Principles of Safe Autonomy: Lecture 13: Kalman filter and SLAM

Sayan Mitra

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

Slides: From the book's website

Filtering, estimation

- Kalman filter
- Overview of SLAM

Particle filtering algorithm

```
$X_t = x_t^{[1]}, x_t^{[2]}, \dots x_t^{[M]}$ particles
Algorithm Particle_filter($X_{t-1}, u_t, z_t$):
$\bar{X}_t = X_t = \emptyset$
for all $m$ in [M] do:
        sample $x_t^{[m]} \sim p(x_t | u_t, x_{t-1}^{[m]})$
        $w_t^{[m]} = p\left(z_t \middle| x_t^{[m]}\right)$
        $\bar{X}_t = \bar{X}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
end for
for all $m$ in [M] do:
        draw $i$ $with\ probability \propto w_t^{[i]}$
        add $x_t^{[i]}$ $to\ X_t$
end for
return $X_t$
```

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

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

Uses state transition distribution

Calculates *importance factor* w_t or weight using measurement model

resampling or importance sampling according to $\eta\, p\left(z_t \middle| x_t^{[m]}\right) \overline{bel}(x_t)$

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

Nonparametric method

Heavy computation

Can handle nonlinear models

Does not rely on analytical expression for distributions

Convergence guarantees only under assumptions on #partiles→ infinity

How to estimate state if you have

- you have linear model of system and measurement
- hard limit on computation
- Need stronger convergence guarantees

~~Nonparametric method~~

~~Heavy computation~~

~~Can handle nonlinear models~~

~~Does not rely on analytical expression for distributions~~

~~Convergence guarantees only under assumptions on #partiles $\rightarrow$ infty~~

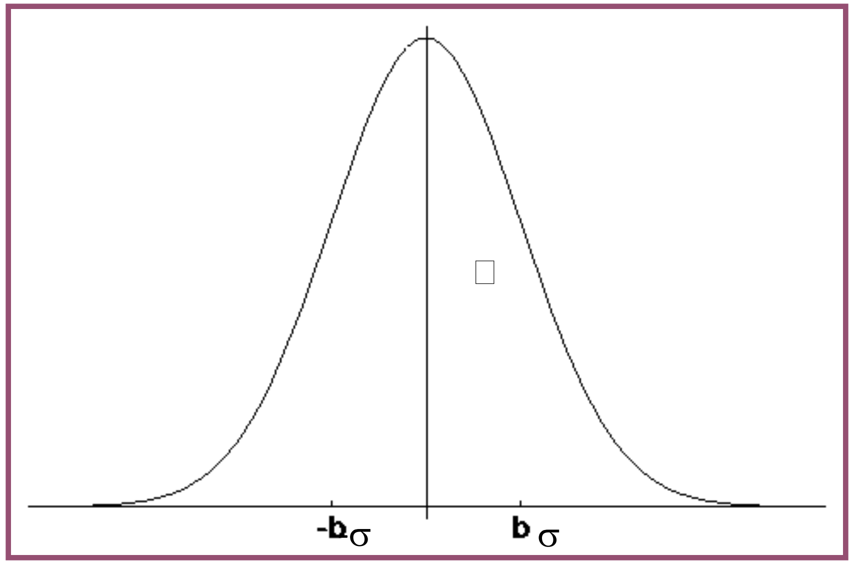
Kalman Filter: State estimation algorithm for linear systems with Gaussian uncertainty

Gaussians

$$p(x) \sim N(\mu, \sigma^2):$$

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}$$

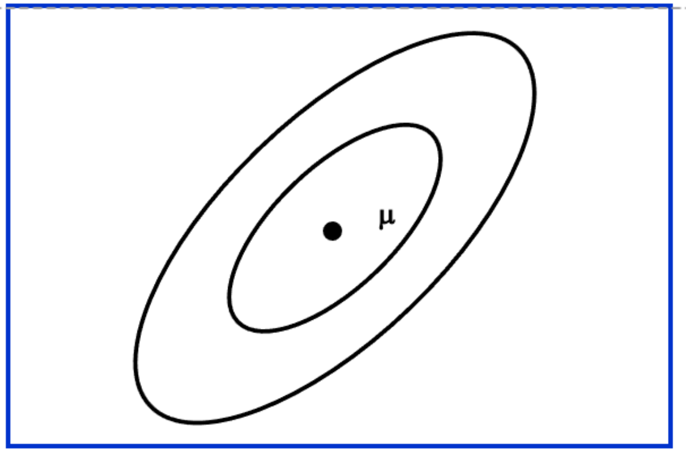
Univariate



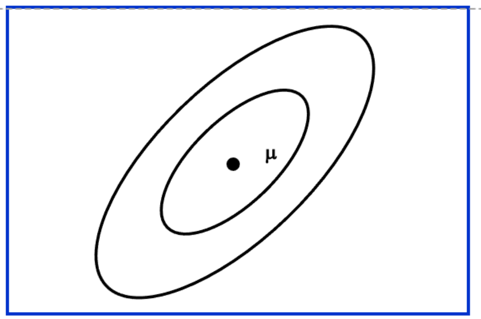
$$p(\mathbf{x}) \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma}):$$

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^t \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}$$

Multivariate

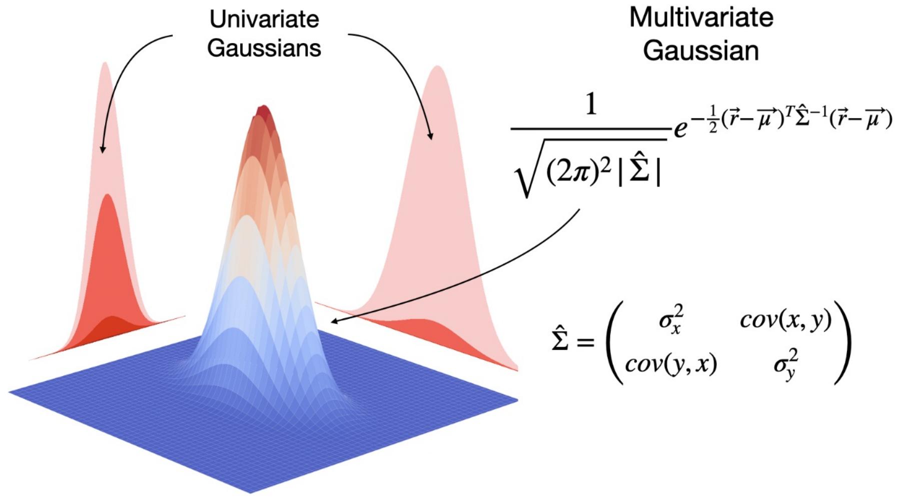


Multivariate Gaussians



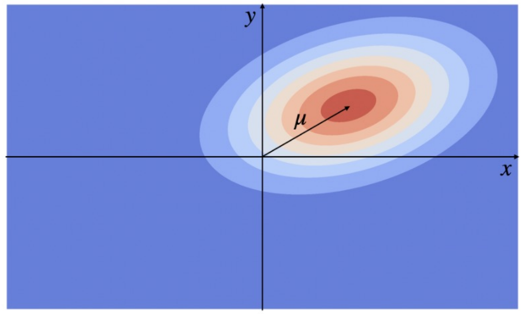
$$p(\mathbf{x}) \sim N(\boldsymbol{\mu},\boldsymbol{\Sigma}):$$

$$p(\mathbf{x}) \square \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^t \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}$$

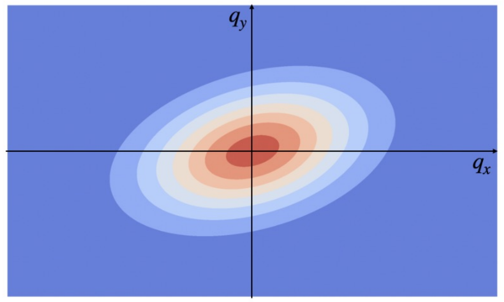

Univariate Gaussians
Multivariate Gaussian
$\frac{1}{\sqrt{(2\pi)^2|\hat{\Sigma}|}} e^{-\frac{1}{2}(\vec{r}-\vec{\mu})^T\hat{\Sigma}^{-1}(\vec{r}-\vec{\mu})}$
$\hat{\Sigma} = \begin{pmatrix} \sigma_x^2 & cov(x,y) \\ cov(y,x) & \sigma_y^2 \end{pmatrix}$

Every single variable x_i in x has a normal distribution $N(\mu_i, \sigma_i)$

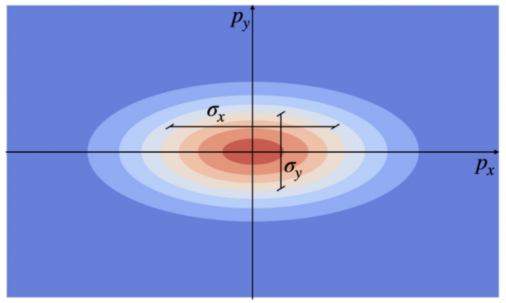
If the variables are uncorrelated then the covariance mat
Σ will be a diagonal matrix with the diagonal terms $\{\sigma_i^2\}$



Random bivariate Gaussian distribution



Bivariate Gaussian distribution centered in zero



Bivariate Gaussian distribution diagonalised

Properties of Gaussians

Closed under linear transformations: Linear transformations of Gaussians are Gaussians

$$\left.\begin{array}{l} X \sim N(\mu, \sigma^2) \\ Y = aX + b \end{array}\right\} \Rightarrow \quad Y \sim N(a\mu + b, a^2\sigma^2)$$

Products of independent Gaussians are Gaussians

$$\left.\begin{array}{l} X_1 \sim N(\mu_1, \sigma_1{}^2) \\ X_2 \sim N(\mu_2, \sigma_2{}^2) \end{array}\right\} \Rightarrow p(X_1) \cdot p(X_2) \sim N\left(\frac{\sigma_2{}^2}{\sigma_1{}^2 + \sigma_2{}^2}\mu_1 + \frac{\sigma_1{}^2}{\sigma_1{}^2 + \sigma_2{}^2}\mu_2, \quad \frac{1}{\sigma_1{}^{-2} + \sigma_2{}^{-2}} \right)$$

Multivariate Gaussians

$$\left.\begin{array}{r} X \sim N(\mu, \Sigma) \\ Y = AX + B \end{array}\right\} \Rightarrow \quad Y \sim N(A\mu + B, A\Sigma A^T)$$

$$\left.\begin{array}{l} X_1 \sim N(\mu_1, \Sigma_1) \\ X_2 \sim N(\mu_2, \Sigma_2) \end{array}\right\} \Rightarrow p(X_1) \cdot p(X_2) \sim N\left(\frac{\Sigma_2}{\Sigma_1 + \Sigma_2} \mu_1 + \frac{\Sigma_1}{\Sigma_1 + \Sigma_2} \mu_2, \quad \frac{1}{\Sigma_1^{-1} + \Sigma_2^{-1}} \right)$$

We stay in the “Gaussian world” as long as we start with Gaussians and perform only linear transformations.

Discrete Kalman Filter

Estimates the state x of a Discrete Linear Time Invariant System with Gaussian noise

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

with a measurement

$$\boxed{z_t = C_t x_t + \delta_t}$$

What is a Kalman Filter?

Suppose we have a system that is governed by a linear difference equation:

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

$\epsilon_t \sim N(0, \mathrm{R_t});\ \delta_t \sim N(0, Q_t)$

with measurement

$$z_t = C_t x_t + \delta_t$$

- Tracks the estimated state of the system by the mean and variance of its state variables -- minimum mean-square error estimator
- Computes the ***Kalman gain,*** which is used to weight the impact of new measurements on the system state estimate against the predicted value from the process model
- Note that we no longer have discrete states or measurements!

Components of a Kalman Filter

$\boxed{A_t}$ Matrix (nxn) that describes how the state evolves from t to $t-1$ without controls or noise.

$\boxed{B_t}$ Matrix (nxl) that describes how the control u_t changes the state from t to $t-1$.

$\boxed{C_t}$ Matrix (kxn) that describes how to map the state x_t to an observation z_t.

$\boxed{\varepsilon_t}$ $\boxed{\delta_t}$ Random variables representing the process and measurement noise that are assumed to be independent and normally distributed with covariance R_t and Q_t respectively.

Linear Gaussian System: Dynamics

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

//Dynamic model: $p(x_t \mid x_{t-1}, u_t) = N(A_t x_{t-1} + B_t u_t, R_t)$

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

Prediction step

$\overline{bel}(x_t) = \int p(x_t | u_t, x_{t-1}) bel\,(x_{t-1}) dx_{t-1}$

...

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

Linear Gaussian System: Observations

- $z_t = C_t x_t + \delta_t$ // $p(z_t|x_t) = N(z_t; C_t x_t, Q_t)$

Correction step

$$bel(x_t) = \eta\, p(z_t|x_t)\ \overline{bel}(x_t)$$

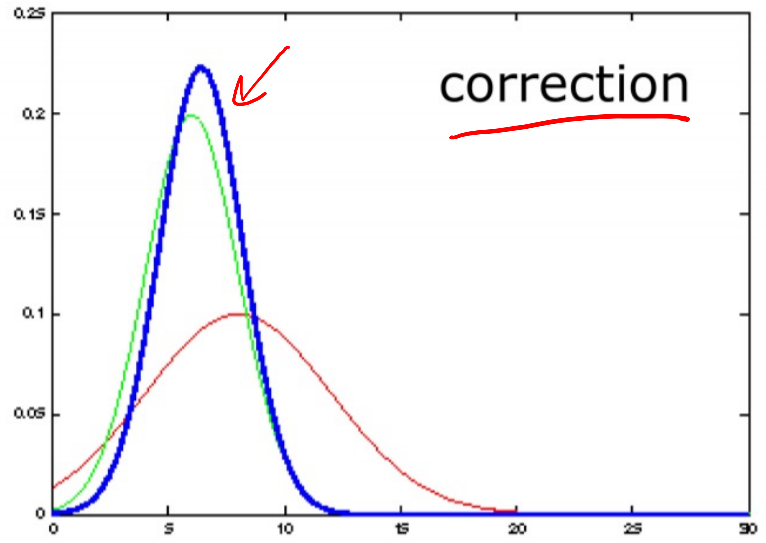
…

$$= \begin{cases} \mu_t = \bar{\mu}_t + K_t(z_t - C_t\,\bar{\mu}_t) \\ \Sigma_t = (I - K_t C_t)\,\bar{\Sigma}_t \end{cases}$$

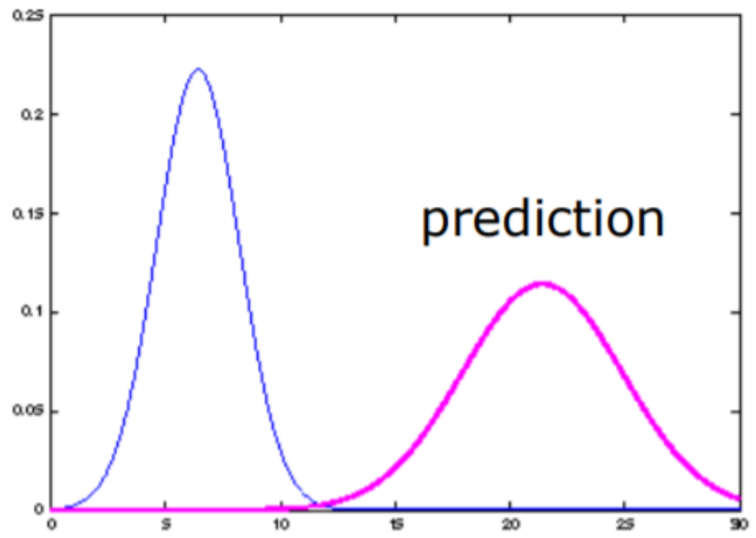
where $K_t = \bar{\Sigma}_t C_t (C_t\,\bar{\Sigma}_t\,C_t^T + Q_t)^{-1}$ is called the Kalman gain

Kalman Filter Algorithm

1. Algorithm Kalman_Filter($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):
2. Prediction
    1. $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$
    2. $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^\top + Q_t$
3. Correction:
    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. $\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$
4. Return μ_t, Σ_t



Apply control action

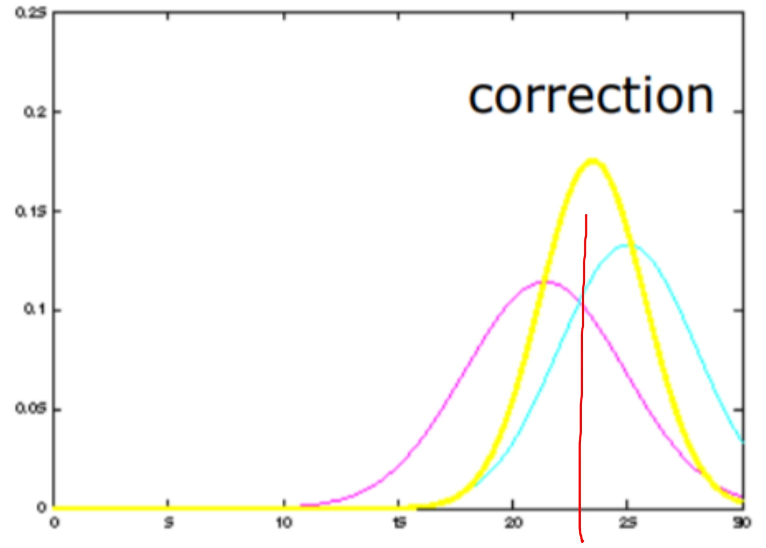


Correction:

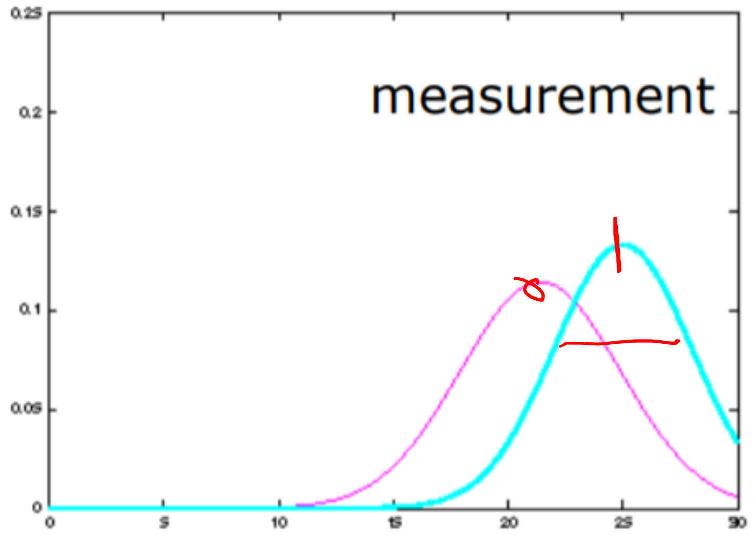
1. $K_t = \bar{\Sigma}_t C_t^{\top} (C_t \bar{\Sigma}_t C_t^{\top} + R_t)^{-1}$
2. $\mu_t = \bar{\mu}_t + K_t(z_t - C_t \bar{\mu}_t)$
3. $\Sigma_t = (I - K_t C_t)\bar{\Sigma}_t$

Prediction:

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

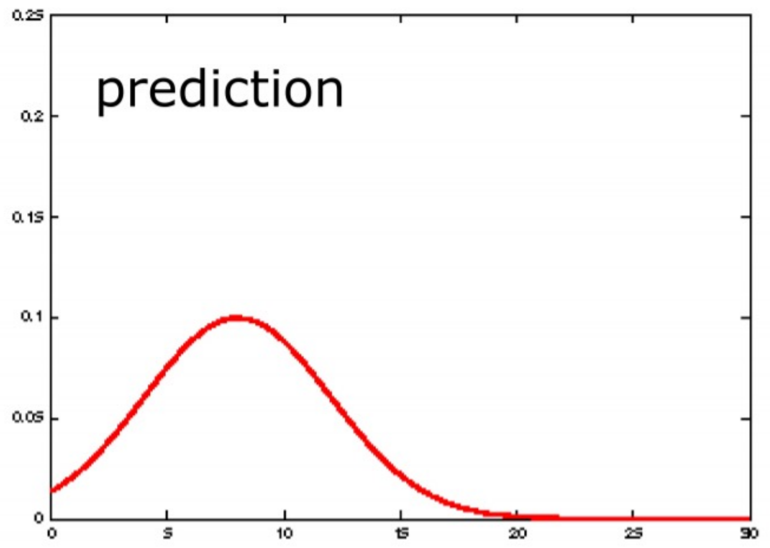


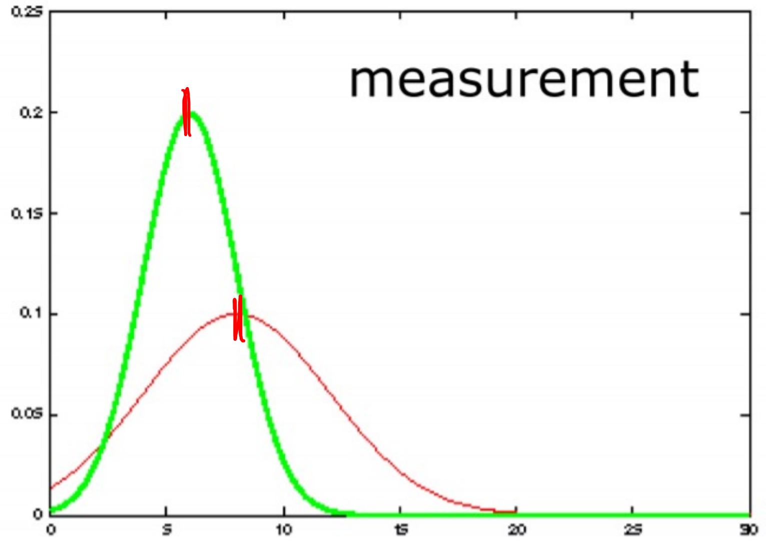
Get sensor measurement

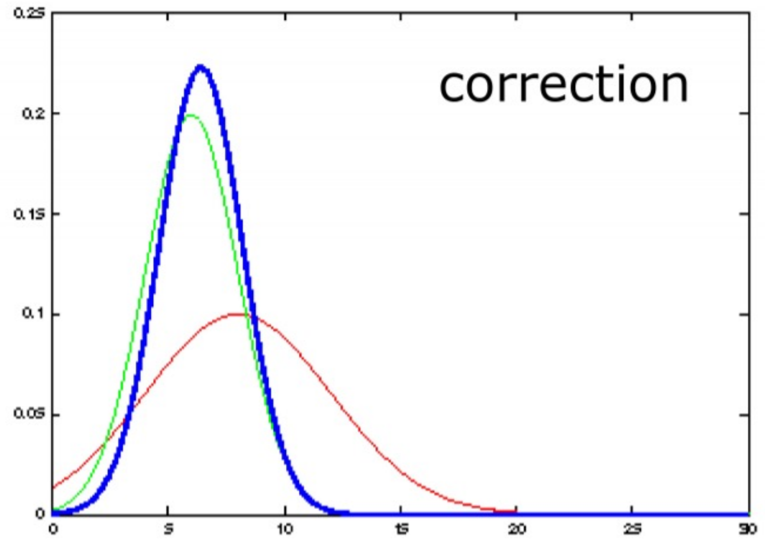


Kalman Filter Example

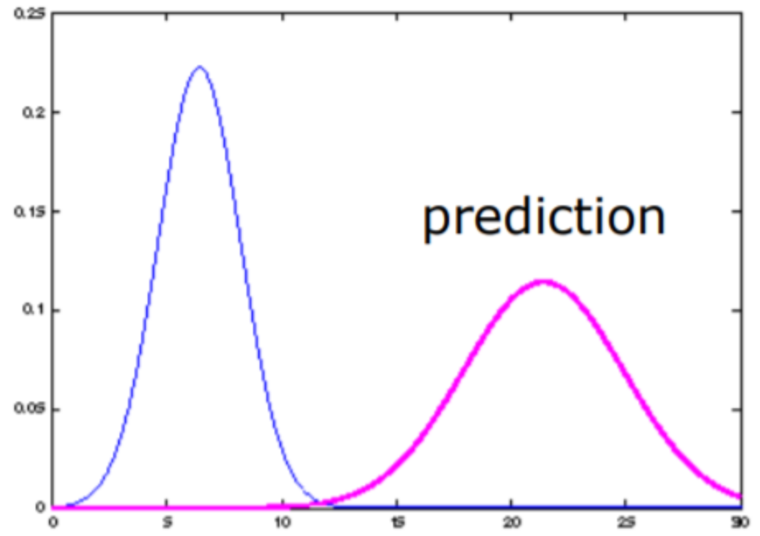
time = 1

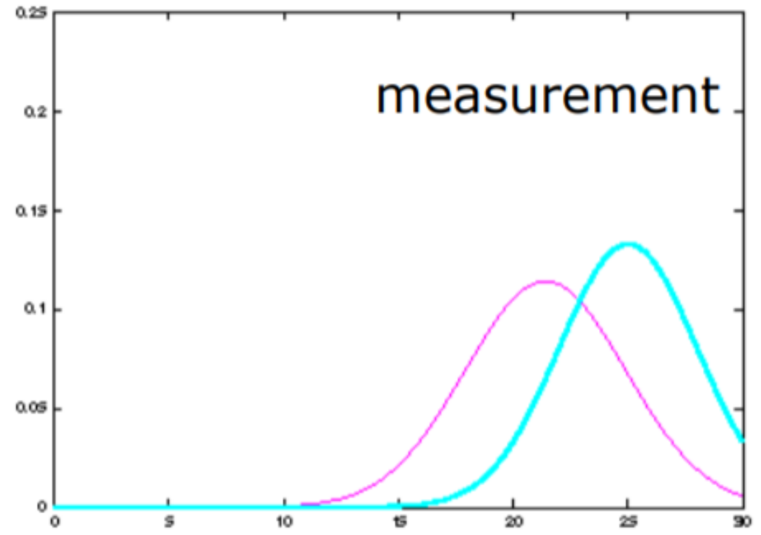


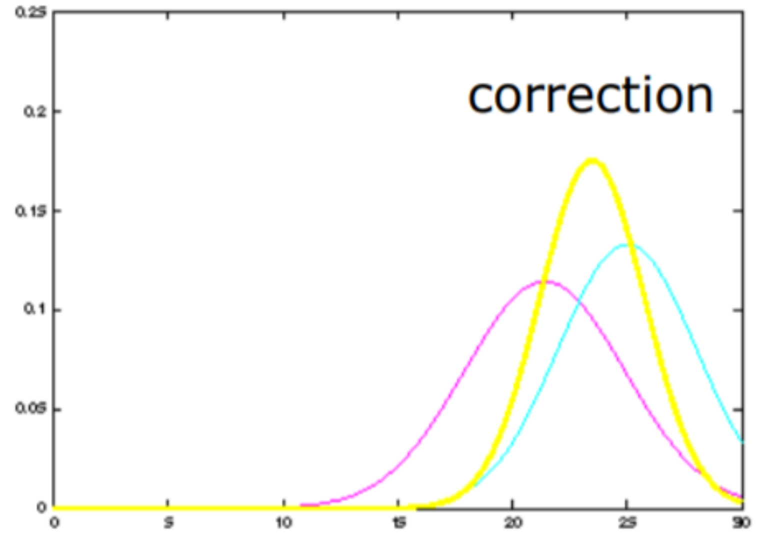




time = 2







Who was Rudolf Kalman?

Image Credit: Wikipedia

- Kálmán was one of the most influential people on control theory and is most known for his co-invention of the Kalman filter (or Kalman-Bucy Filter)
- The filtering approach was initially met with vast skepticism, so much so that he was forced to do the first publication of his results in mechanical engineering, rather than in electrical engineering or systems engineering
  - This worked out fine as some of the first use cases was with NASA on the Apollo spacecraft
- *Kalman filters are inside every robot, commercial airplanes, uses in seismic data processing, nuclear power plant instrumentation, and demographic models, as well as applications in econometrics*

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

The SLAM Problem

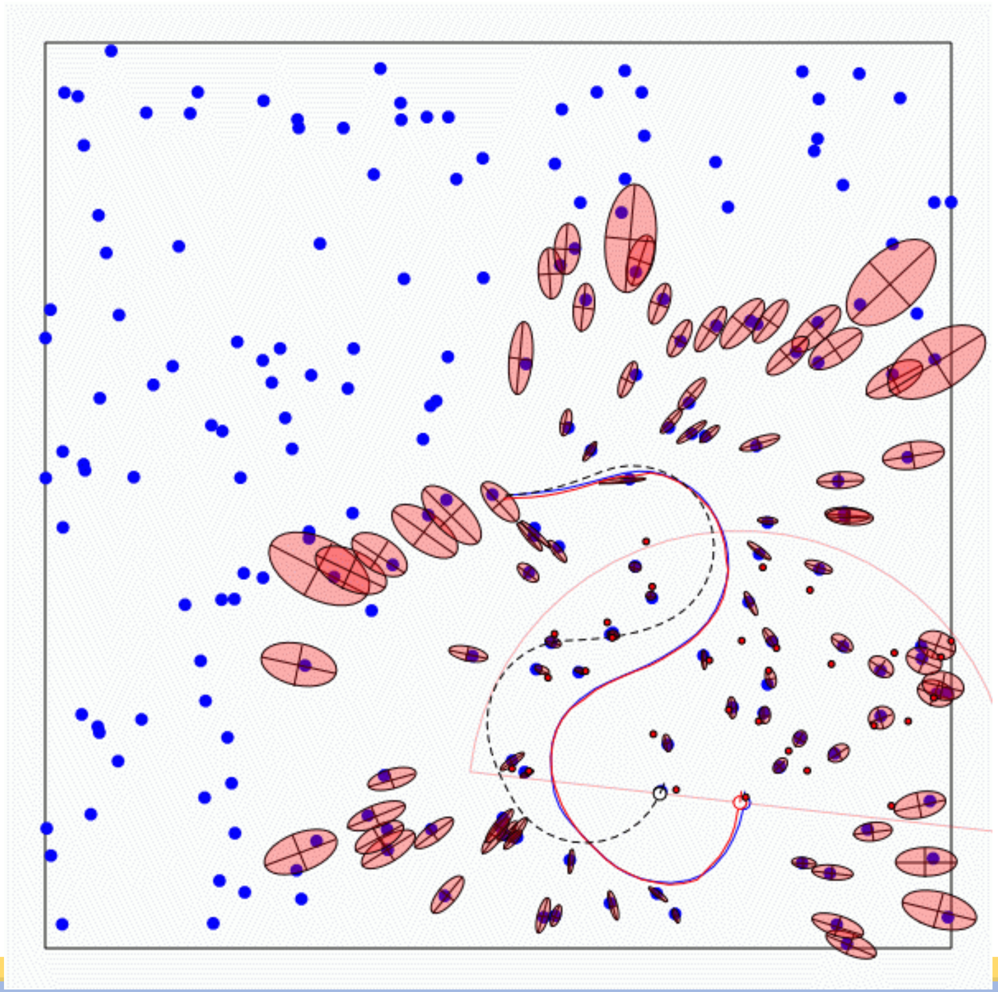
A robot moving though an unknown, static environment

Given:

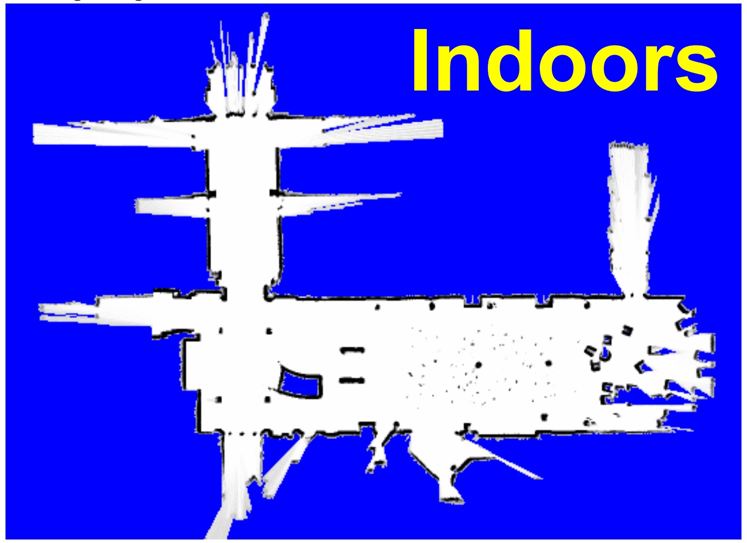
- The robot’s controls
- Observations of nearby features

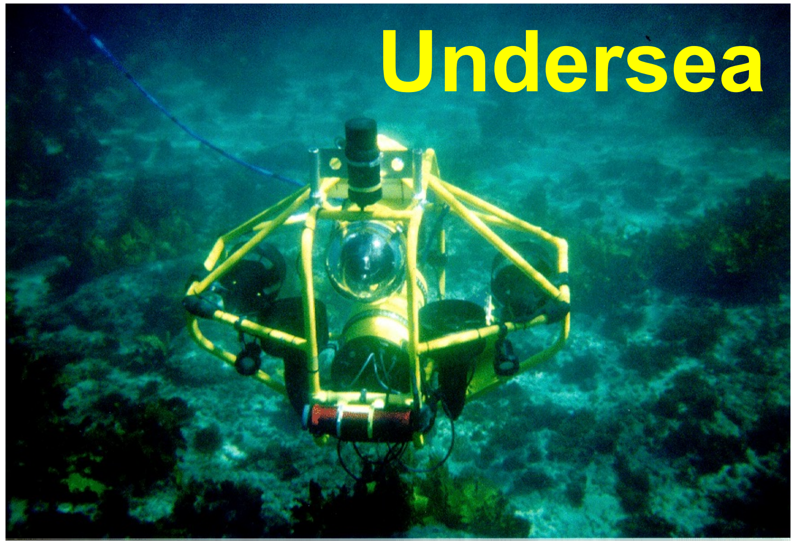
Estimate:

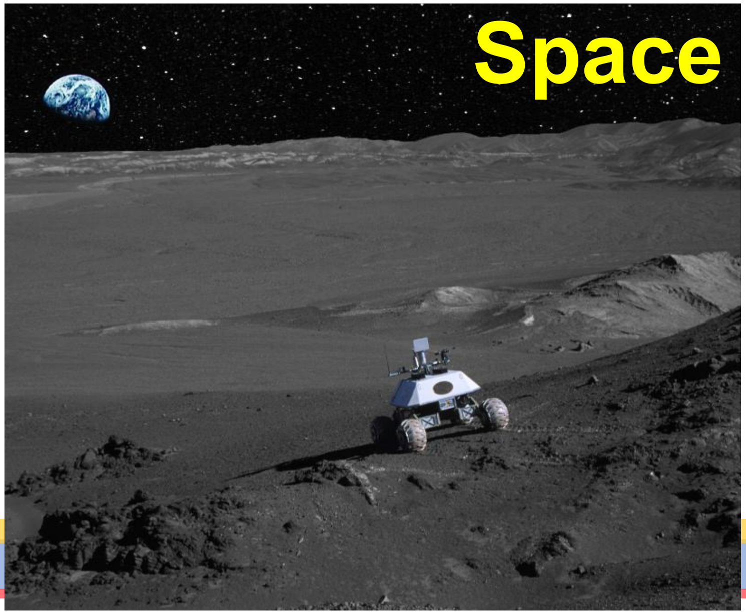
- Map of features
- Path of the robot



SLAM Applications



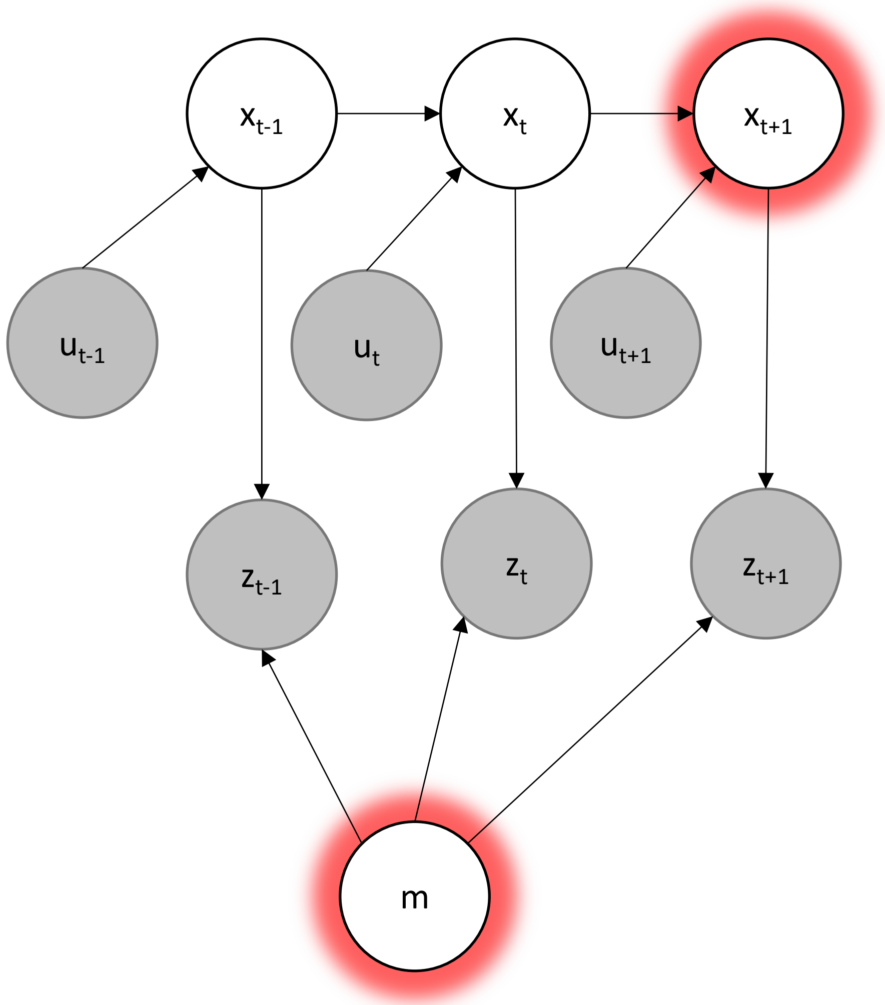




Forms of SLAM

- 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*
  - $p(x_t, m, c_t \,|z_{1:t}, u_{1:t})$
- Many algorithms: EKFSLAM, GraphSLAM, FastSLAM

Online SLAM

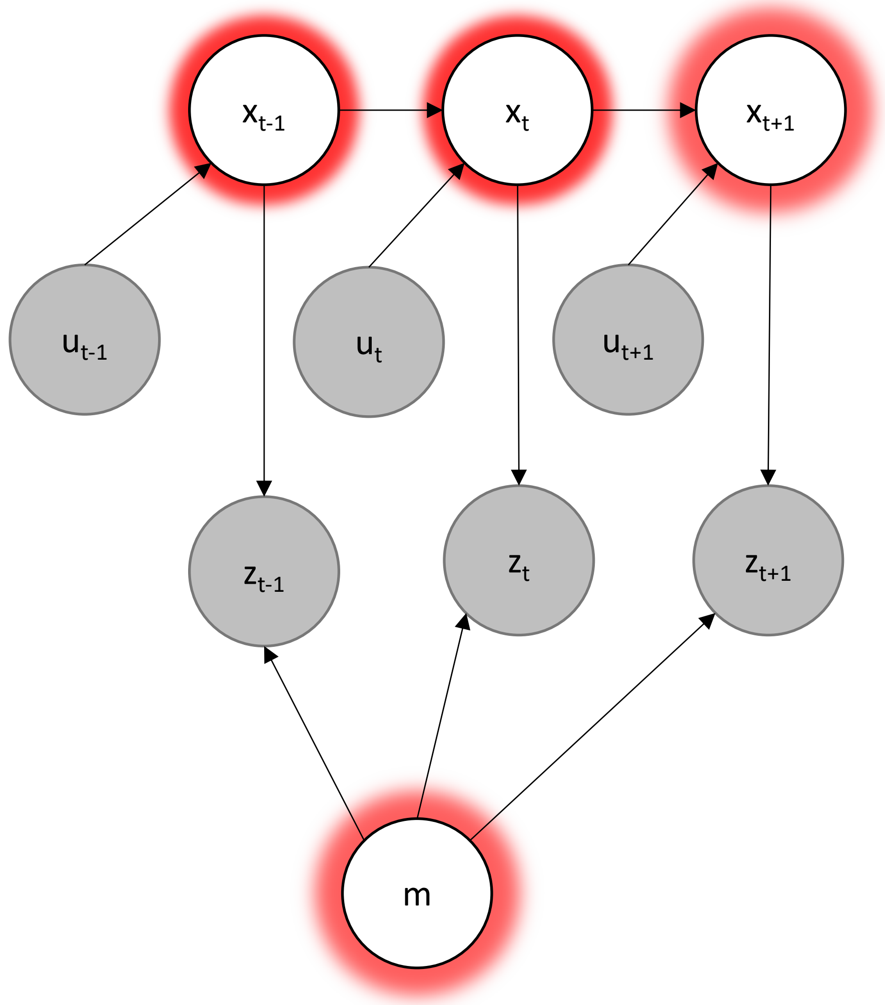


Shaded known:
control inputs (u),
measurements(z).

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

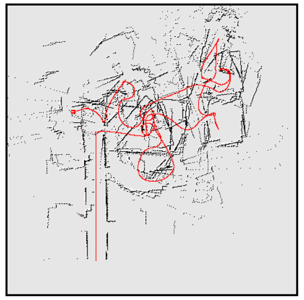
Continuous unknowns: $x_{1:t}, m$
Discrete unknowns: Relationship of detected objects to new objects

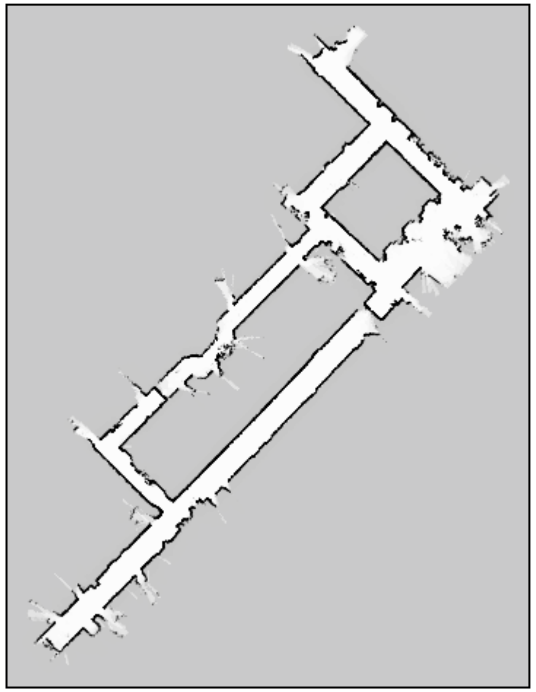
$$p(x_{1:t}, c_t, m | z_{1:t}, u_{1:t})$$

c_t: corrsnpondence variable

Representations

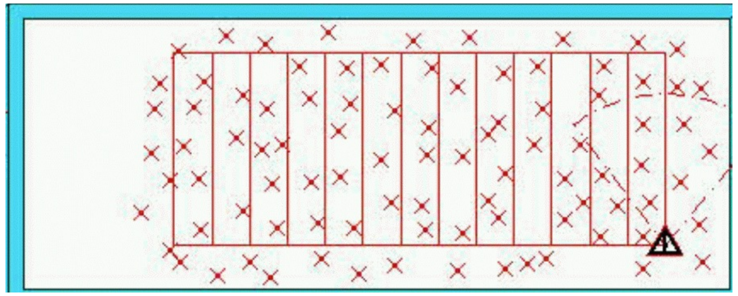
Grid maps or scans

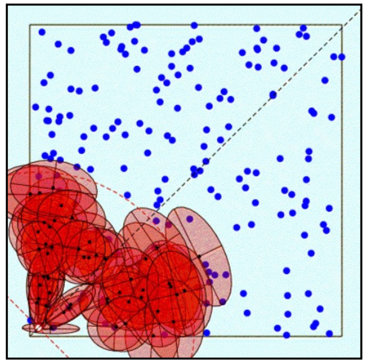


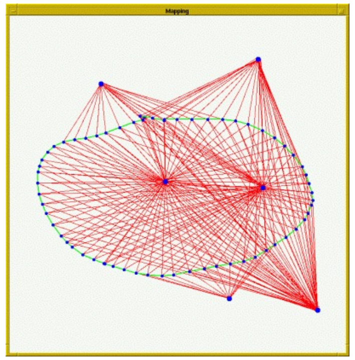


[Lu & Milios, 97; Gutmann, 98: Thrun 98; Burgard, 99; Konolige & Gutmann, 00; Thrun, 00; Arras, 99; Haehnel, 01;…]

Landmark-based



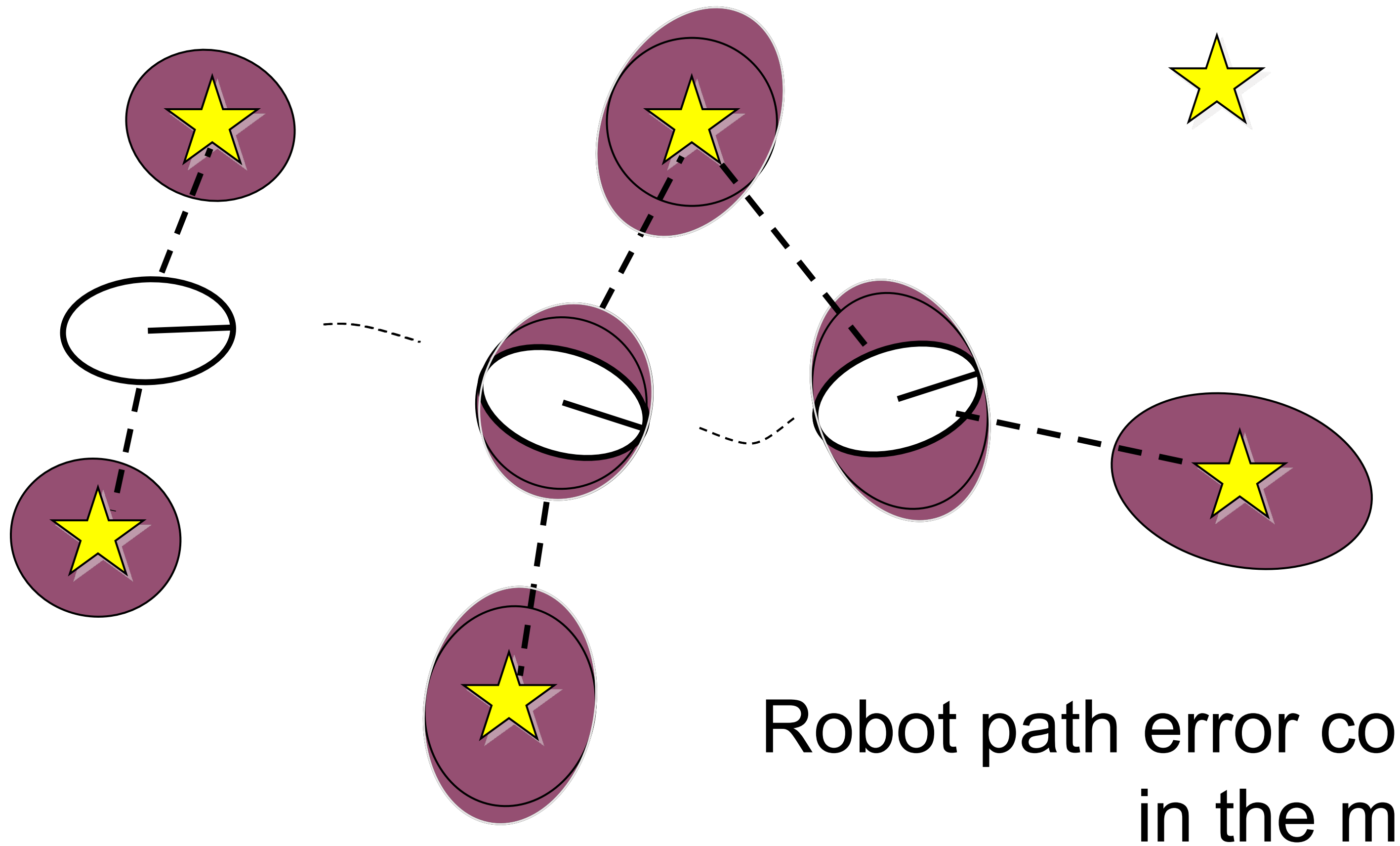




[Leonard et al., 98; Castelanos et al., 99: Dissanayake et al., 2001; Montemerlo et al., 2002;…

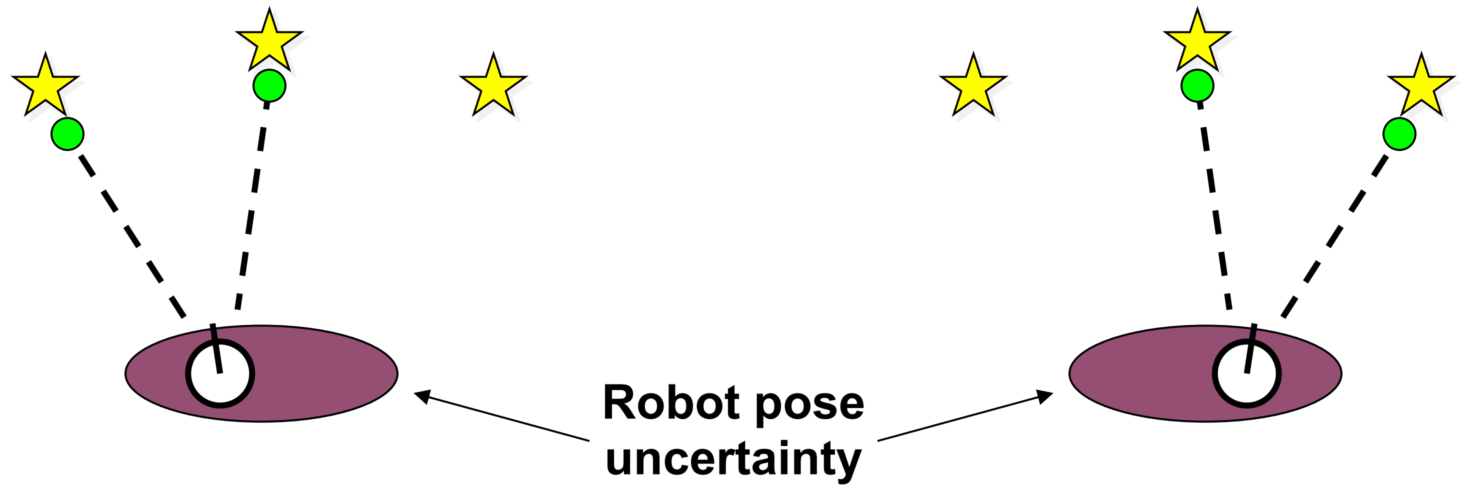
Why is SLAM a hard problem?

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

SLAM:

Simultaneous Localization and Mapping

- Full SLAM:

Estimates entire path and map!

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

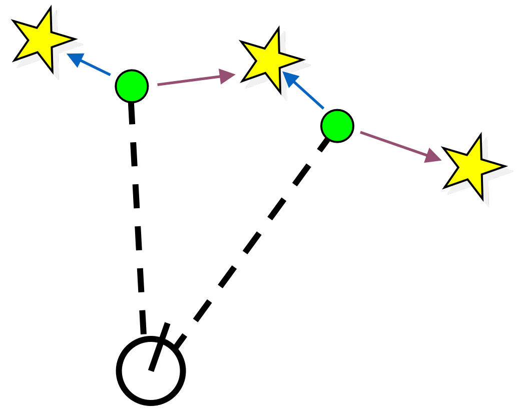
- Online SLAM:

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

Integrations typically done one at a time

Estimates most recent pose and map!

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 $\langle x, y, \theta \rangle$
- SLAM: state space $\langle x, y, \theta, map \rangle$
  - for landmark maps = $\langle l_1, l_2, \ldots, l_m \rangle$
  - for grid maps = $\langle c_{11}, c_{12}, \ldots, c_{1n}, c_{21}, \ldots, c_{nm} \rangle$
- **Problem:** The number of particles needed to represent a posterior grows exponentially with the dimension of the state space!

- 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?

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

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

- Height and vocabulary are not independent
- But they are conditionally independent given age

Factored Posterior (Landmarks)

poses map observations & movements

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

SLAM posterior

Robot path posterior

landmark positions

Does this help to solve the problem?

Factorization first introduced by Murphy in 1999

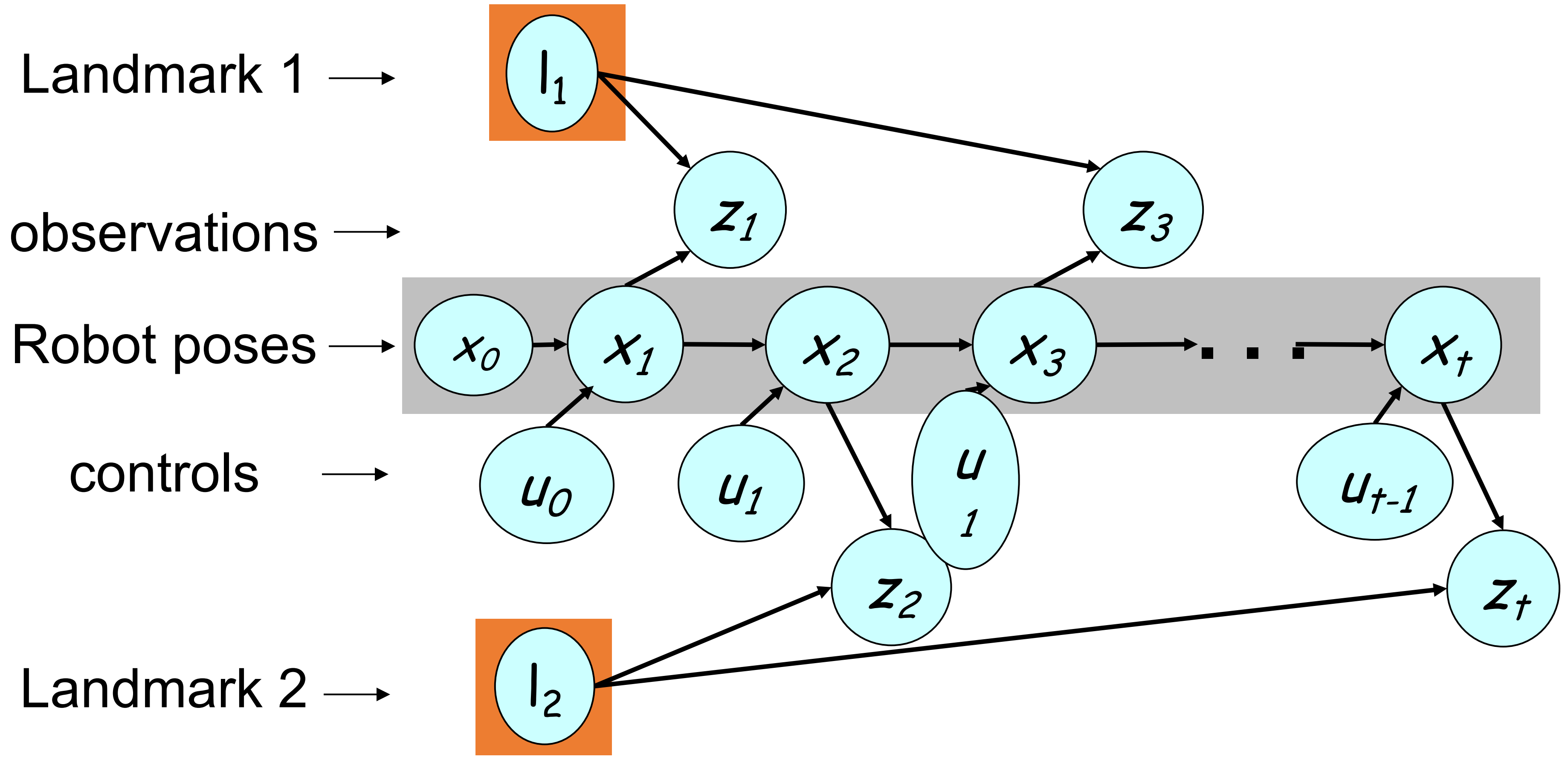
Factored Posterior (Landmarks)

$$p(\overset{\text{poses}}{x_{1:t}}, \overset{\text{map}}{l_{1:m}} \mid \overbrace{z_{1:t}, u_{0:t-1}}^{\text{observations \& movements}}) =$$

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

Factorization first introduced by Murphy in 1999

Mapping using Landmarks



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

Factored Posterior

$$
\begin{aligned}
& p(x_{1:t}, l_{1:m} \mid z_{1:t}, u_{0:t-1}) \\
&\quad = \; p(x_{1:t} \mid z_{1:t}, u_{0:t-1}) \cdot p(l_{1:m} \mid x_{1:t}, z_{1:t}) \\
&\quad = \; \underbrace{p(x_{1:t} \mid z_{1:t}, u_{0:t-1})}_{\text{Robot path posterior (localization problem)}} \cdot \prod_{i=1}^{M} \underbrace{p(l_i \mid x_{1:t}, z_{1:t})}_{\text{Conditionally independent landmark positions}}
\end{aligned}
$$

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 Harold Blackwell (1919-2010)

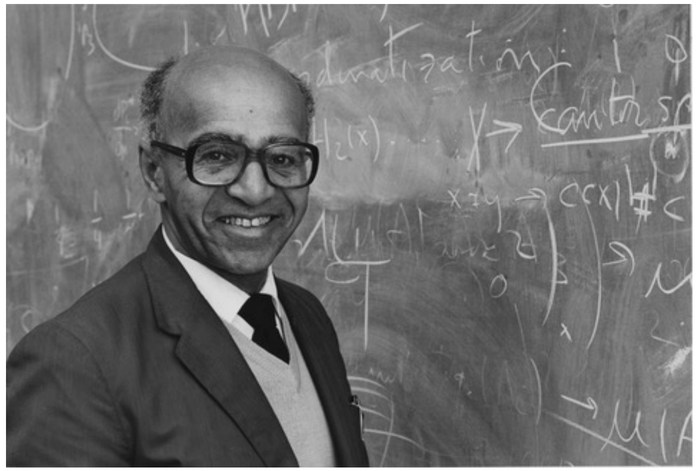


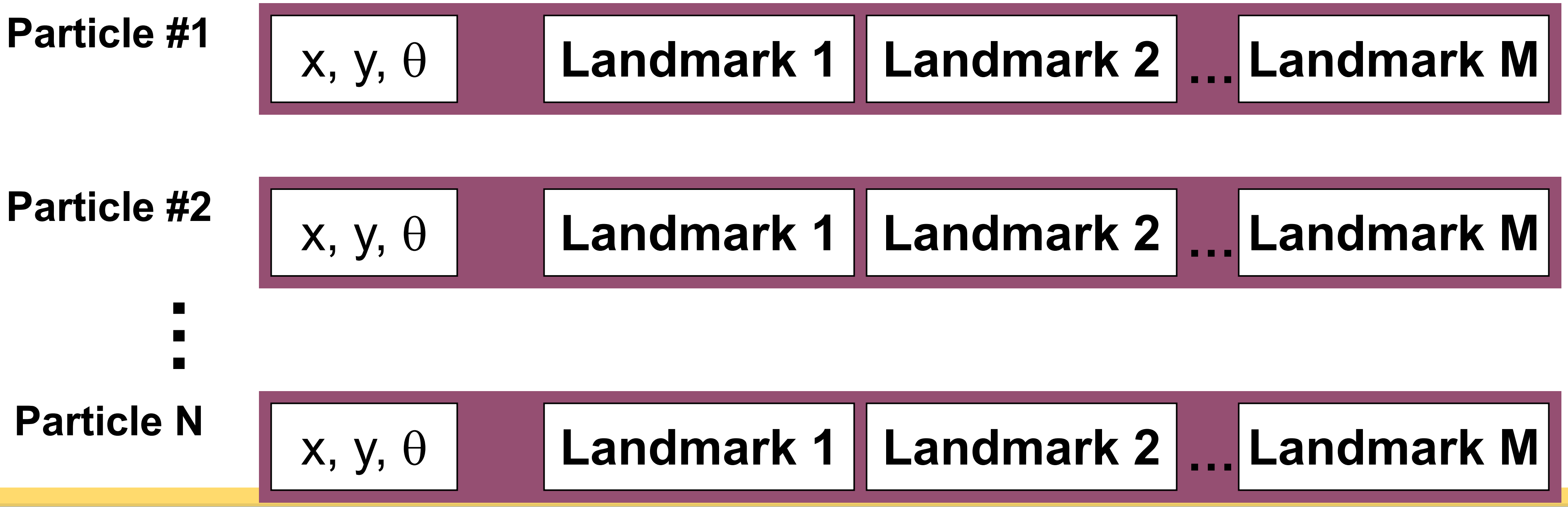
photo from stat.illinois

Independently developed dynamic programming. Several theorems that bear his name, including the Blackwell renewal theorem, used in engineering, and the Rao-Blackwell theorem in statistics.

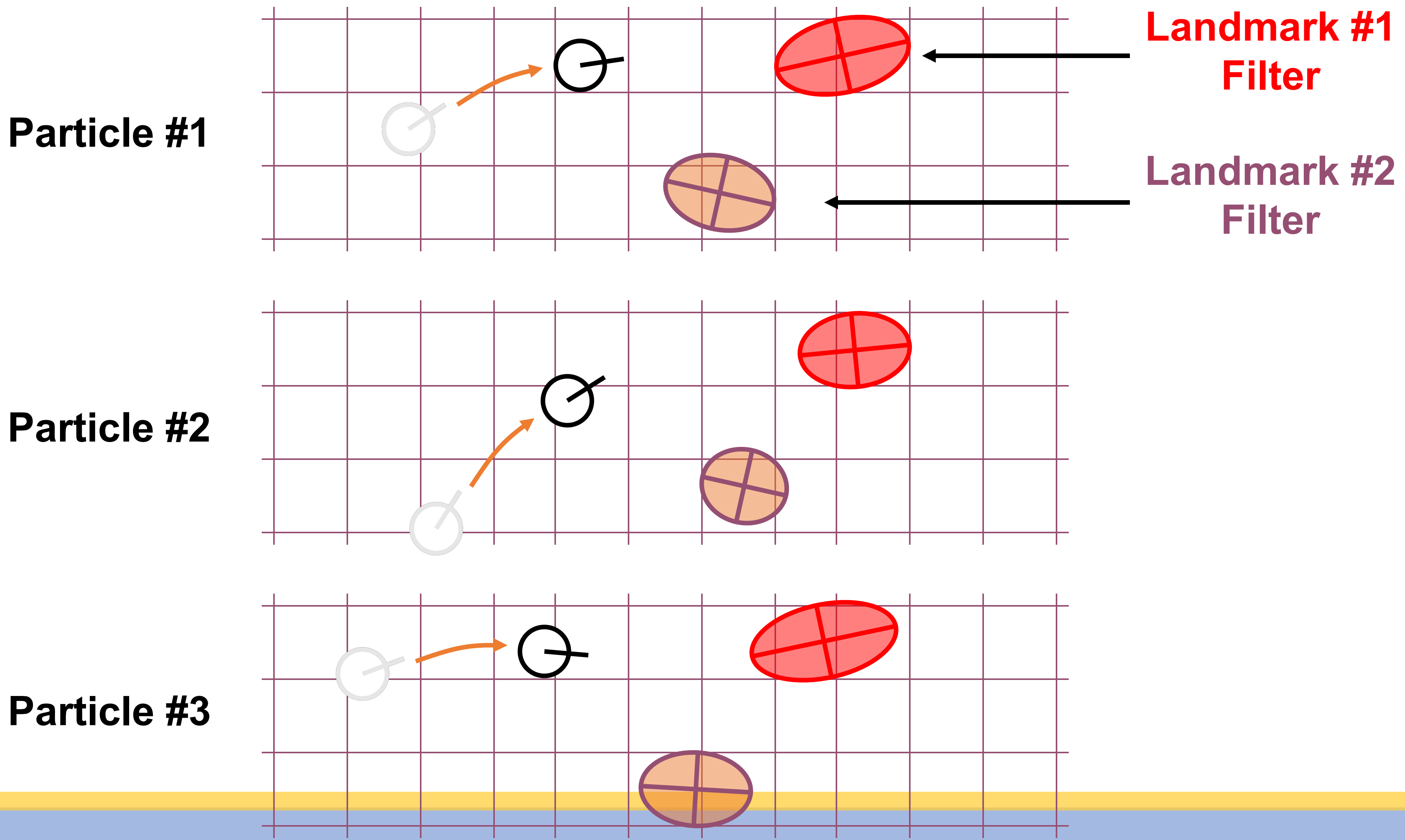
University of Illinois at Urbana-Champaign (BA, MA, PhD 1941)

FastSLAM

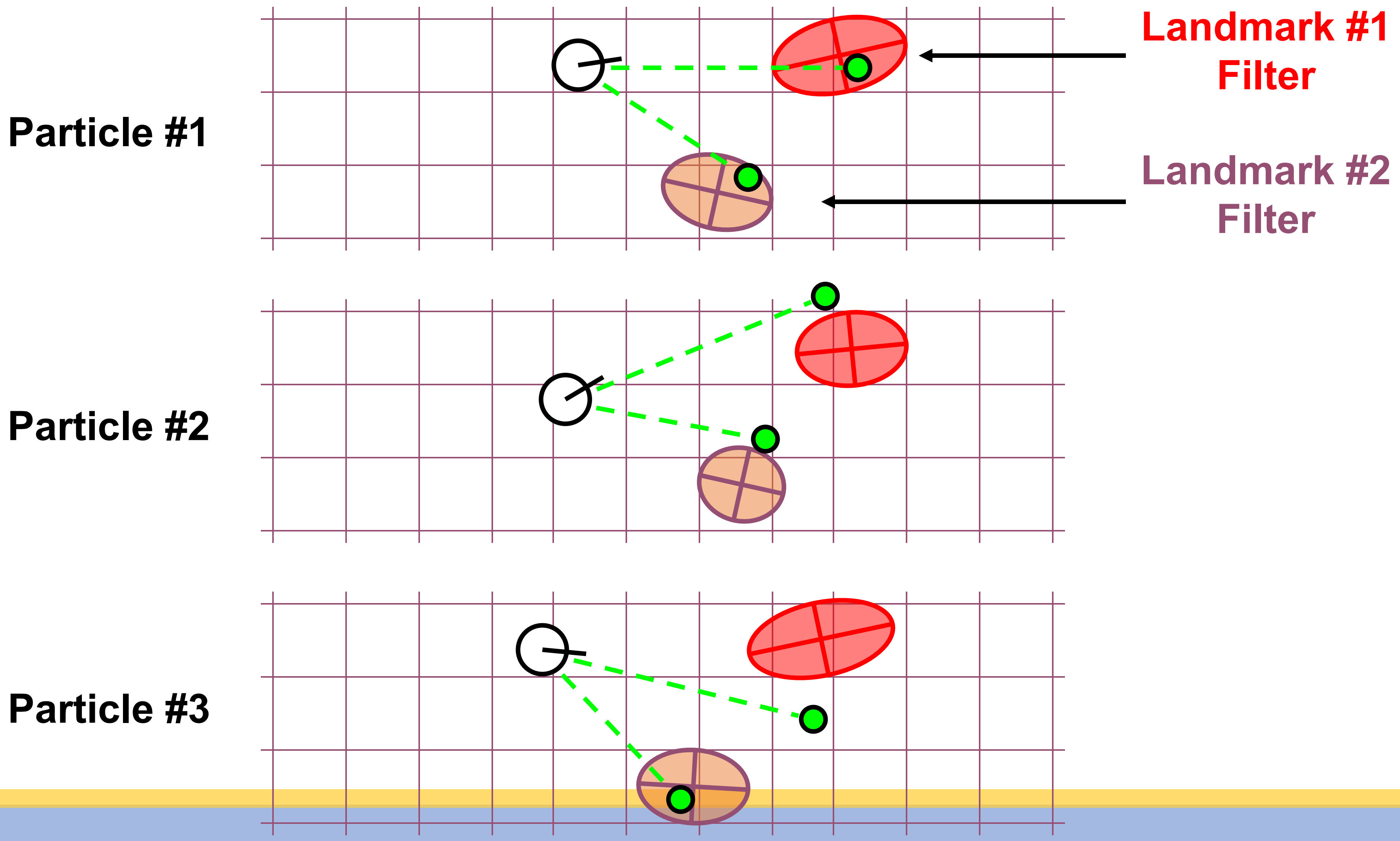
- Rao-Blackwellized particle filtering based on landmarks [Montemerlo et al., 2002]
- Each landmark is represented by a 2x2 Extended Kalman Filter (EKF)
- Each particle therefore has to maintain *M* EKFs



FastSLAM – Action Update

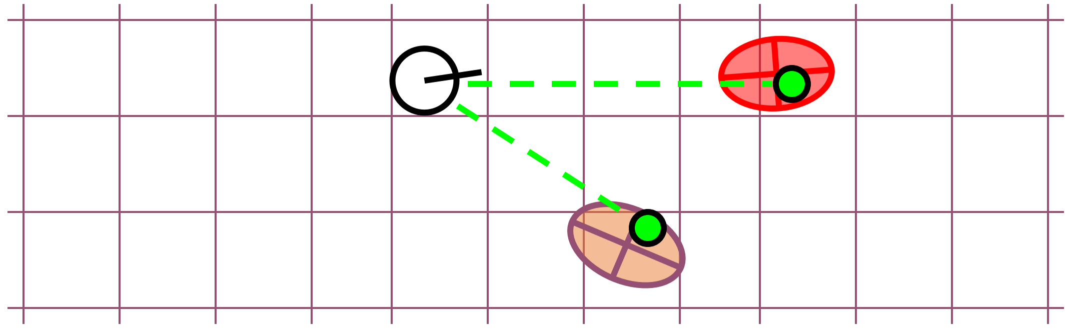


FastSLAM – Sensor Update



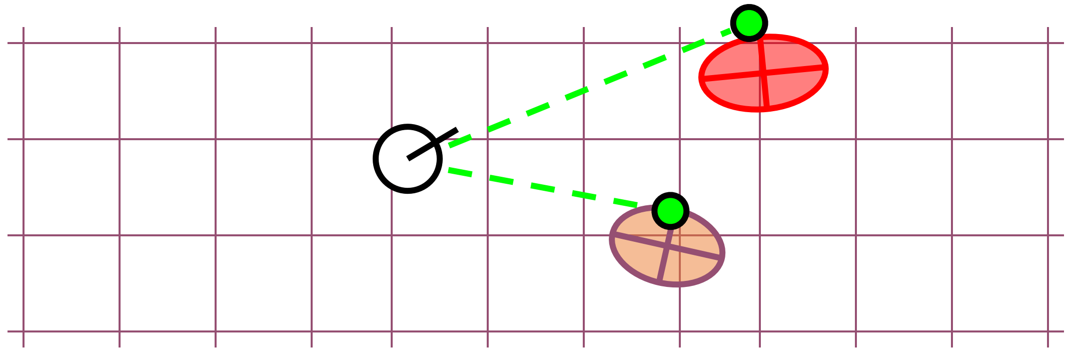
FastSLAM – Sensor Update

Particle #1



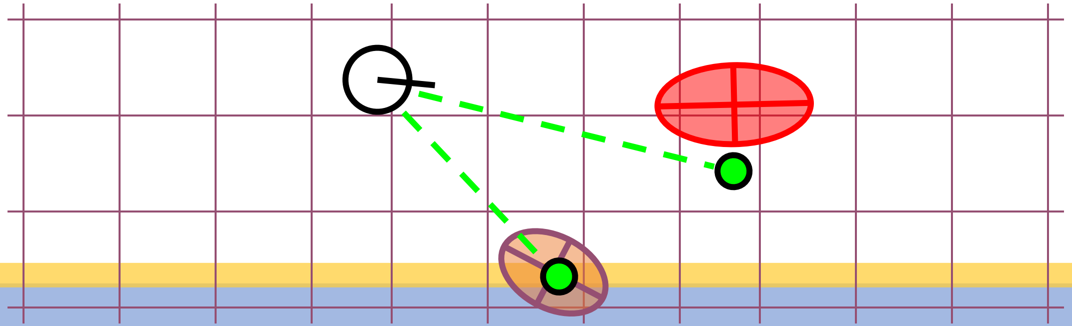
Weight = 0.8

Particle #2



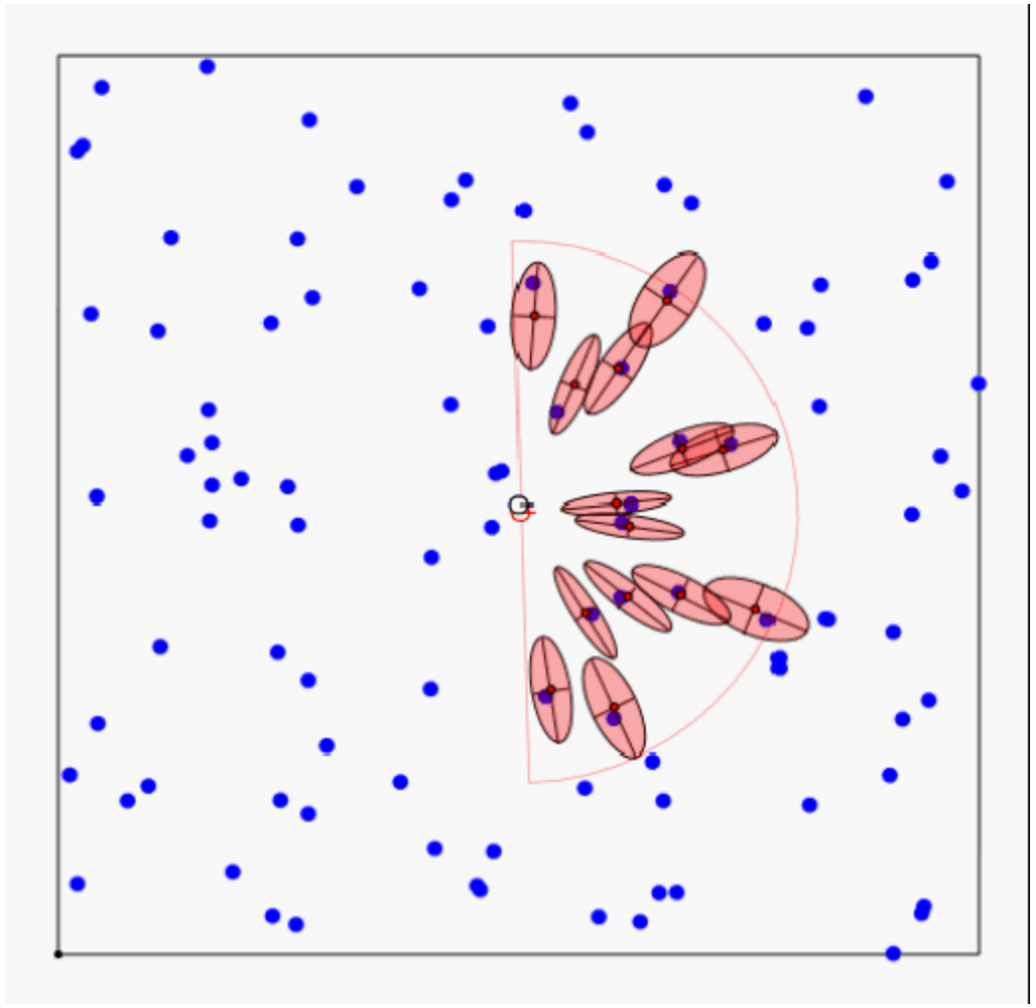
Weight = 0.4

Particle #3



Weight = 0.1

FastSLAM - Video



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

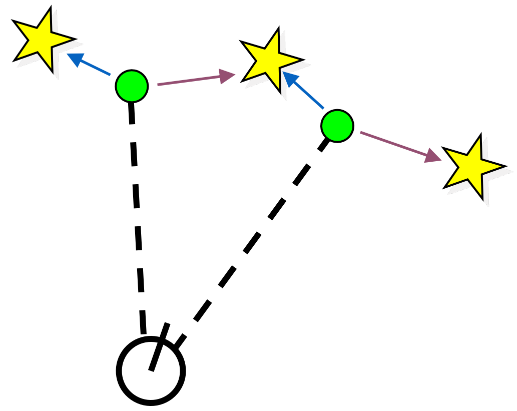
  $O(N \cdot \log(M))$
  Log time per particle

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

N = Number of particles
M = Number of map features

Data Association Problem

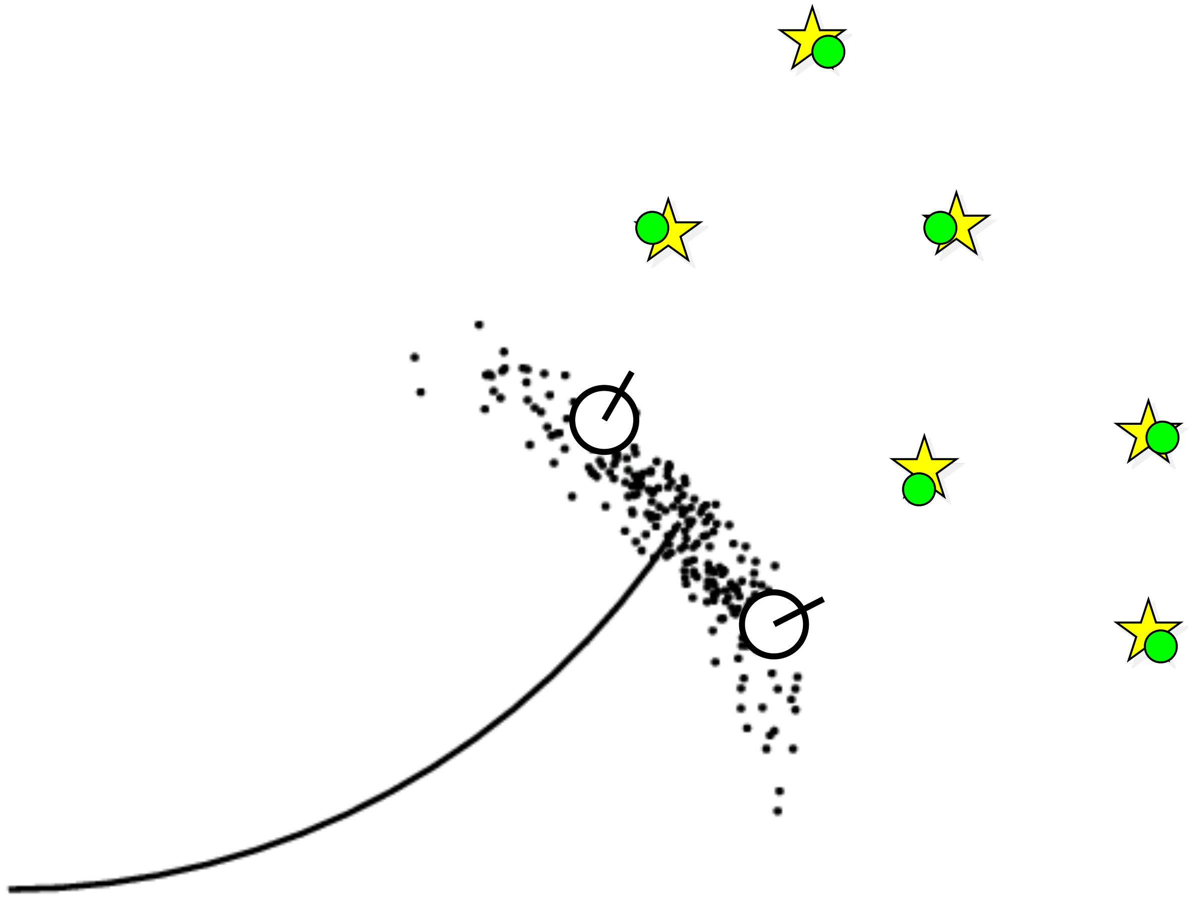
- Which observation belongs to which landmark?



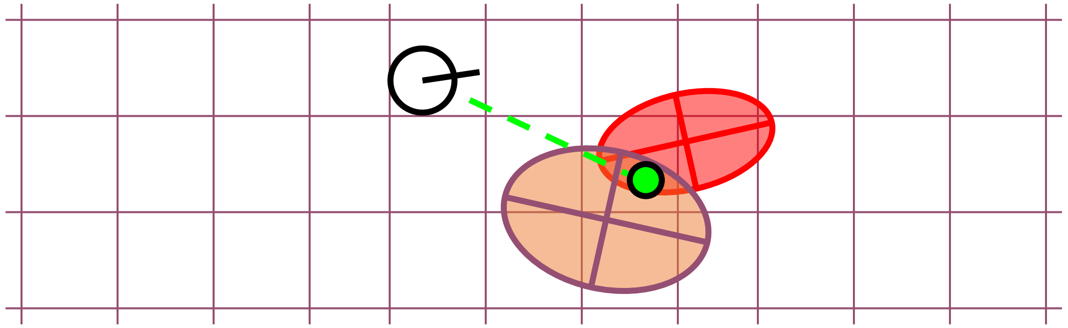
- A robust SLAM must consider possible data associations
- Potential data associations depend also on the pose of the robot

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 blue landmark?

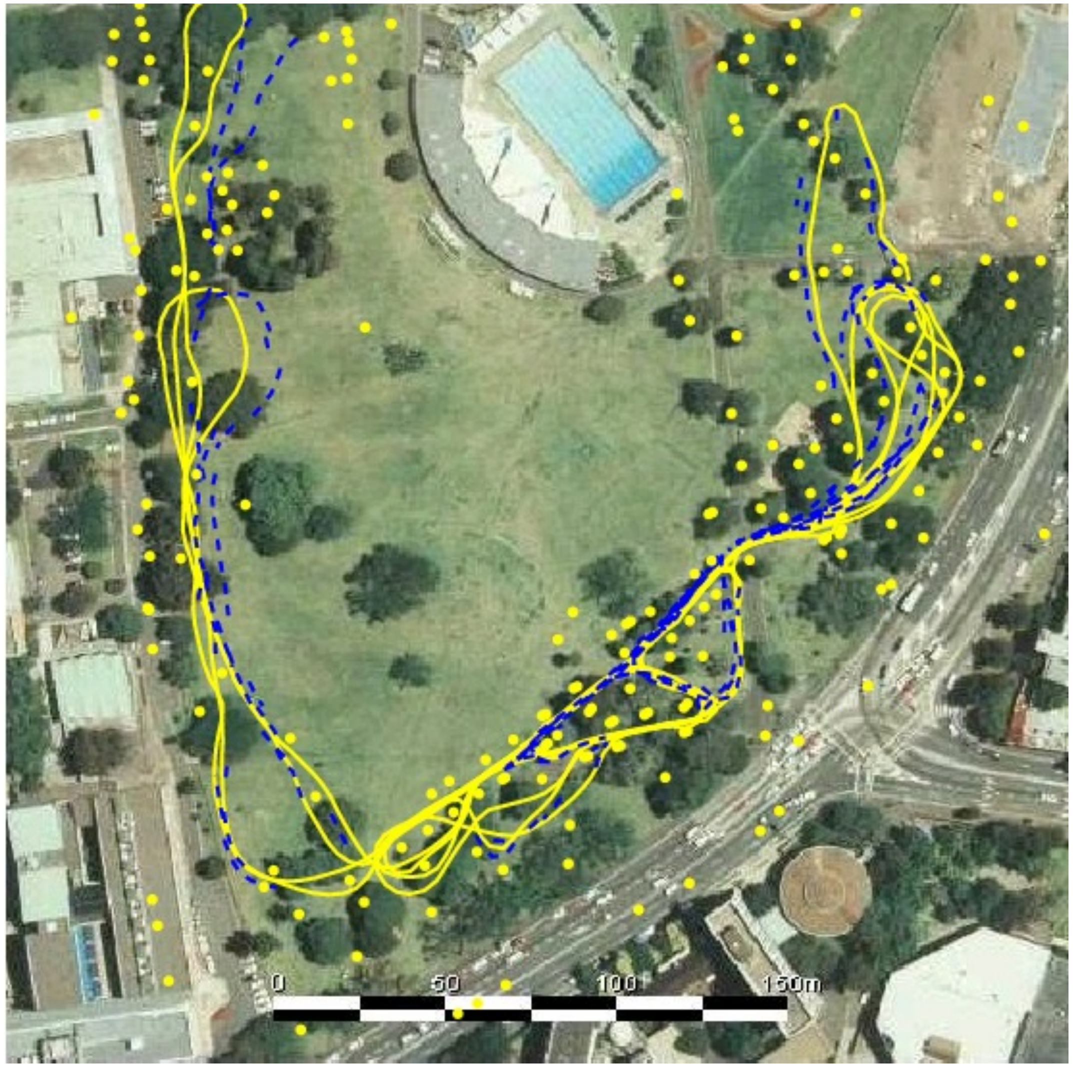
P(observation|red) = 0.3 P(observation|blue) = 0.7

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

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 linear is 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
- Update rule similar to conventional particle filter
- Each particle can be based on a different data association