Introduction to Mobile Robotics

Landmark-based FastSLAM Implementation in Webots

Wolfram Burgard, Michael Krawez

UTN

Project Outline

- Today
  - Recap of FastSLAM
  - Landmark-based FastSLAM algorithm
  - Intro to Webots and code framework
- Following sessions (TBD)
  - Implementation of basic algorithm
  - Efficient data structure
  - Data association
  - Landmark detection
  - …

The SLAM Problem

- SLAM stands for simultaneous localization and mapping
- The task of building a map while estimating the pose of the robot relative to this map

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

Particle Filters

- Represent belief by random samples
- Estimation of non-Gaussian, nonlinear processes

- Sampling Importance Resampling (SIR) principle
  - Draw the new generation of particles
  - Assign an importance weight to each particle
  - Resample

- Typical application scenarios are tracking, localization, …

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!

Dependencies

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

Dependencies

- Is there a dependency between certain 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.

Factored Posterior (Landmarks)

poses map observations & movements

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

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

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

Rao-Blackwellization

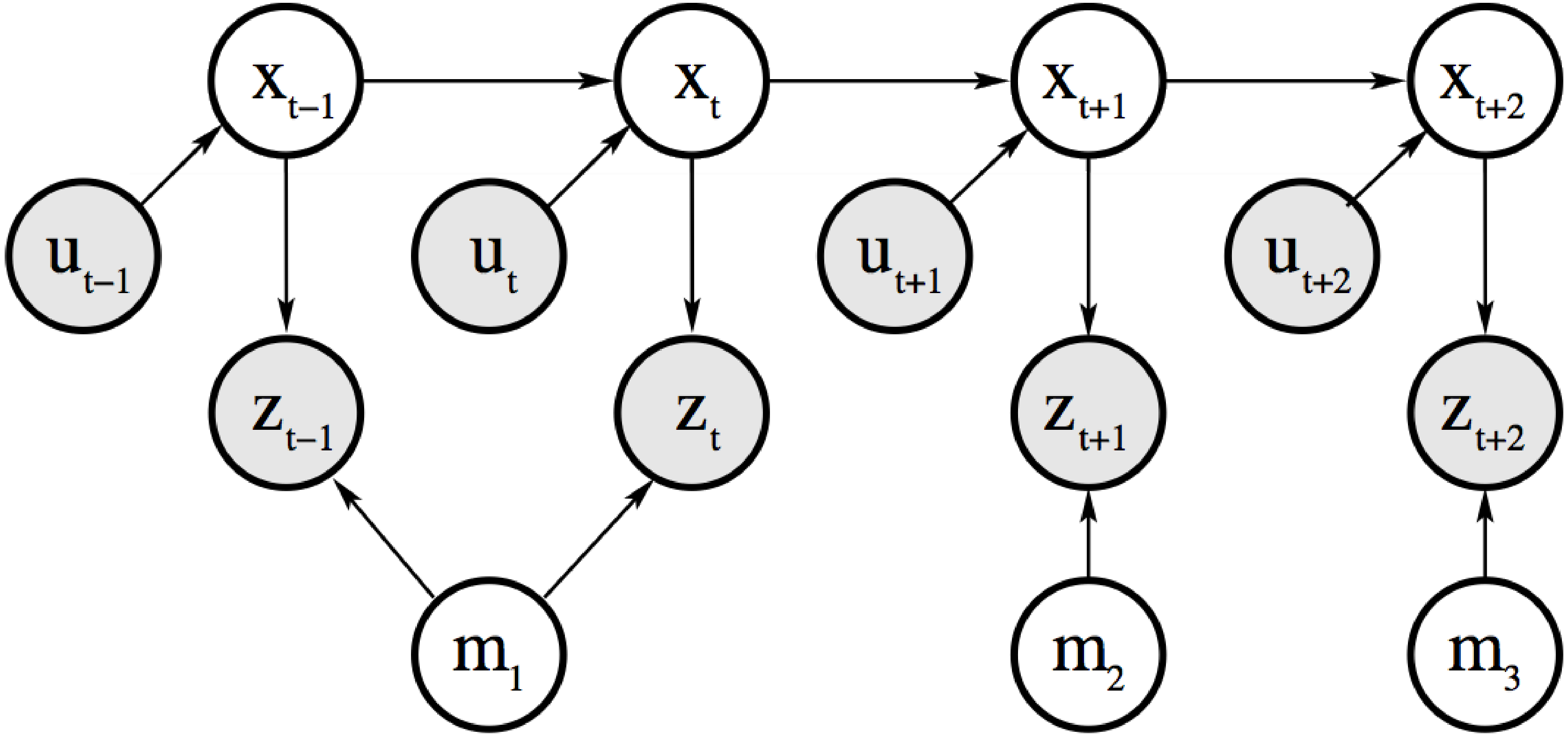
- Factorization to exploit dependencies between variables:

$$p(a, b) \;=\; p(a) \cdot p(b \mid a)$$

- If $p(b \mid a)$ can be computed in closed form, represent only with samples $p(a)$ and compute $p(b \mid a)$ for every sample

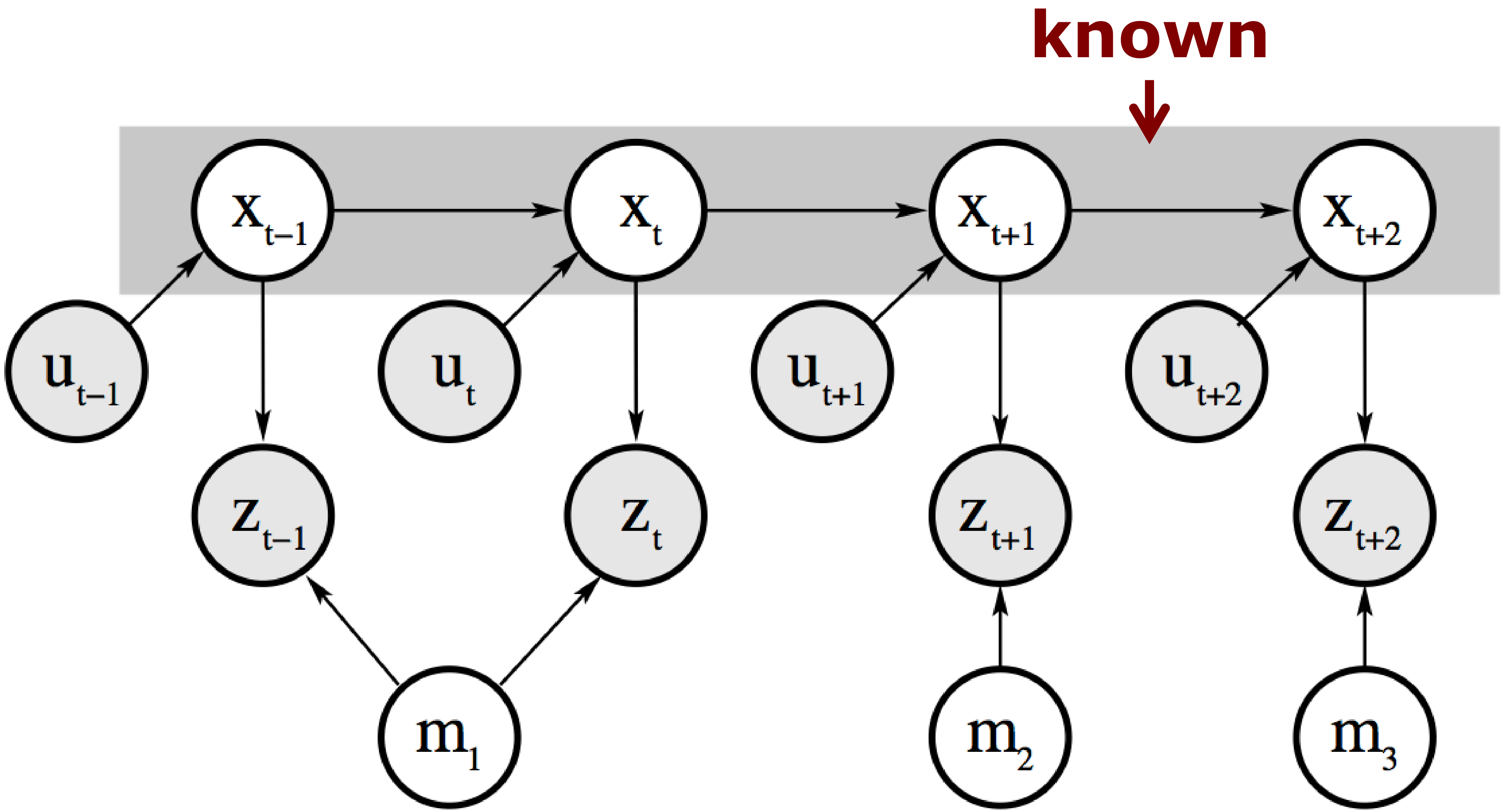
- It comes from the Rao-Blackwell theorem

Revisit the Graphical Model



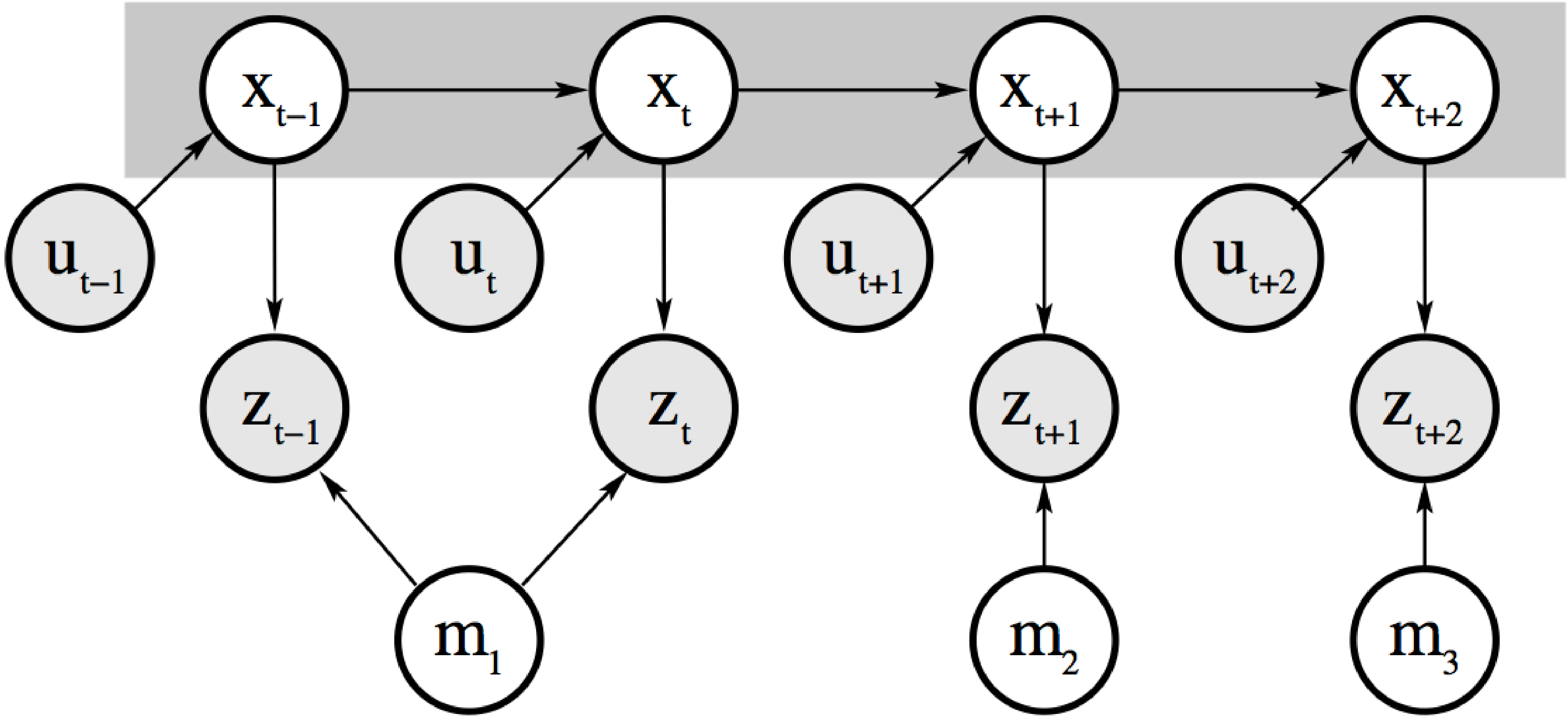
Courtesy: Thrun, Burgard, Fox

Revisit the Graphical Model



Courtesy: Thrun, Burgard, Fox

Landmarks are Conditionally Independent Given the Poses



Landmark variables are all disconnected (i.e. independent) given the robot's path

Factored Posterior

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

Robot path posterior (localization problem)

Conditionally independent landmark positions

Rao-Blackwellization for SLAM

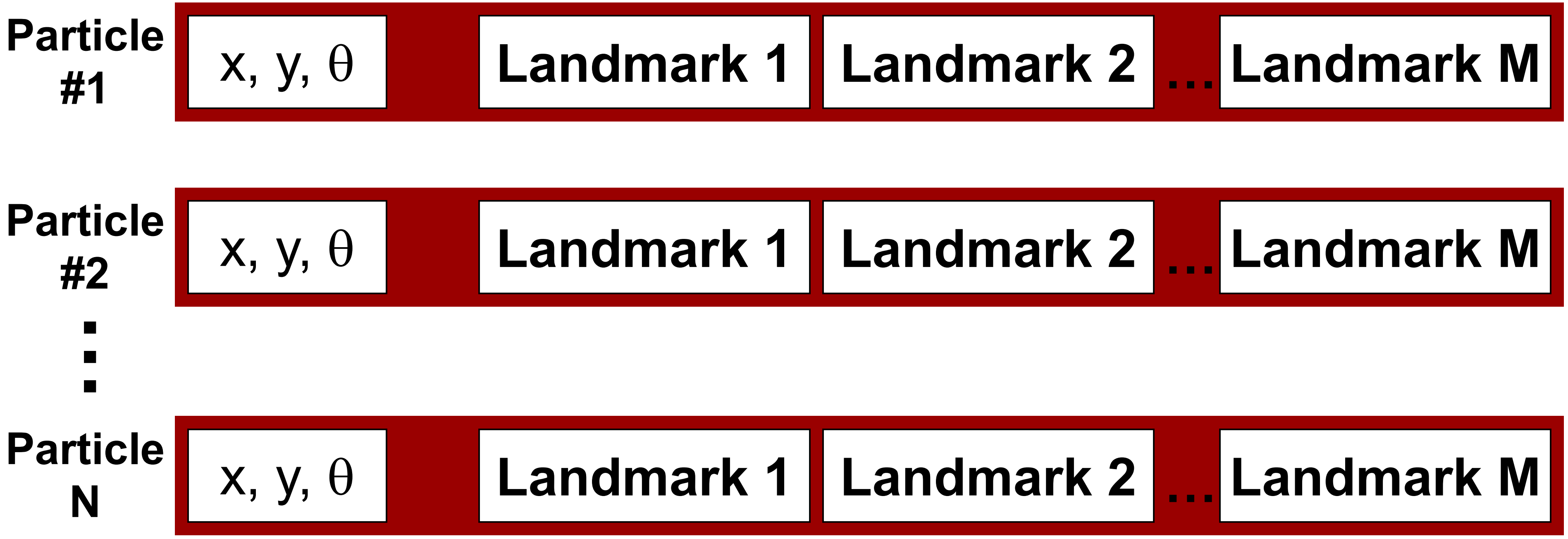
$$p(x_{1:t}, l_{1:m} \mid z_{1:t}, u_{0:t-1}) =$$

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

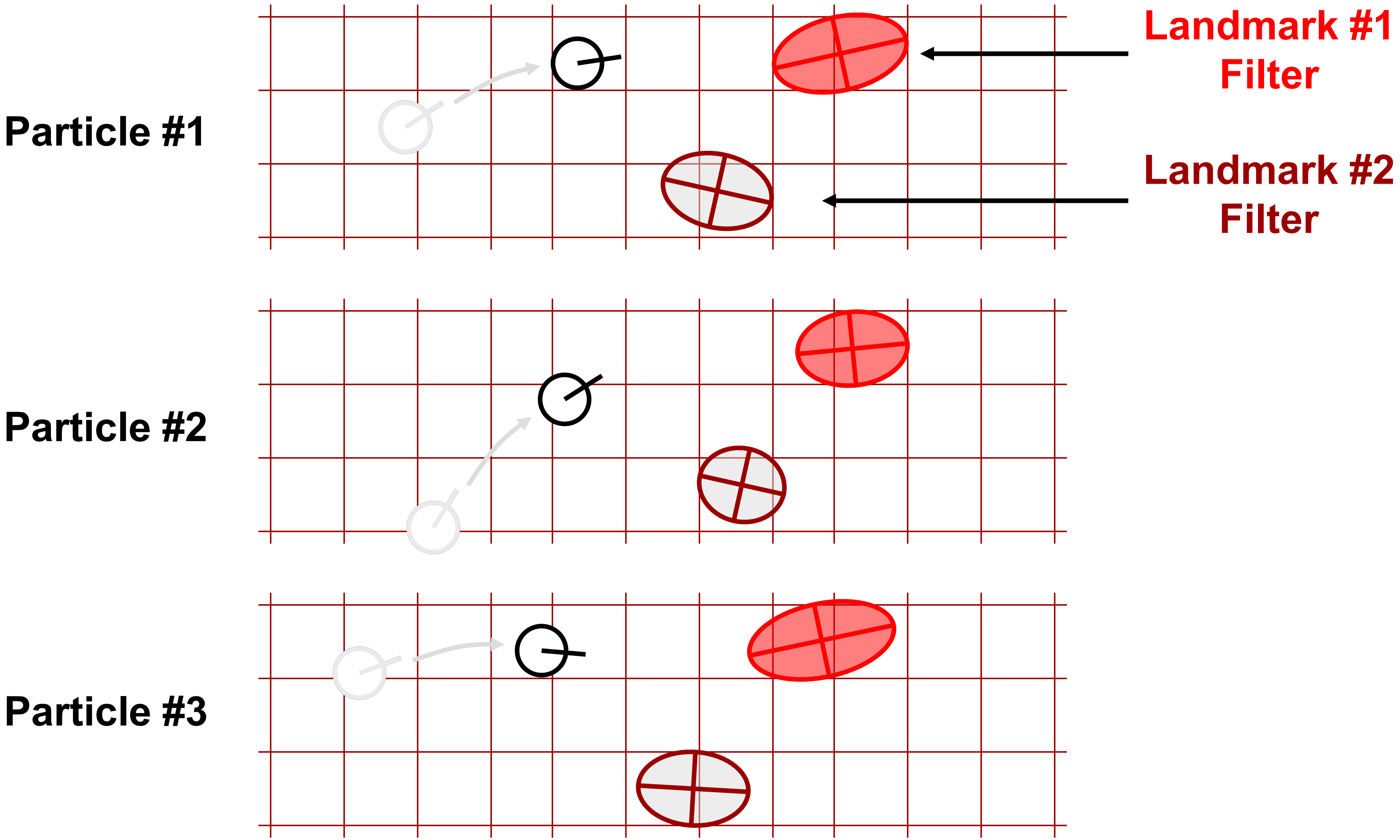
- Given that the second term can be computed efficiently, particle filtering becomes possible!

FastSLAM

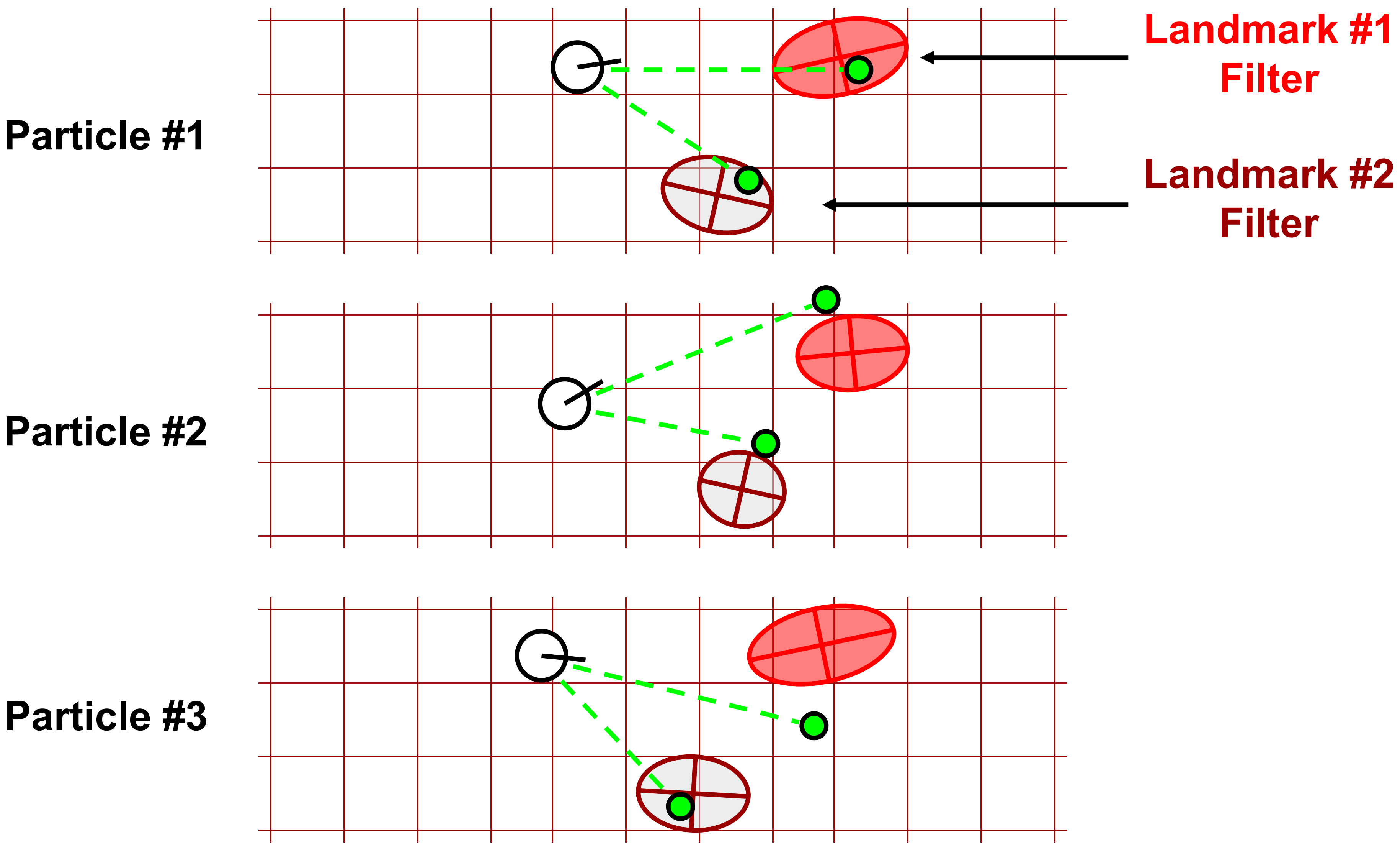
- Rao-Blackwellized particle filtering based on landmarks [Montemerlo et al., 2002]
- Each landmark is represented by a 2x2 Extended Kalman Filter (EKF)
- Each particle therefore has to maintain *M* EKFs



FastSLAM – Action Update



FastSLAM – Sensor Update



FastSLAM – Sensor Update

Particle #1

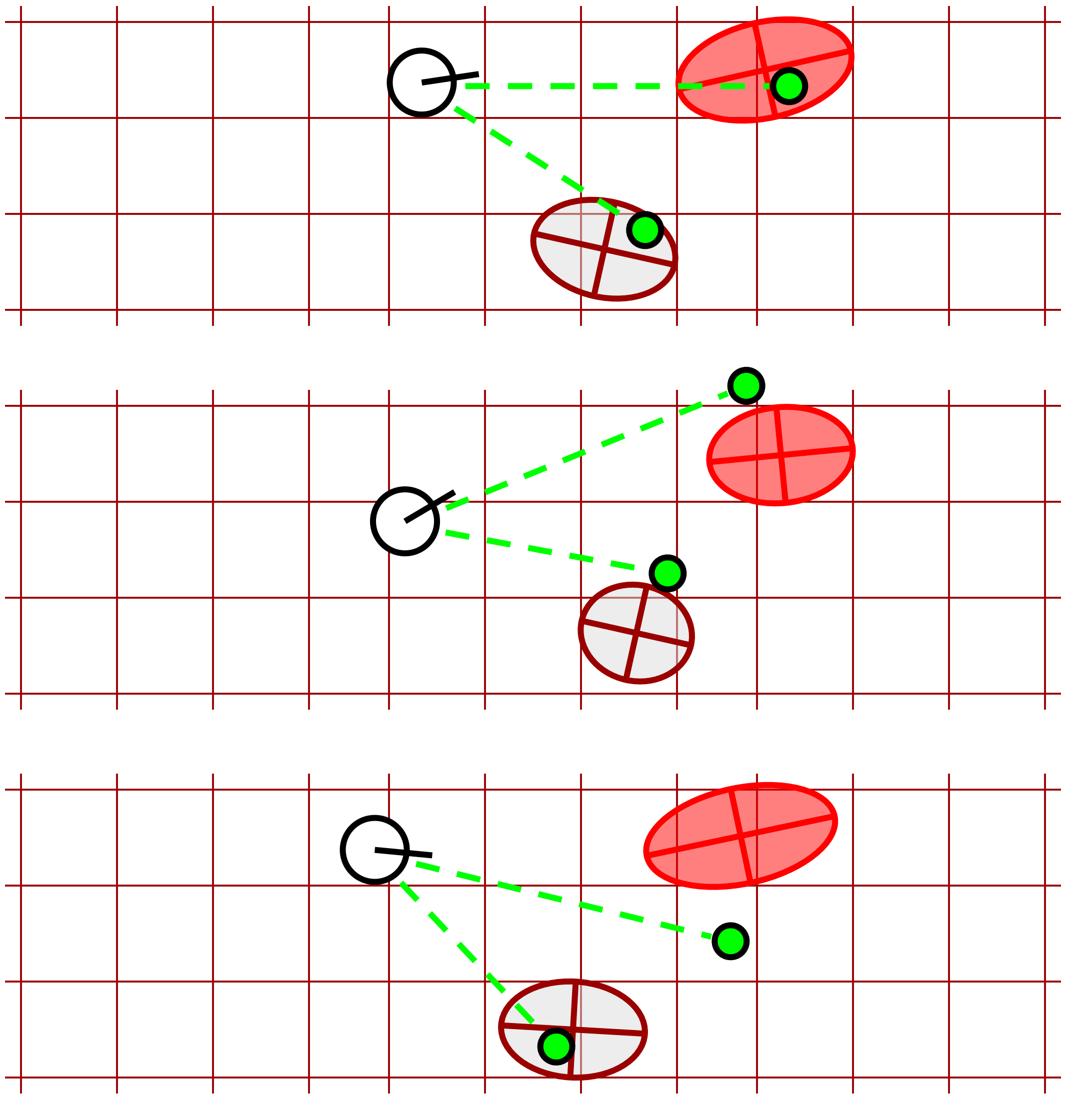
Weight = 0.8

Particle #2

Weight = 0.4

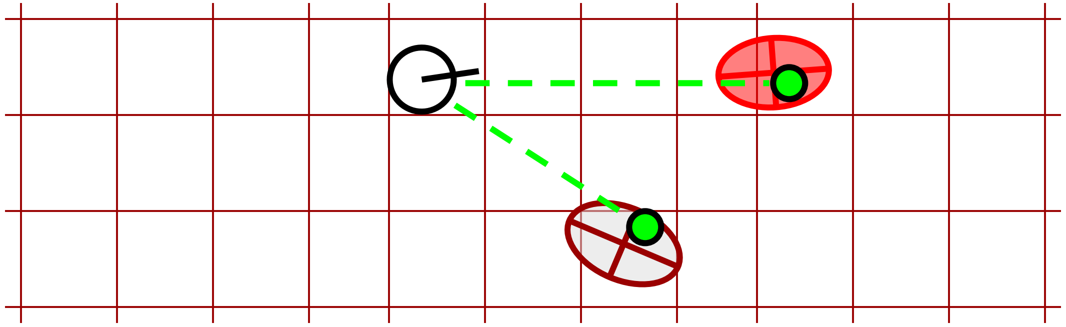
Particle #3

Weight = 0.1



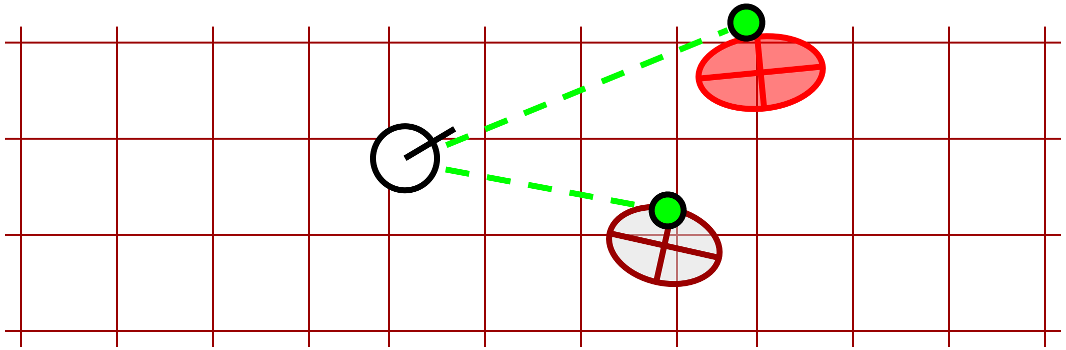
FastSLAM – Sensor Update

Particle #1



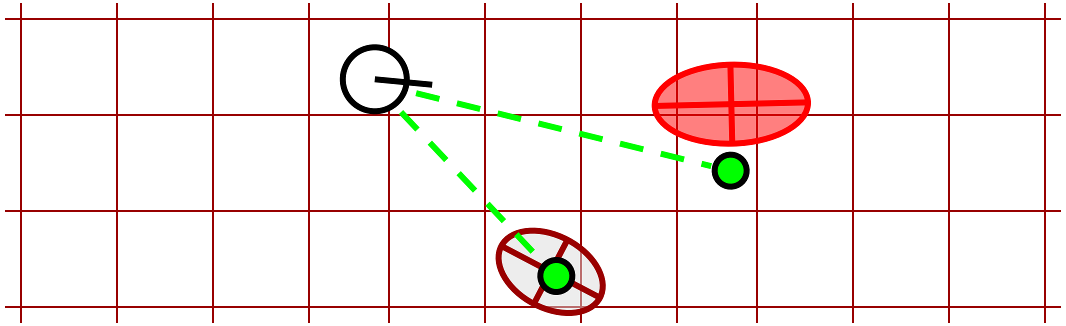
Update map
of particle #1

Particle #2



Update map
of particle #2

Particle #3



Update map
of particle #3

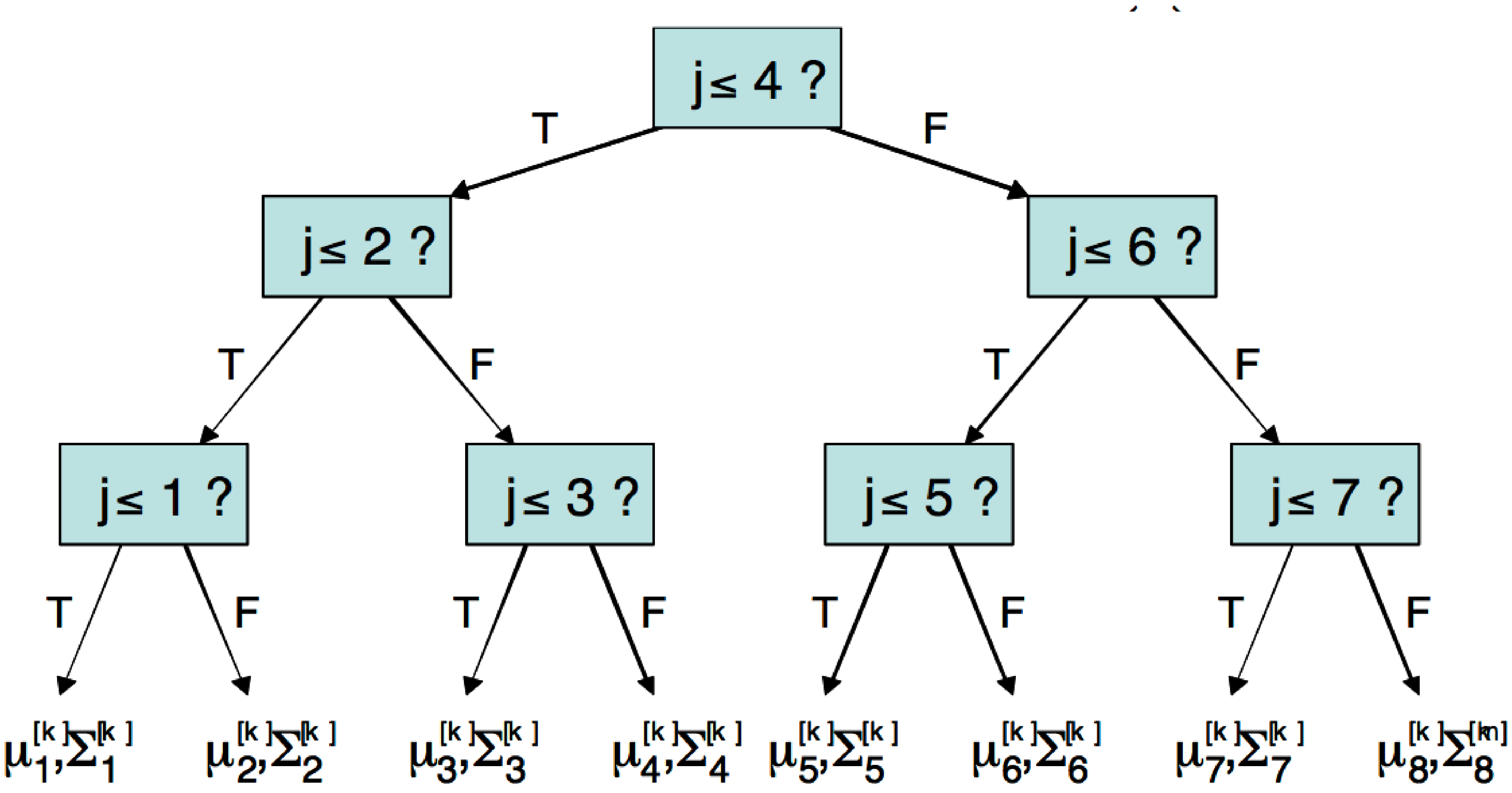
FastSLAM Complexity – Naive

- Update robot particles based on the control $\mathcal{O}(N)$
- Incorporate an observation into the Kalman filters (given the data association) $\mathcal{O}(N)$
- Resample particle set $\mathcal{O}(NM)$

$$\mathcal{O}(NM)$$

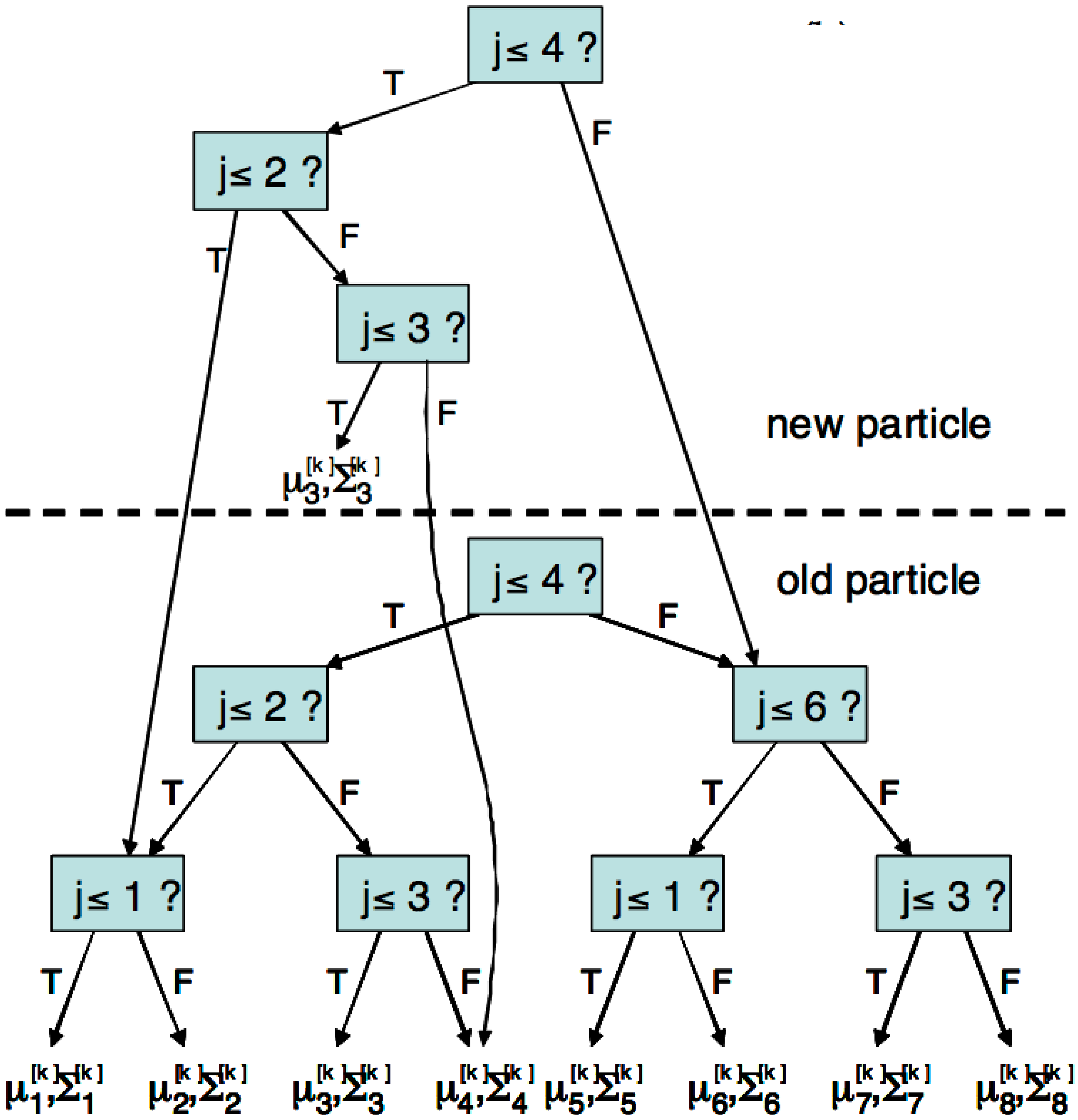
N = Number of particles
M = Number of map features

A Better Data Structure for FastSLAM



Courtesy: M. Montemerlo

A Better Data Structure for FastSLAM



FastSLAM Complexity

- Update robot particles based on the control $\mathcal{O}(N)$

- Incorporate an observation into the Kalman filters (given the data association) $\mathcal{O}(N \log M)$

- Resample particle set $\mathcal{O}(N \log M)$

$$\overline{\mathcal{O}(N \log M)}$$

$$\mathcal{O}(N \log M)$$

N = Number of particles
M = Number of map features