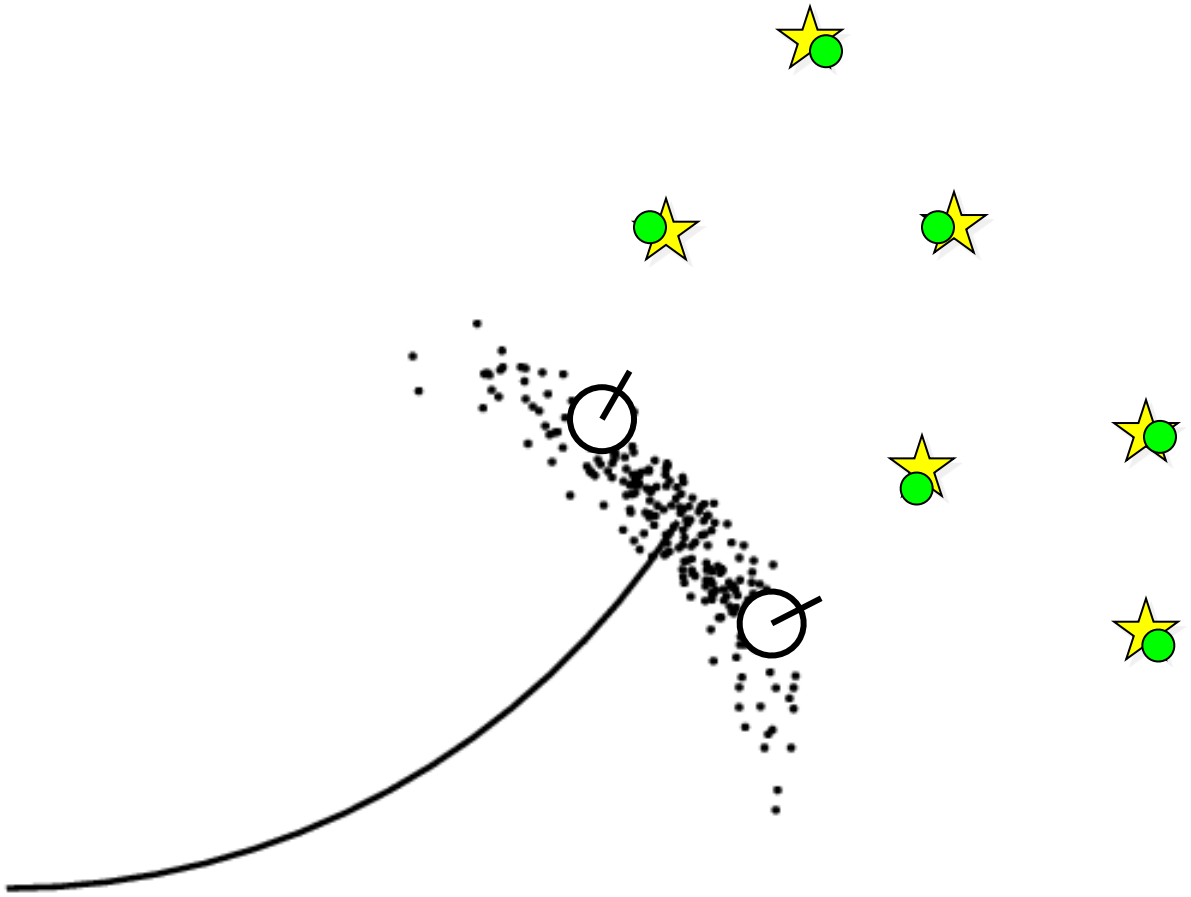
The plausible match depends on the **robot pose** and the **map uncertainty**.

Multi-Hypothesis Data Association

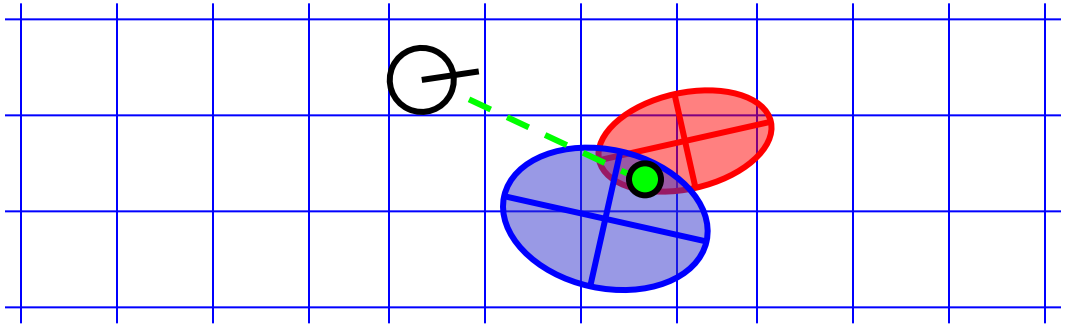
Associate each observation using **that particle's pose and map**.

Different particles can retain **different landmark matches**.

Pose uncertainty remains across the particle set.



Per-Particle Data Association



Was the observation generated by the red or the purple landmark?

P(observation|red) = 0.3 P(observation|purple) = 0.7

- Two options for per-particle data association
  - Pick the most probable match
  - Pick a random association weighted by the observation likelihoods
- If the probability is too low, generate a new landmark

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•log(M)) **Log time per particle**
Resample particle set	O(N•log(M)) **Log time per particle**
	O(N•log(M)) **Log time per particle**

N = Number of particles
M = Number of map features

See https://robots.stanford.edu/papers/Thrun03g.pdf for tricks of log time

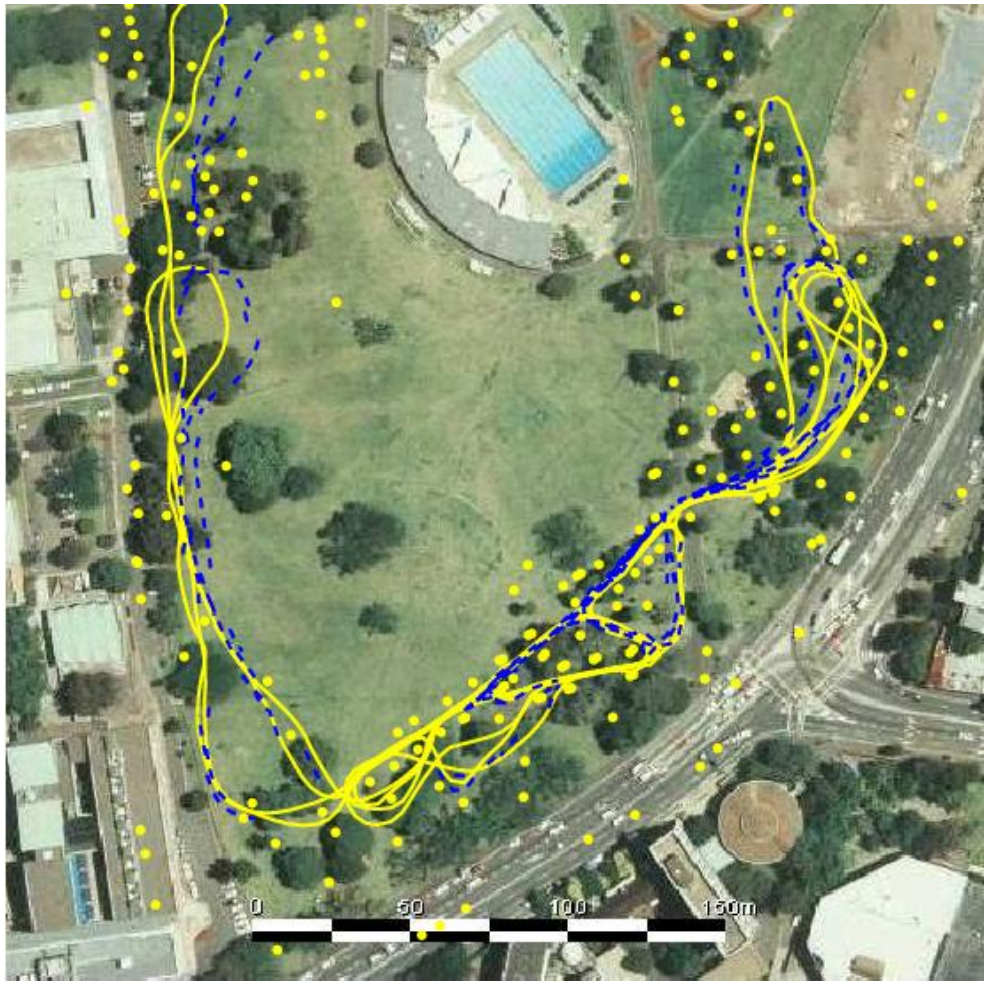
Results – Victoria Park

Reported demonstration:

4 km traverse

RMS position error just over 4 m

100 particles



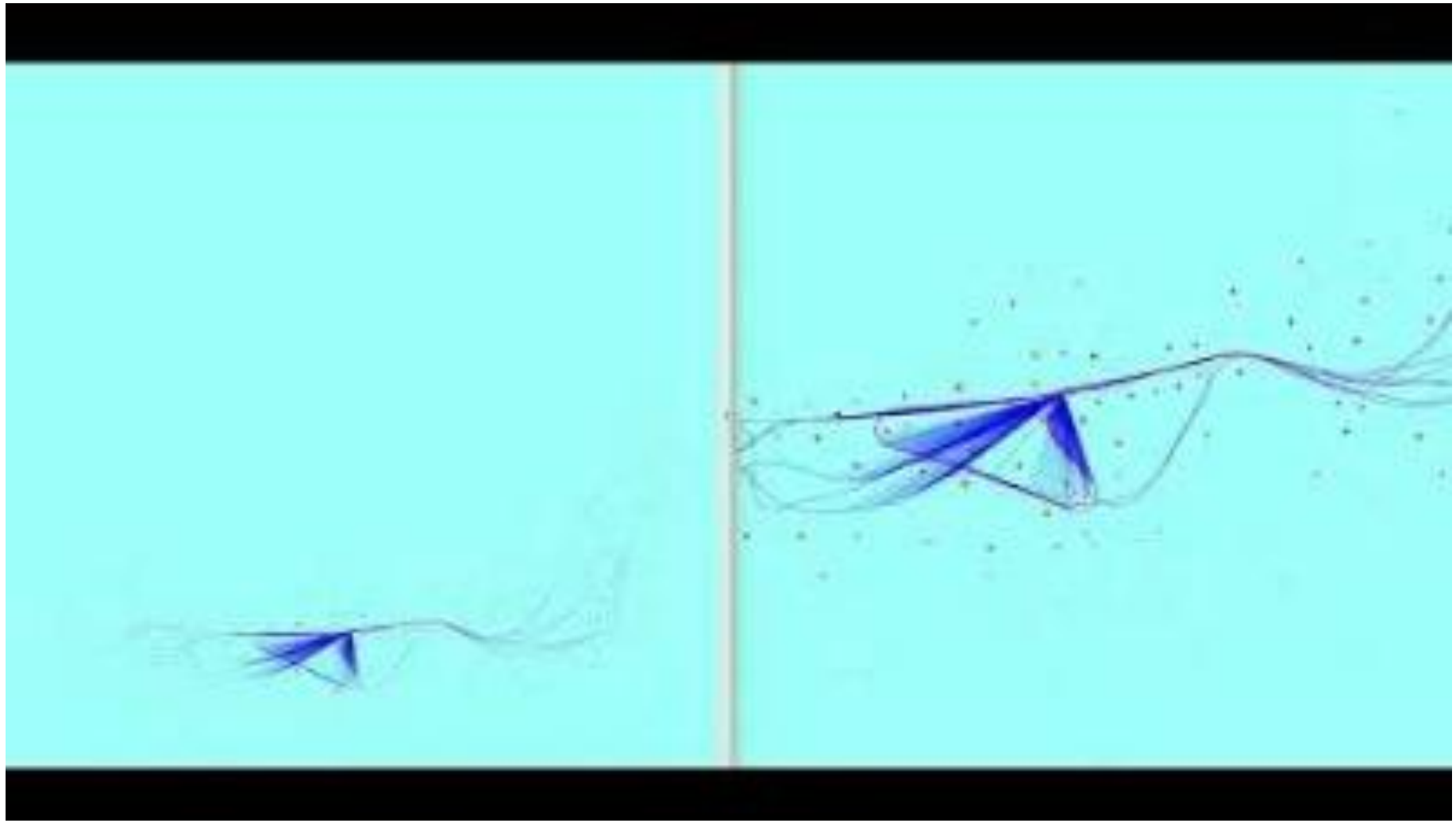
Dataset courtesy of University of Sydney

Blue = GPS reference

Yellow = FastSLAM path + landmarks

Thrun et al., FastSLAM report, §9.1; dataset: University of Sydney.

Results – Victoria Park



https://www.youtube.com/watch?v=BIOJSNHYSbc

Dataset courtesy of University of Sydney

Conclusions FastSLAM

Maintain set of particles

- Each particle contains s sampled robot path and a map
- Each feature in the map represented by local gaussian
- Result is linear in size of map and number of particles

Trick is to represent map as a set of separate Gaussians instead of a giant joint distribution

- Possible because of conditional independence given a path
- **Rao-Blackwellization**

Update rule similar to conventional particle filter

Each particle can be based on a different data association