ECE 484: Principles of Safe Autonomy (Fall 2025) Lecture 14: SLAM (simultaneous localization and mapping)

Professor: Huan Zhang

https://publish.illinois.edu/safe-autonomy/

https://huan-zhang.com

huanz@illinois.edu

Slides adapted from Prof. Sayan Mitra's slides for Spring 2025;

Reference: Probabilistic Robotics by Sebastian Thrun, Wolfram Burgard, and Dieter Fox

Slides: From the book's website

Review: state estimation module

Problem. Estimate the current state x_t of the system from knowledge about past observations $z_{0:t}$, control inputs $u_{0:t}$, and map m

- Discrete Bayes Filter/Grid localization
- Particle filter
- Kalman filter

Review: Discrete Bayes Filter

Notation: $bel(X_t = x_k) \coloneqq p_{k,t}$

Finitely many states $x_i, x_k, etc.$ Random state vector X_t

$p_{k,t}$: belief at time t for state x_k; discrete probability distribution

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

for all k do:

$$\bar{p}_{k,t} = \sum_i p(X_t = x_k | u_t, X_{t-1} = x_i)\, p_{i,t-1}$$

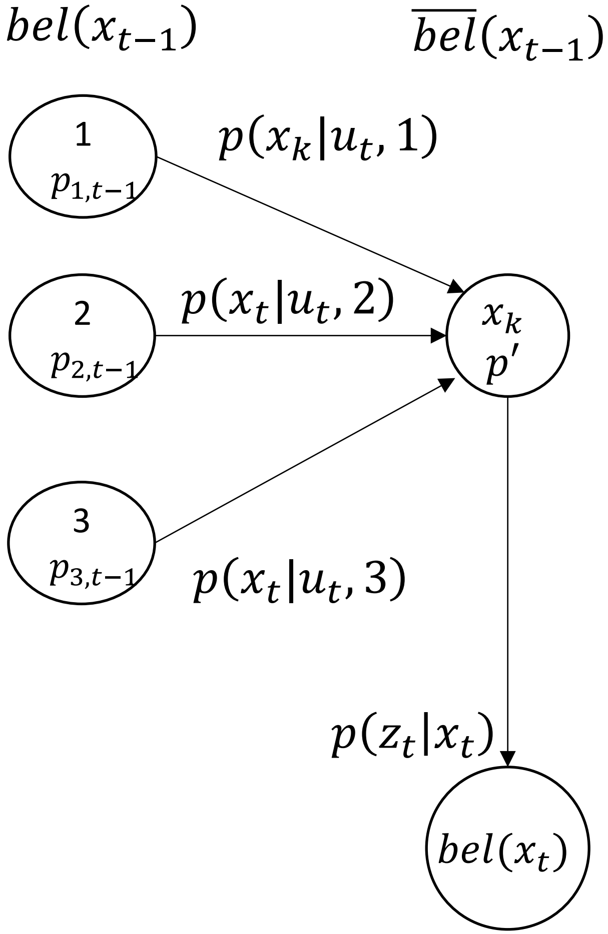
Prediction step with motion model

$$p_{k,t} = \eta\, p(z_t \,| X_t = x_k) \bar{p}_{k,t}$$

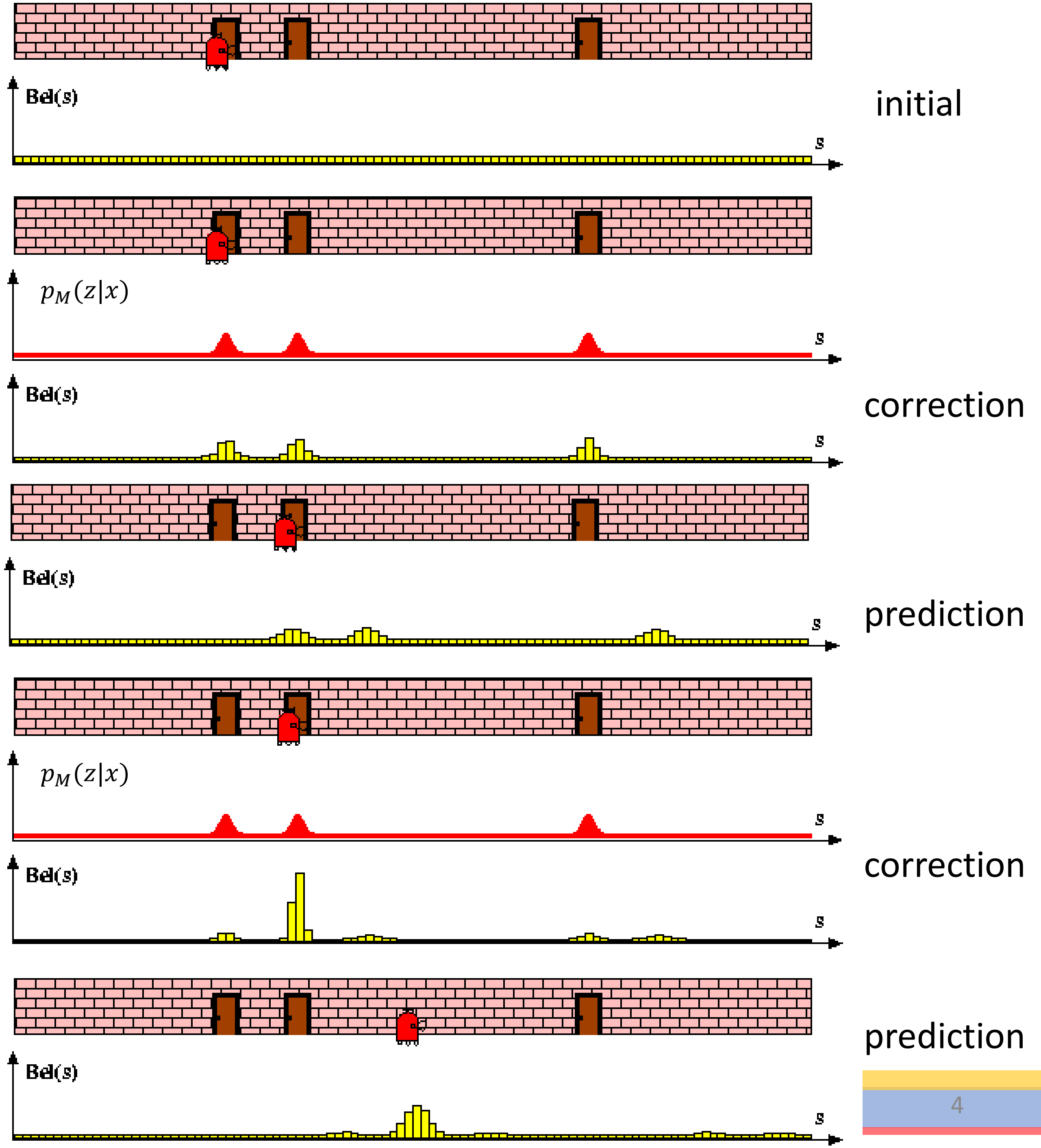
correction step with measurement model

end for

return $\{p_{k,t}\}$



Grid localization, $bel(x_t)$represented by a histogram over grid



Review: Particle filtering algorithm

$X_t := \{x_t^{[1]}, x_t^{[2]}, \dots x_t^{[M]}\}$ set of particles

Algorithm Particle_filter(X_{t-1}, u_t, z_t):
$\bar{X}_t = X_t = \emptyset$
for all m in [M] do:
  sample $x_t^{[m]} \sim p_D(x_t | u_t, x_{t-1}^{[m]})$
  $w_t^{[m]} = p_M\left(z_t \middle| x_t^{[m]}\right)$
  Add $\langle x_t^{[m]}, w_t^{[m]} \rangle$ to $\bar{X}_t$
for all m in [M] do:
  draw i $with\ probability \propto w_t^{[i]}$
  add $x_t^{[i]}$ $to\ X_t$
return X_t

ideally, $x_t^{[m]}$ is selected with probability prop. to $p(x_t \mid z_{1:t}, u_{1:t})$

$\bar{X}_t$ is the temporary particle set

sampling new particles using motion model p_D

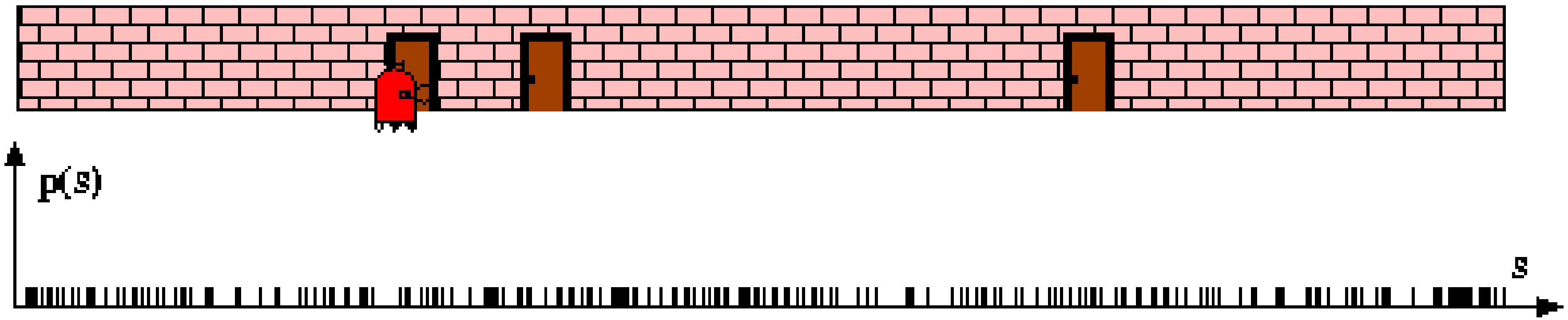
calculates *importance factor* w_t or weight according to measurement p_M

before resampling particles in $\bar{X}_t$ distributed ~ $\overline{bel}(x_t)$

after resampling particles X_t distributed ~ $bel(x_t) = \eta\, p\left(z_t \middle| x_t^{[m]}\right) \overline{bel}(x_t)$

survival of fittest: moves/adds particles to parts of the state space with higher probability, lower probability particles are eliminated

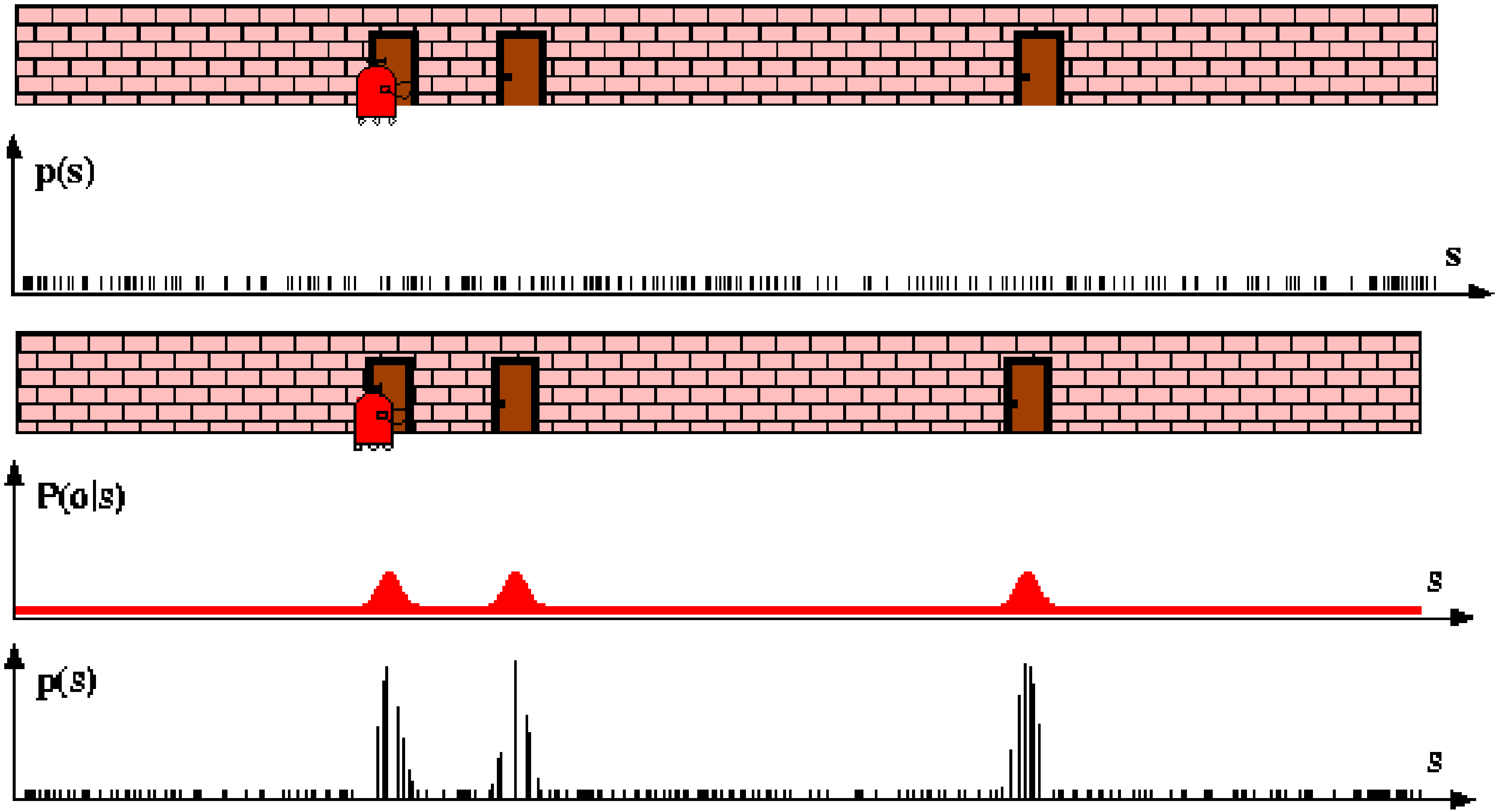
Particle Filters



Sensor Information: Importance Sampling

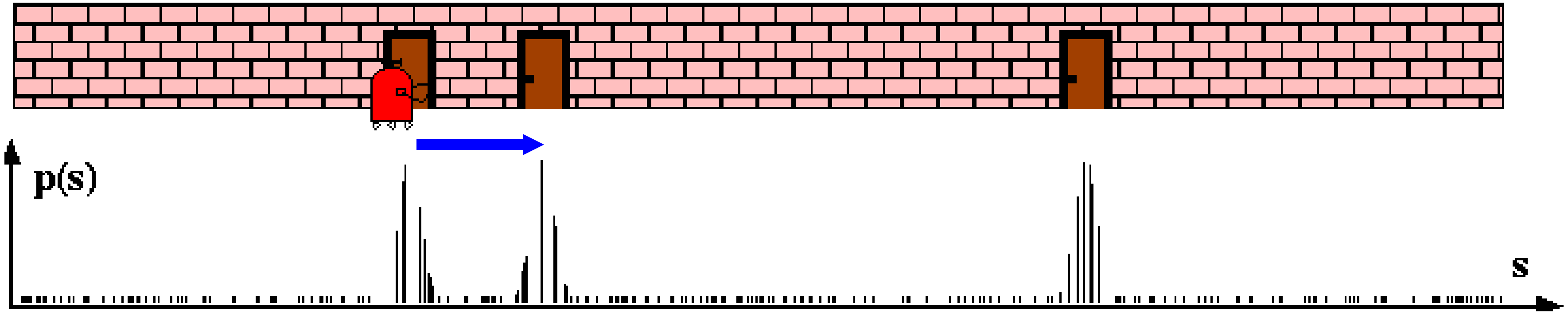
$$Bel(x) \leftarrow \alpha \, p(z \mid x) \, Bel^{-}(x)$$

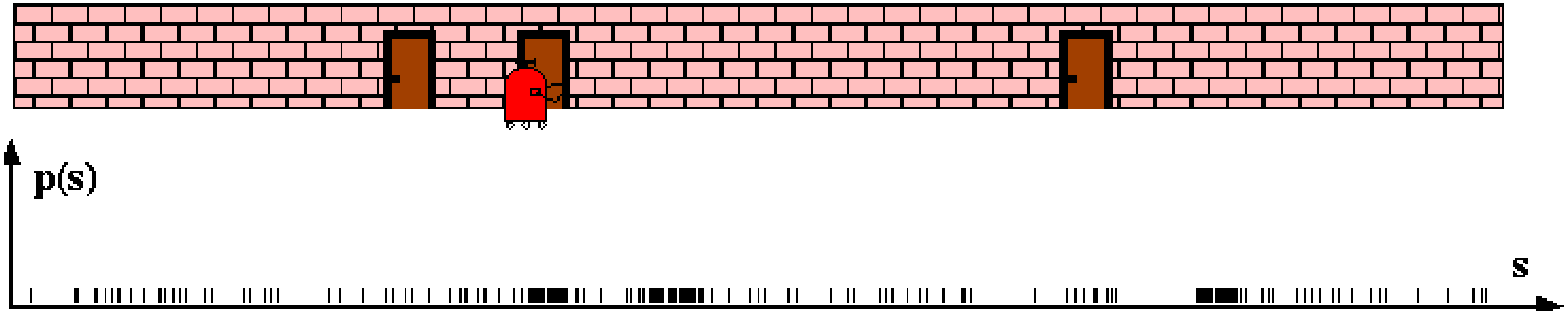
$$w \leftarrow \frac{\alpha \, p(z \mid x) \, Bel^{-}(x)}{Bel^{-}(x)} = \alpha \, p(z \mid x)$$



Robot Motion

$$\overline{Bel}(x) \leftarrow \int p(x \mid u, x')\, Bel(x')\, \mathrm{d}x'$$





Review: Discrete Kalman Filter

The Kalman filter estimates state of a Discrete Linear System with Gaussian noise
Note that we **no longer have discrete states or measurements**! No grids, particles, etc.

Intuitive insight: Assume the belief is represented as **Gaussian** distributions. **Linear transformation** (addition, scaling) and **multiplications** over Gaussians are still Gaussians

$$x_t = A_t x_{t-1} + B_t u_t + \epsilon_t$$
$$z_t = C_t x_t + \delta_t$$

x_t: State vector
u_t: Input vector
z_t: Output vector
$\varepsilon_t \sim N(0, \mathrm{Q}_t)$: Process noise with covariance Q_t
$\delta_t \sim N(0, R_t)$: Measurement noise with covariance R_t
$p(x_t | x_{t-1}, u_t) = N(A_t x_{t-1} + B_t u_t, \mathrm{Q}_t)$
$p(z_t | x_t) = N(C_t x_t, R_t)$

Review: Kalman Filter Algorithm

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

Prediction: get $\bar{\mu}_t$ and $\bar{\Sigma}_t$ (linear motion)

1. $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$
2. $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^{\top} + Q_t$

Correction: correct $\bar{\mu}_t$ and $\bar{\Sigma}_t$ (linear meas.)

1. $K_t = \bar{\Sigma}_t C_t^{\top} (C_t \bar{\Sigma}_t C_t^{\top} + R_t)^{-1}$
2. $\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$
3. $\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$

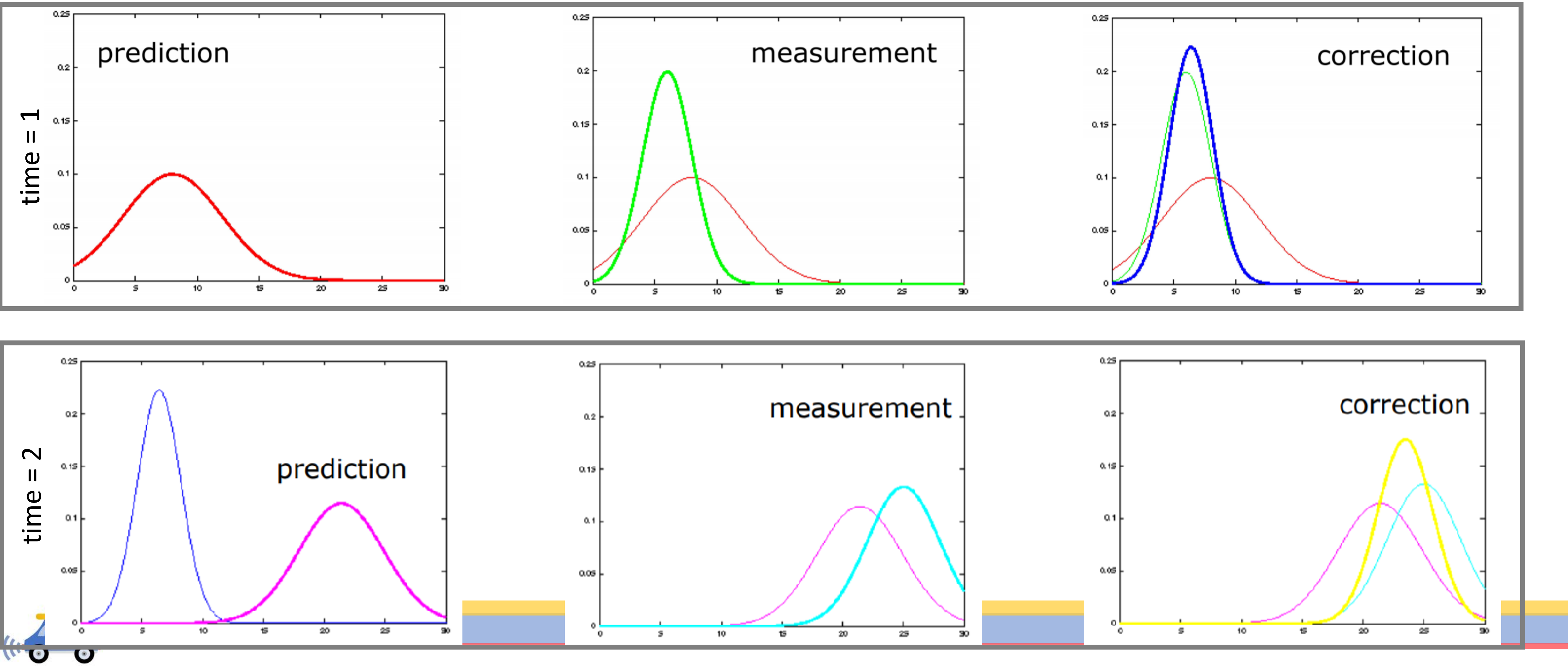
Return μ_t, Σ_t

Given $\mathrm{bel}(x_{t-1}) \sim N(\mu_{t-1}, \Sigma_{t-1})$
Apply motion model to find $\bar{x}_t$:
Linear transformation of Gaussian $\mathrm{bel}(x_{t-1})$
where $x_t = A_t x_{t-1} + B_t u_t + \epsilon_t;\ \varepsilon_t \sim N(0, Q_t)$
=> $\bar{x}_t \sim N(\bar{\mu}_t, \bar{\Sigma}_t)$

Given $\bar{x}_t \sim N(\bar{\mu}_t, \bar{\Sigma}_t)$
Apply measurement model to find $\mathrm{bel}(x_t)$:
Product of Gaussians $\bar{x}_t$ and $\mathrm{p}(z_t | x_t)$
Where $p(z_t | x_t)$ is a Gaussian (variable is x_t)
=> $\mathrm{bel}(x_t) \sim N(\mu_t, \Sigma_t)$

Kalman Filter Example

Demo: https://colab.research.google.com/drive/1qcINZgx8ebwWtRQROh3z8cpvtmuE4Dt0?usp=sharing



Summary

- Grid localization
  - Can represent arbitrary, multi-modal distributions; minimal assumption on dynamics and sensor models
  - High computational cost; impractical for high dimension; inaccurate if grid is coarse
- Particle Filters (Monte Carlo Localization)
  - Model arbitrary distirbutions via samples; more scalable to higher-dimensional spaces than grid localization; easy to implement and understand
  - Approximation quality and time complexity depends on the number of particles; no guarantee on approximation error, particle degeneracy problem
- Kalman filters
  - Extremely efficient computationally; optimality guarantees; close form & well studied
  - Gaussian distributions only (no multi-modal); known linear motion and measurement models
  - Extension to nonlinear system possible (extended Kalman filters), yet EKF has its own limitations

The SLAM Problem

- SLAM: simultaneous localization and mapping
- The task of **building a map** while estimating the pose of the robot relative to this map
- Robot does not have a map, unlike in localization

- Why is SLAM hard?
  Chicken and egg problem:
  a map is needed to localize the robot and
  a pose estimate is needed to build a map

A SLAM Solution

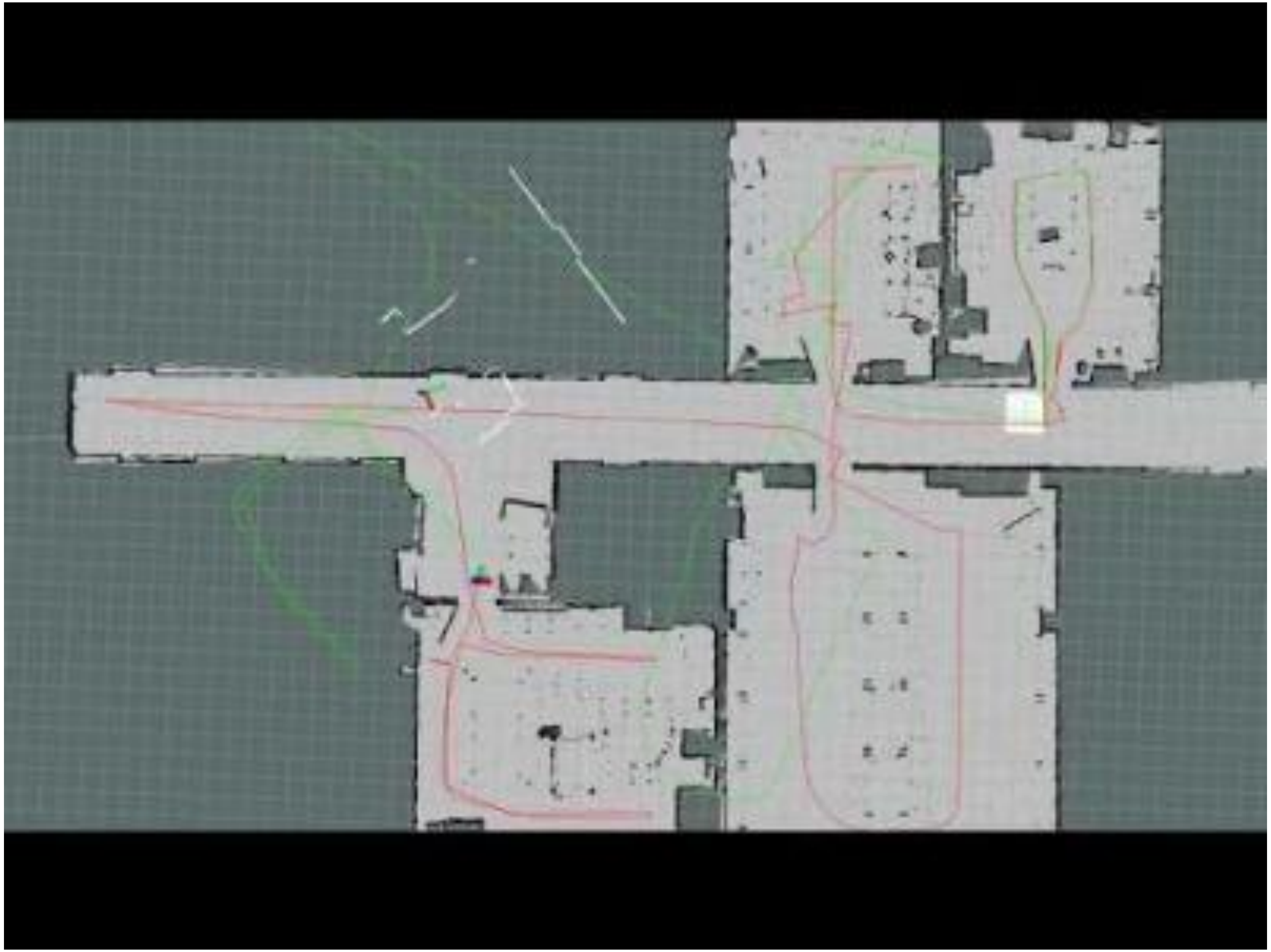
A robot moving though an unknown, static environment

Given:

- The robot's controls
- Observations of nearby features

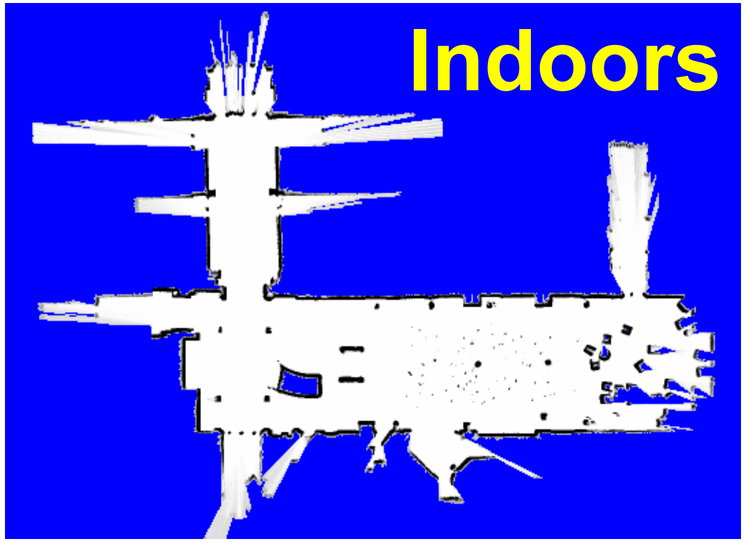
Estimate:

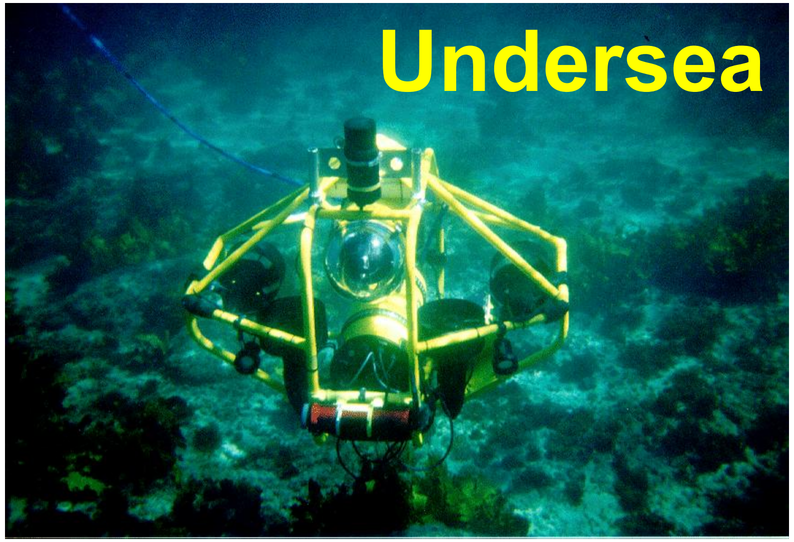
- Map of features
- Path of the robot

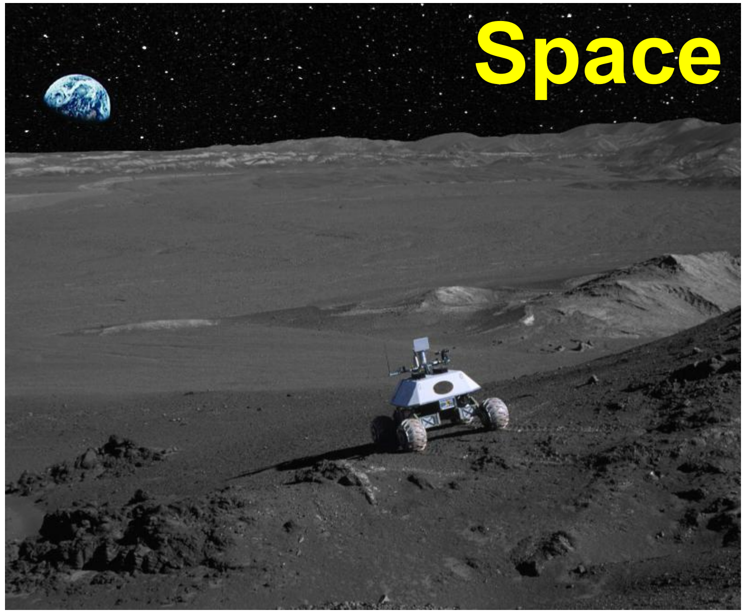


Video from Miklós Tóth https://www.youtube.com/watch?v=v4flz0AtENk

SLAM Applications



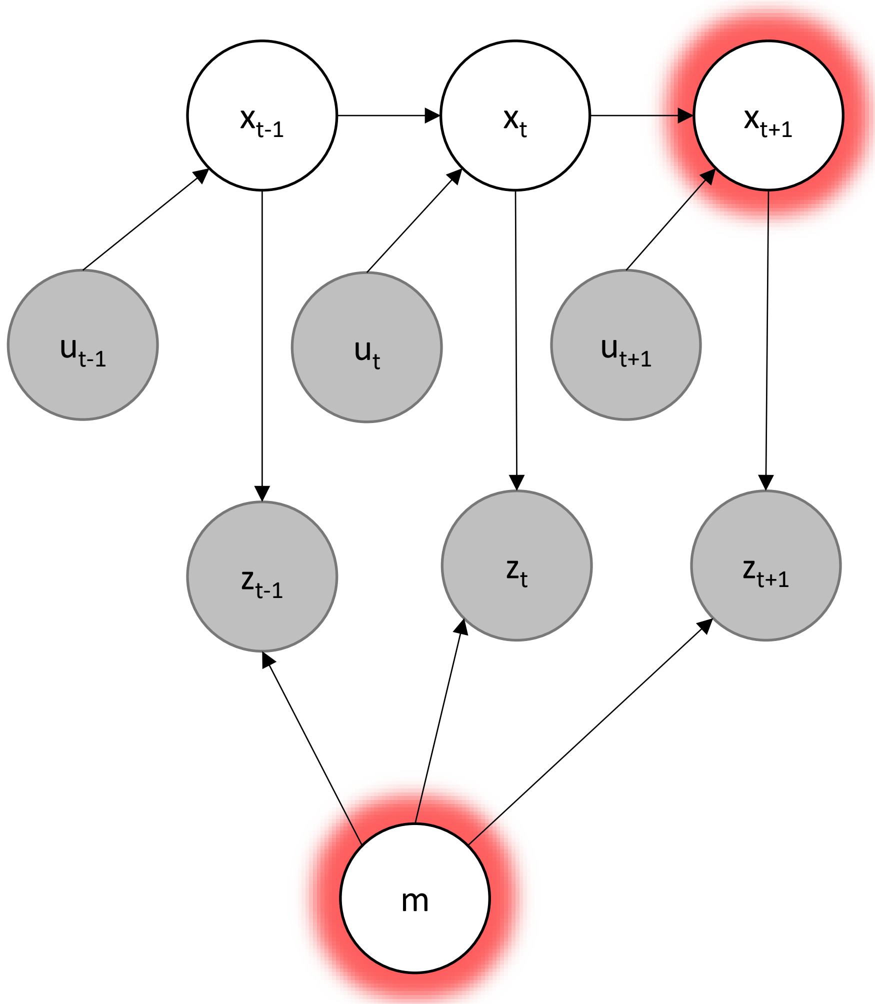




Forms of SLAM

- State / history
  - Online SLAM: $p(x_t, m \,|z_{1:t}, u_{1:t})$
  - Full SLAM: $p(x_{1:t}, m \,|z_{1:t}, u_{1:t})$
- Continuous or discrete *correspondence variables*
  - $p(x_t, m|z_{1:t}, u_{1:t})$
- Many algorithms: EKFSLAM, GraphSLAM, FastSLAM

Online SLAM

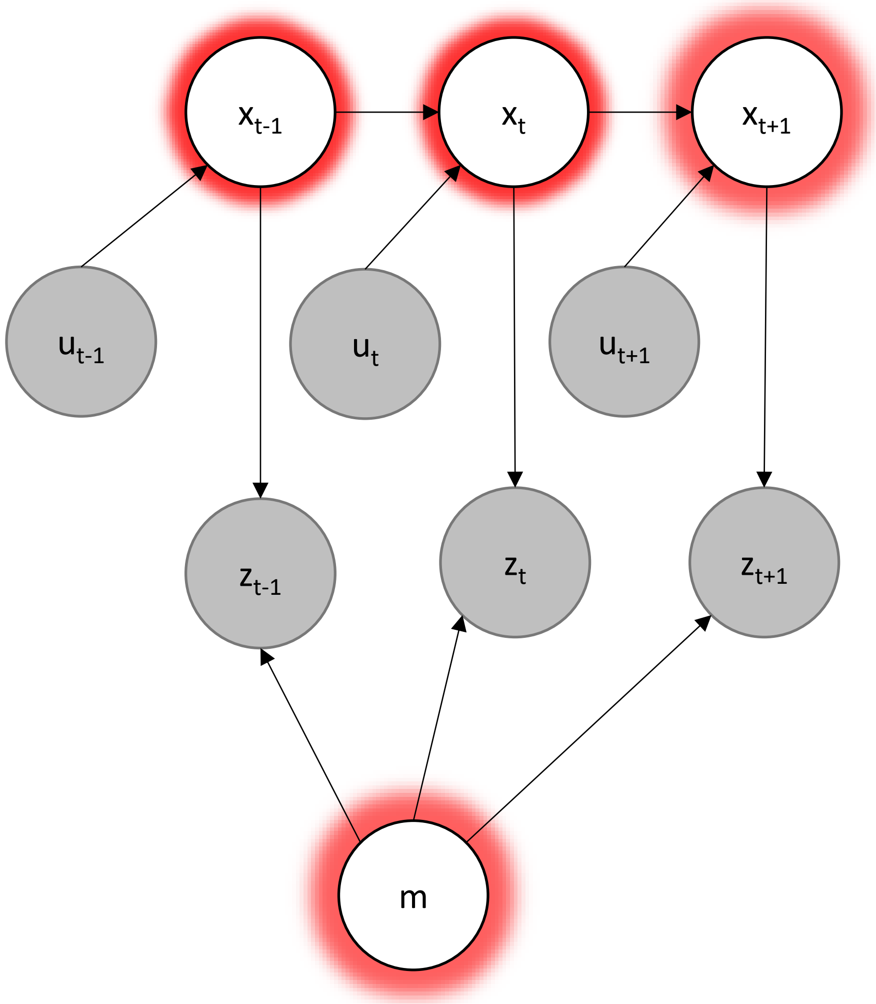


Shaded known:
control inputs (u),
measurements(z).

White nodes to be determined (x,m)

want to calculate
$p(x_t, m | z_{1:t}, u_{1:t})$

Full SLAM



Shaded known:
control inputs (u), measurements(z).

White nodes to be determined (x,m)

want to calculate
$p(x_{1:t}, m | z_{1:t}, u_{1:t})$

Continuous unknowns: $x_{1:t}, m$
Discrete unknowns: Relationship of detected objects to new objects

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

c_t: corrsnpondence variable

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:

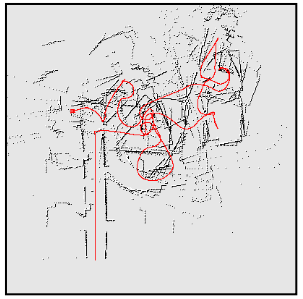
$$p(x_t, m \mid z_{1:t}, u_{1:t}) = \int \int \cdots \int p(x_{1:t}, m \mid z_{1:t}, u_{1:t}) \, dx_1 \, dx_2 \ldots dx_{t-1}$$

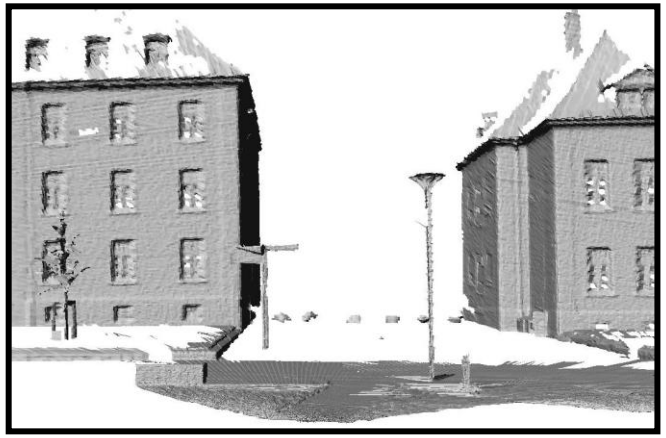
Integration of the marginals typically done one at a time

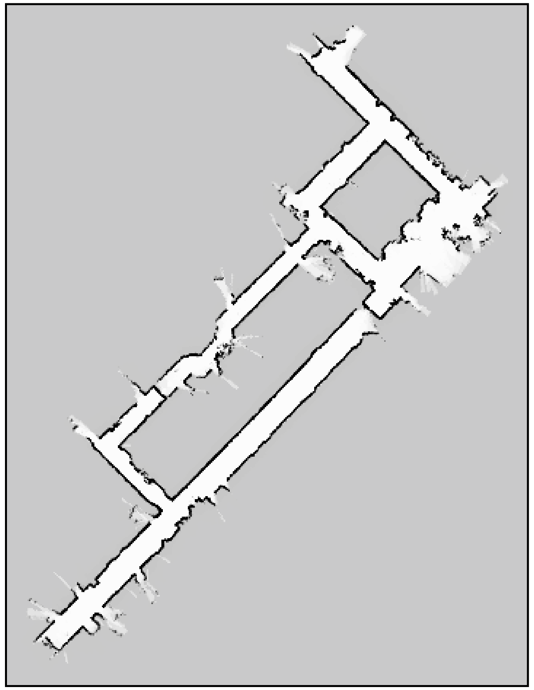
Estimates most recent pose and map!

Representations

Grid maps or scans

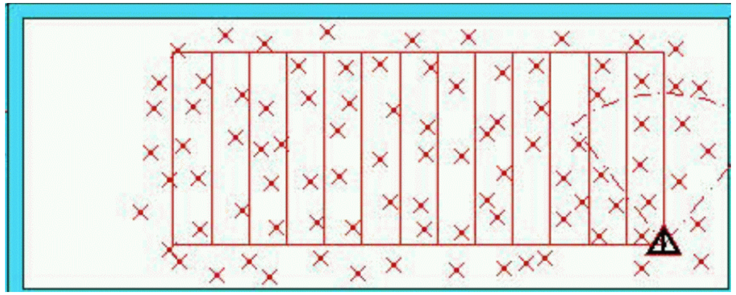


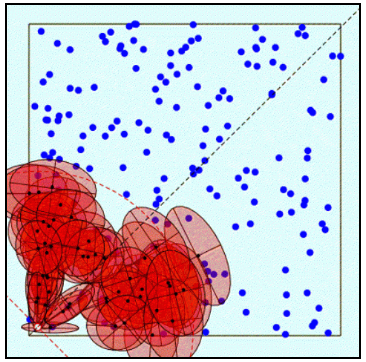


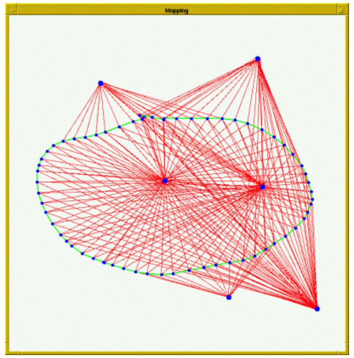


[Lu & Milios, 97; Gutmann, 98: Thrun 98; Burgard, 99; Konolige & Gutmann, 00; Thrun, 00; Arras, 99; Haehnel, 01;…]

Landmark-based



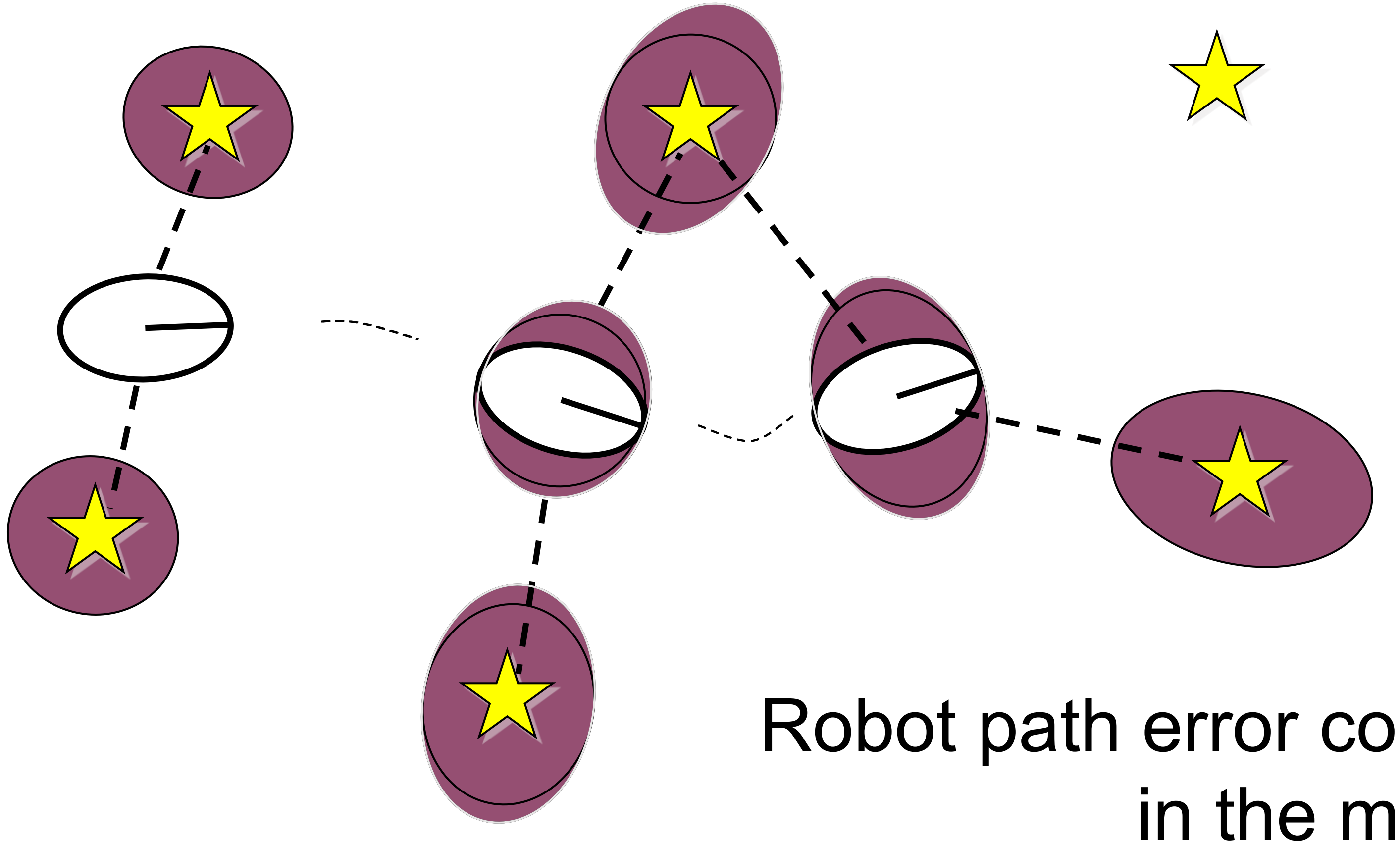




[Leonard et al., 98; Castelanos et al., 99: Dissanayake et al., 2001; Montemerlo et al., 2002;…

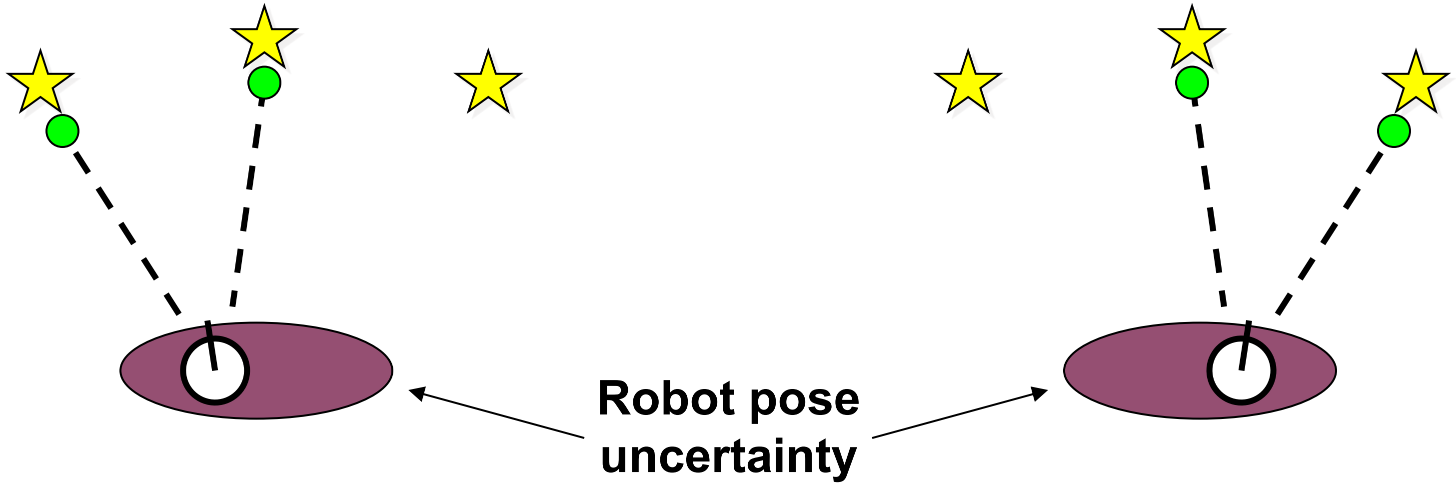
Why is SLAM a hard problem?

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



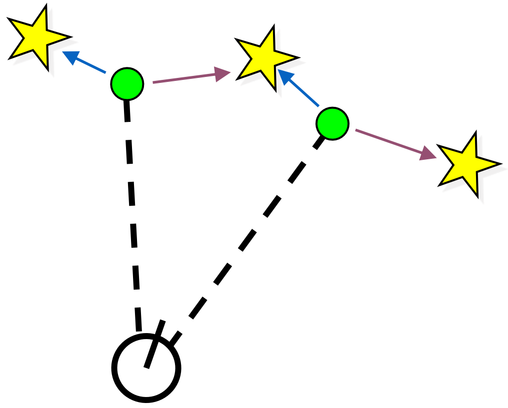
Robot path error correlates errors in the map

Why is SLAM a hard problem?



- In the real world, the mapping between observations and landmarks is unknown
- Picking wrong data associations can have catastrophic consequences
- Pose error correlates data associations

Data Association Problem



- A data association is an assignment of observations to landmarks
- In general there are more than $\binom{n}{m}$ (n observations, m landmarks) possible associations
- Also called "assignment problem"

Localization vs. SLAM

- A particle filter can be used to solve both problems
- Localization: state space $<x, y, \theta>$
- SLAM: state space $<x, y, \theta, map>$
  - for landmark maps = $<l_1, l_2, \ldots, l_m>$
  - for grid maps = $<c_{11}, c_{12}, \ldots, c_{1n}, c_{21}, \ldots, c_{nm}>$
- **Problem:** The number of particles needed to represent a posterior grows exponentially with the dimension of the state space!

$X_t = x_t^{[1]}, x_t^{[2]}, \ldots x_t^{[M]}$ particles

Particle_filter(X_{t-1}, u_t, z_t):

$\bar{X}_t = X_t = \emptyset$

for all m in [M] do:

  sample $x_t^{[m]} \sim p_D(x_t | u_t, x_{t-1}^{[m]})$

  $w_t^{[m]} = p_M\left(z_t \middle| x_t^{[m]}\right)$

  $\bar{X}_t = \bar{X}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$

for all m in [M] do:

  draw i with probability $\propto w_t^{[i]}$

  add $x_t^{[i]}$ to X_t

return X_t

- Naïve implementation of particle filters to SLAM will be crushed by the curse of dimensionality

Dependencies

- Is there a dependency between the dimensions of the state space?
- If so, can we use the dependency to solve the problem more efficiently?

- In the SLAM context
  - The map depends on the poses of the robot.
  - We know how to build a map given the position of the sensor is known.

Conditional Independence

- A and B are conditionally independent given C if

  $$P(A, B \mid C) = P(A|C)\, P(B|C)$$

- Conditional independence enables us to **factor** a high-dimensional distribution $P(A, B \mid C)$ as a product of two lower-dimensional distributions

Conditional Independence

- A and B are conditionally independent given C if

$$P(A, B \mid C) = P(A|C)\ P(B|C)$$

Example: A mobile robot estimating its position using two sensors (sonar and laser rangefinder):

x = Robot's true position

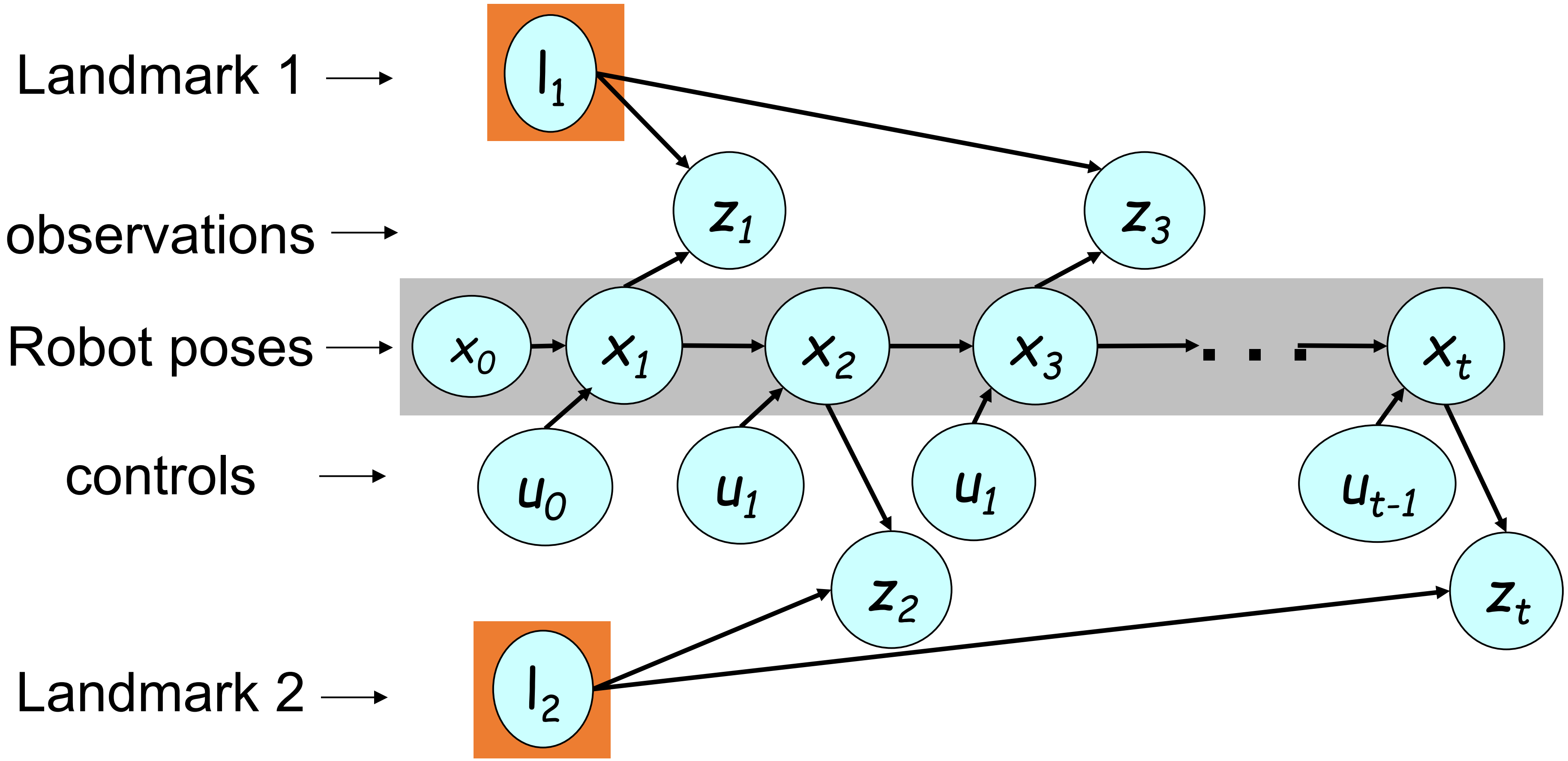
z_1 = Sonar measurement

z_2 = Laser rangefinder measurement

$$P(z_1, z_2 | x) = P(z_1 | x) \cdot P(z_2 | x)$$

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)

poses map observations & movements

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

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

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

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

Robot path posterior
(localization problem)

Conditionally
independent
landmark positions

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**
- Given that the second term can be computed efficiently, particle filtering becomes possible!

David H. Blackwell (1919-2010)

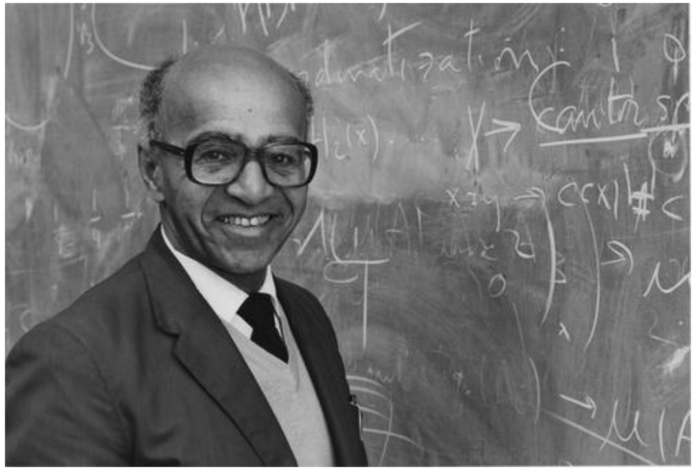


photo from stat.illinois

Independently developed dynamic programming. Several results including the Blackwell renewal theorem and the Rao-Blackwell theorem in statistics.

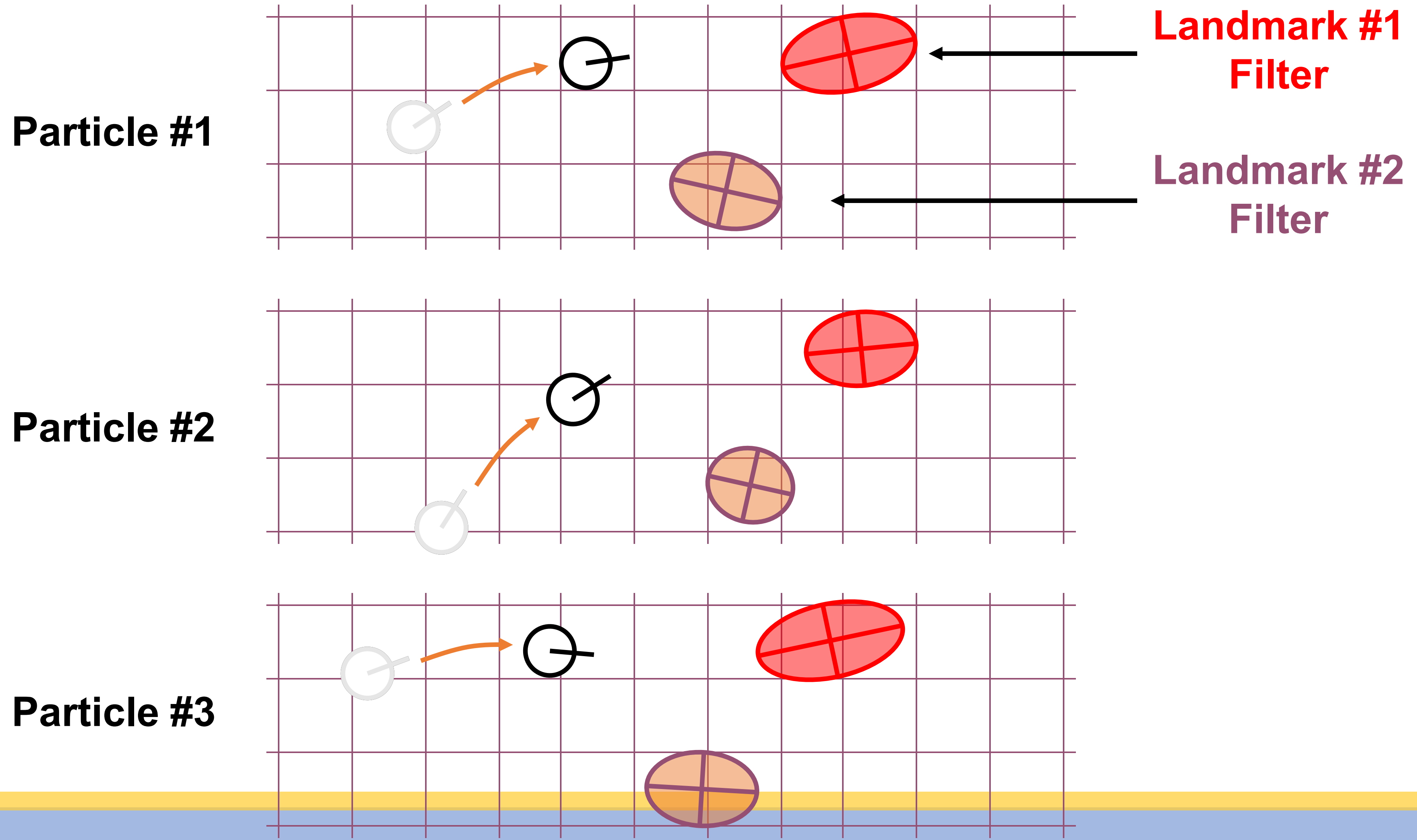
University of Illinois at Urbana-Champaign (BA, MA, PhD 1941)

FastSLAM

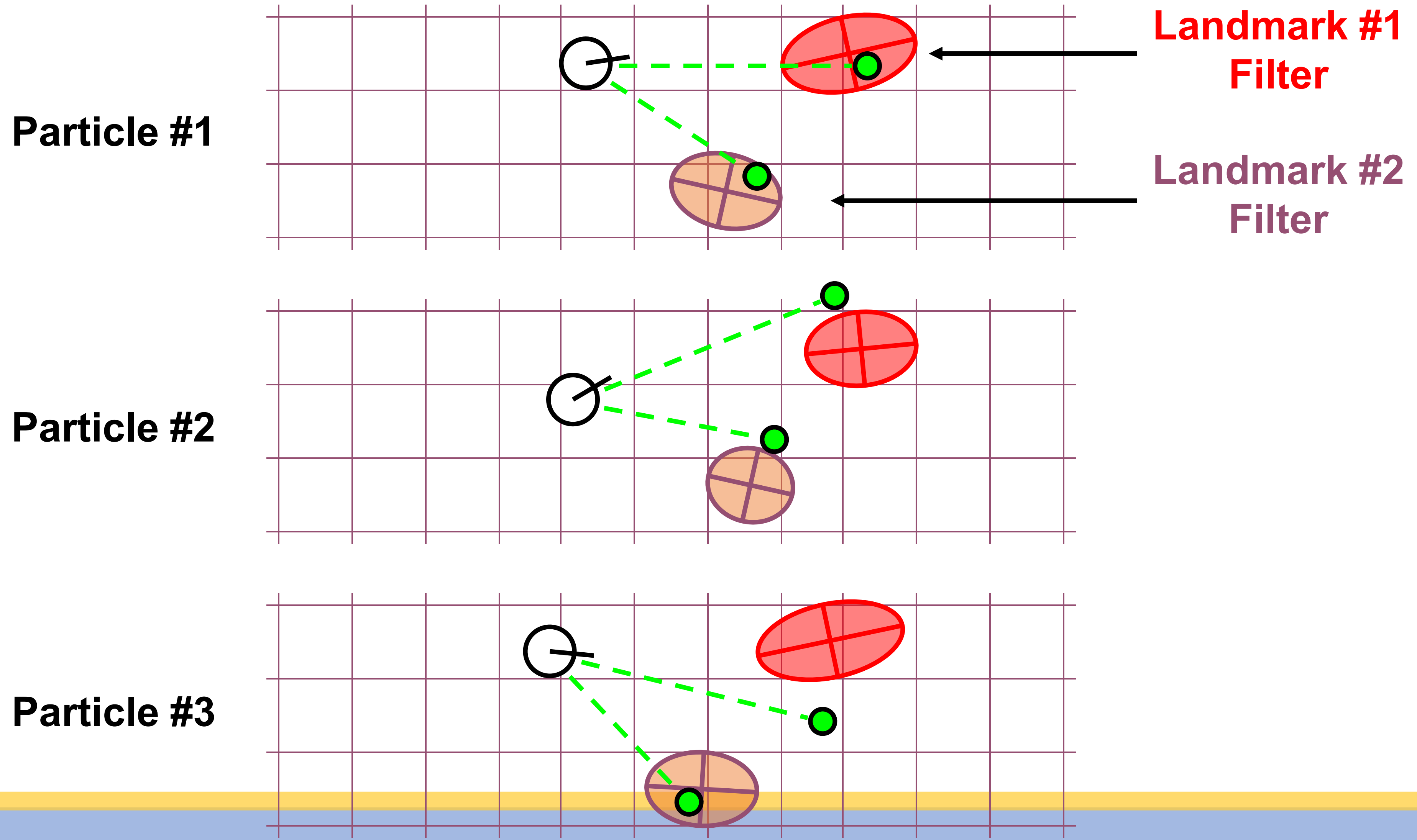
- Rao-Blackwellized particle filtering based on landmarks [Montemerlo et al., 2002]
- Each landmark is represented by a 2x2 Extended Kalman Filter (EKF)
- Each particle therefore has to maintain *M* EKFs

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**

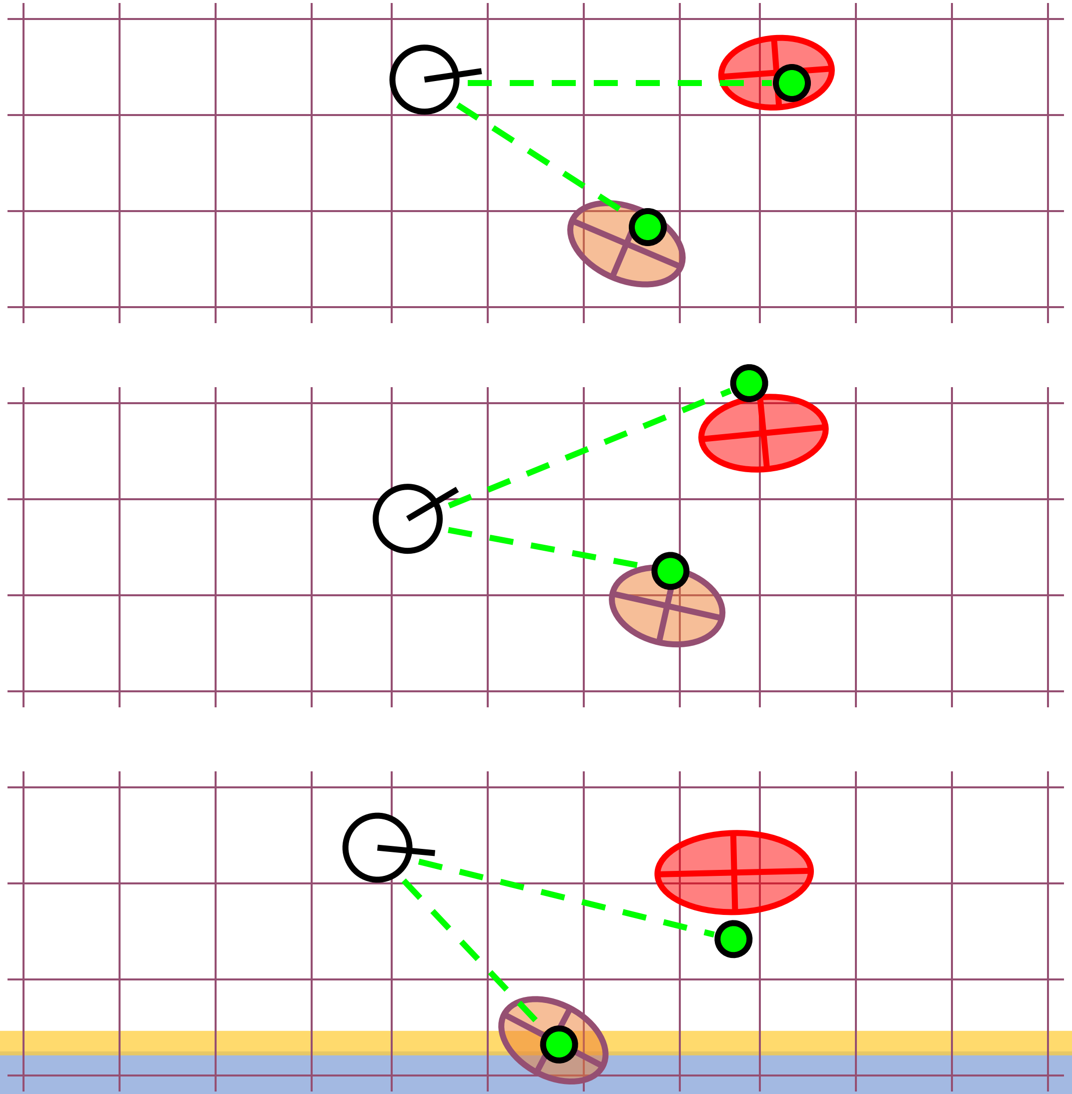
FastSLAM – Prediction (Motion model)



FastSLAM – Correction (Measurement model)



FastSLAM – Sensor Update

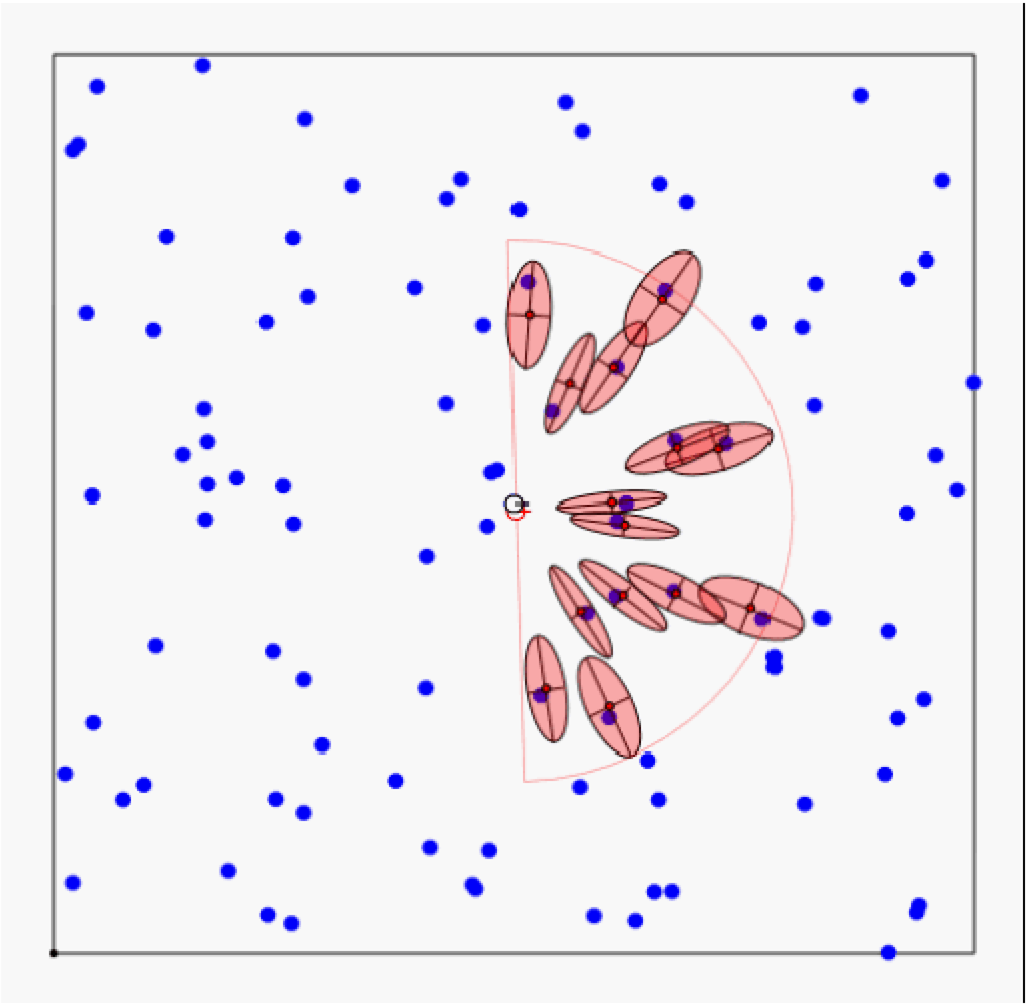


Particle #1 **Weight = 0.8**

Particle #2 **Weight = 0.4**

Particle #3 **Weight = 0.1**

FastSLAM - Video

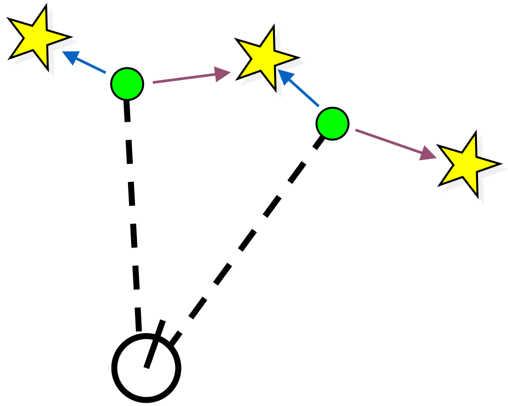


https://www.youtube.com/watch?v=6xRu7Xgmwcc

https://www.youtube.com/watch?v=ATj-DrwrHx0

Data Association Problem

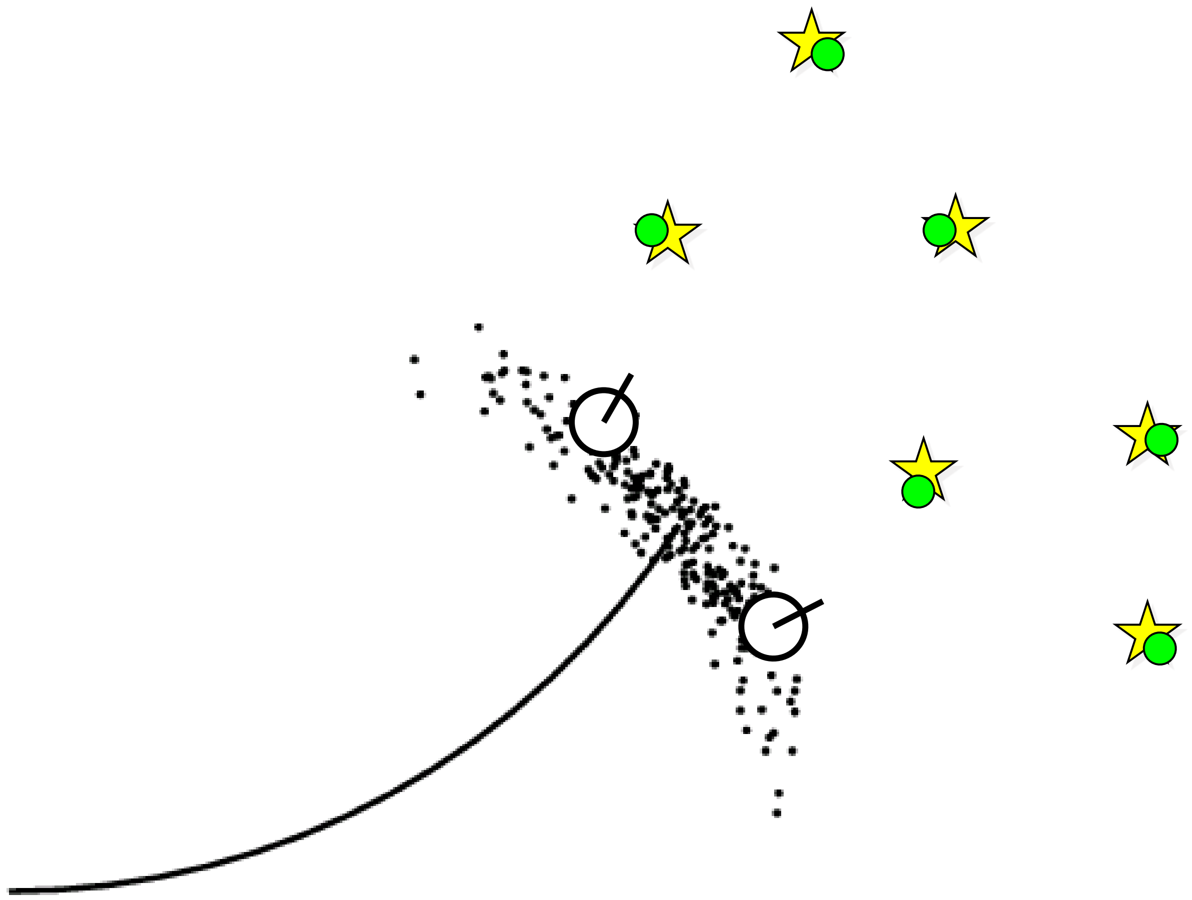
- Which observation belongs to which landmark?



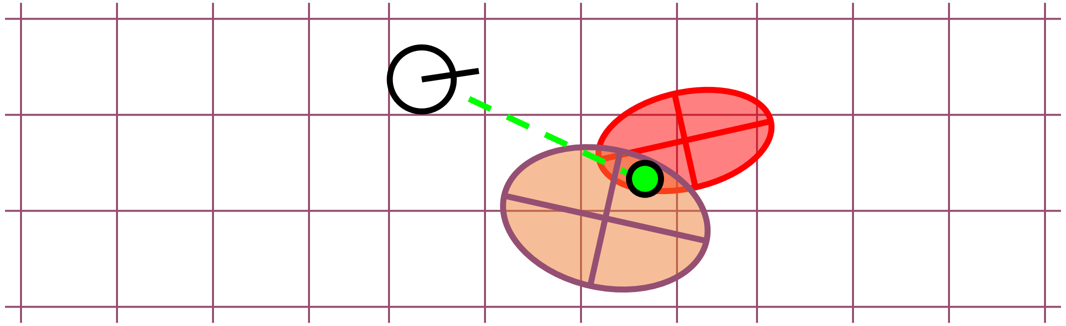
- A robust SLAM must consider possible data associations
- Potential data associations depend also on the pose of the robot

Multi-Hypothesis Data Association

- Data association is done on a per-particle basis
- Robot pose error is factored out of data association decisions



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

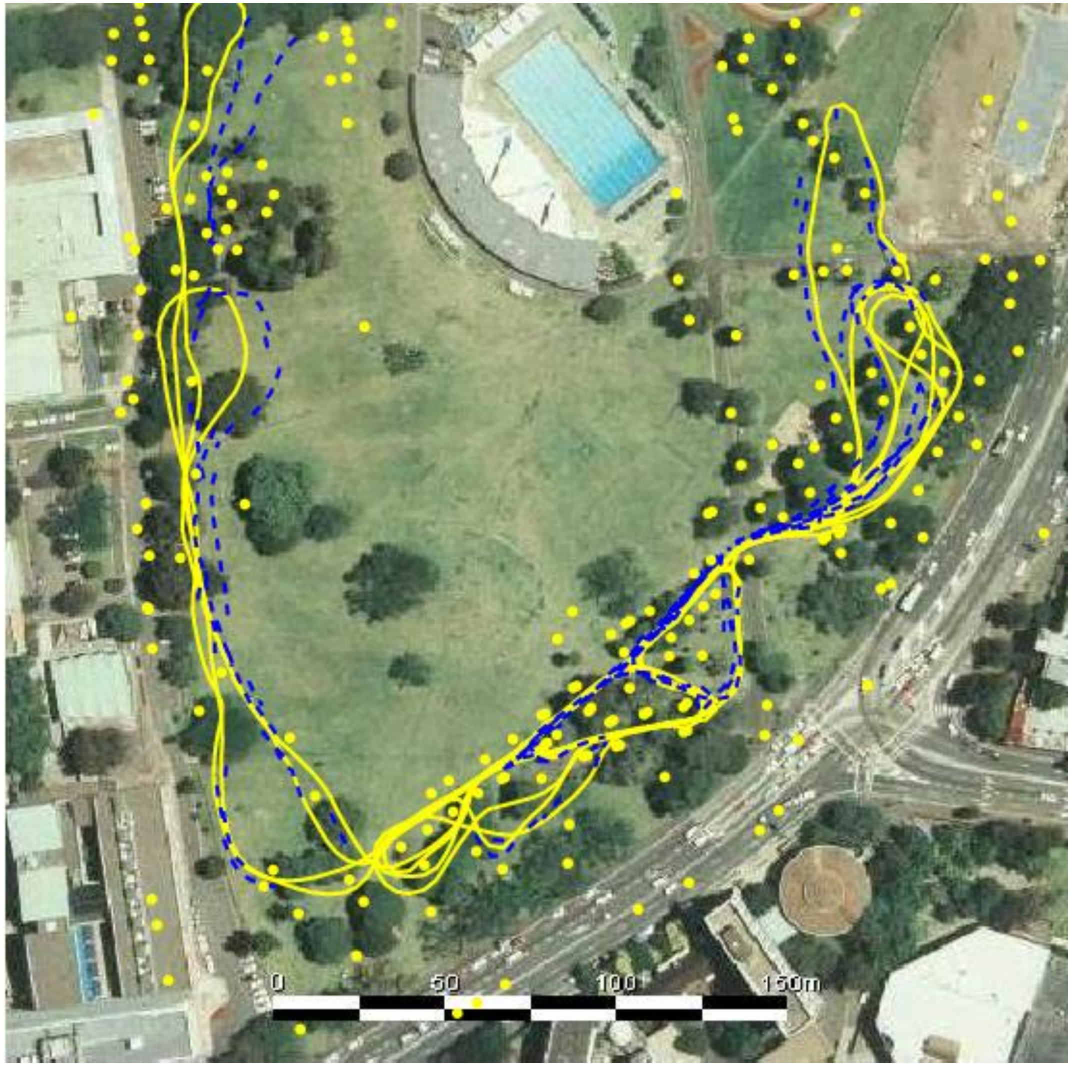
N = Number of particles
M = Number of map features

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

Results – Victoria Park

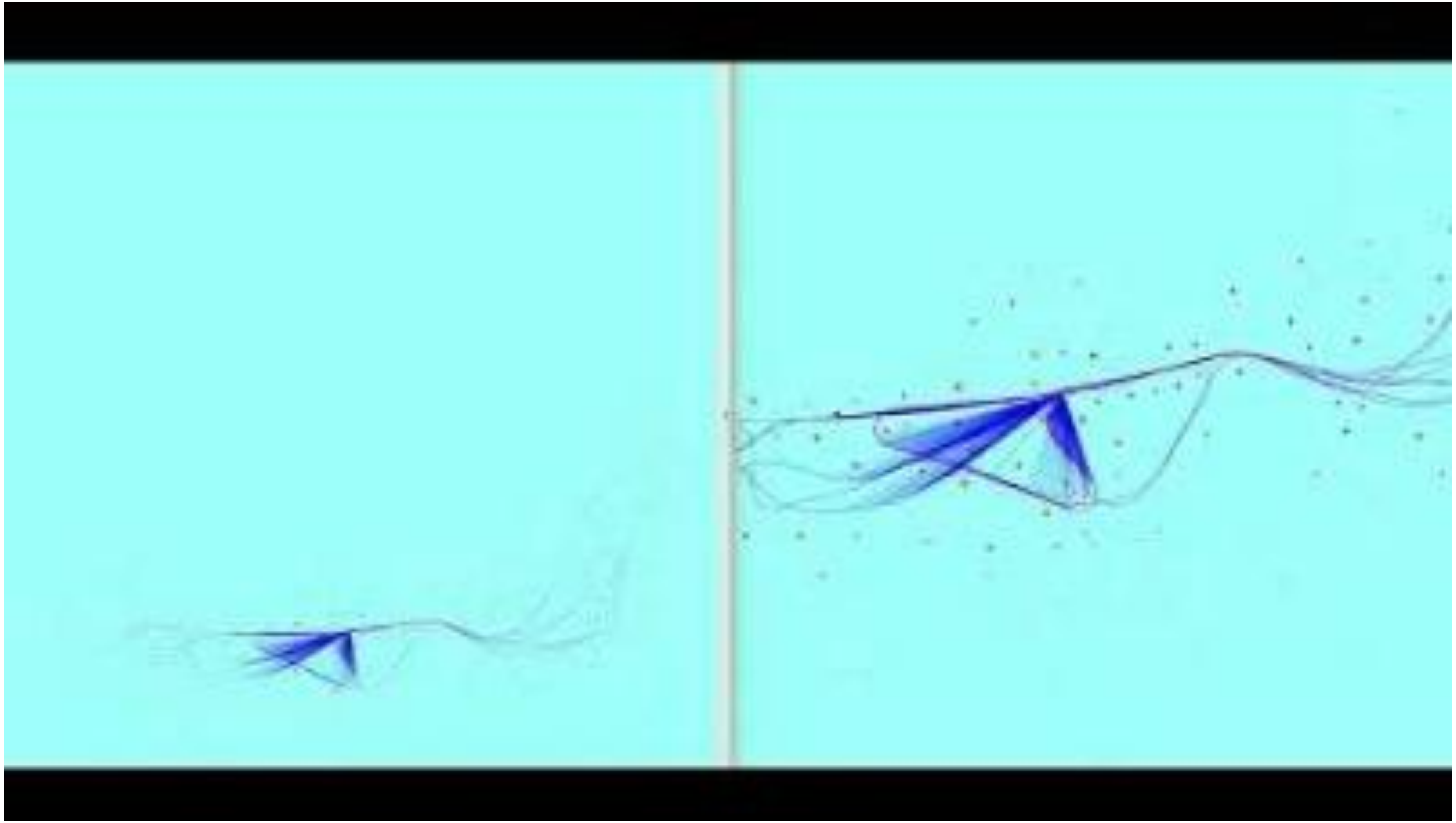
- 4 km traverse
- < 5 m RMS position error
- 100 particles

Blue = GPS
Yellow = FastSLAM



Dataset courtesy of 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