FastSlam and variants

D.A. Forsyth, UIUC

(with a lot of help from borrowed slides….!)

Particle filters

- We've seen basic particle filters
- Can deal with
  - non-linear state updates
  - non-linear measurements
- Dislike
  - high dimensions

Localization vs. SLAM

- A particle filter can be used to solve both problems
- Localization: state space $<x, y, \theta>$ ← Easy for pf
- SLAM: state space $<x, y, \theta, map>$ ← Bad news for pf
  - 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!

From Burgard et al slides

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

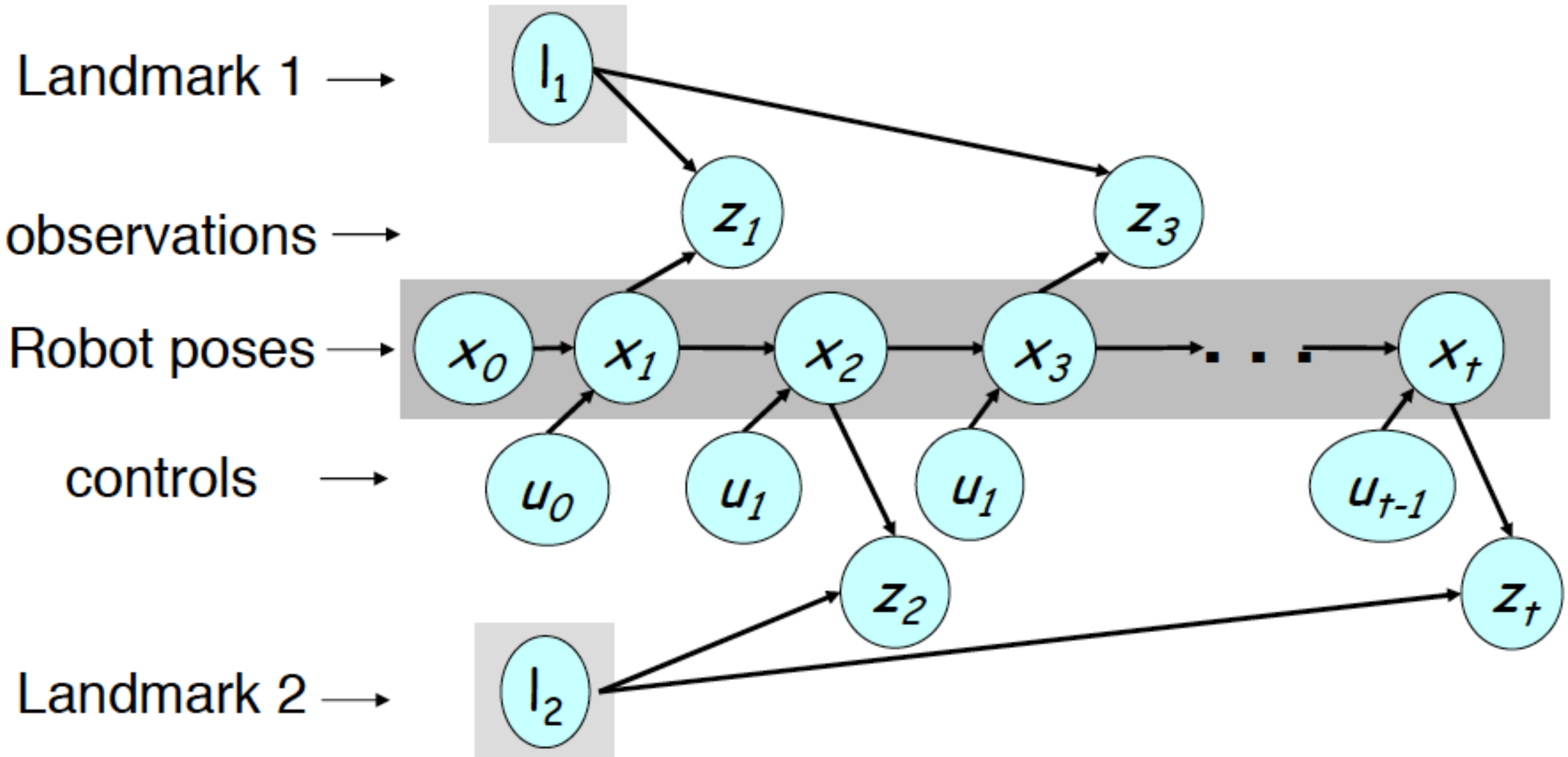
landmark positions

Does this help to solve the problem?

Factorization first introduced by Murphy in 1999

From Burgard et al slides

Mapping using Landmarks



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

From Burgard et al slides

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

From Burgard et al slides

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!

The factorization isn't Rao-Blackwellization
It's the consequences that are. What's important
here is that estimating p(l|x, z) is very well behaved;
you can bung these terms in an Extended Kalman filter

From Burgard et al slides

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

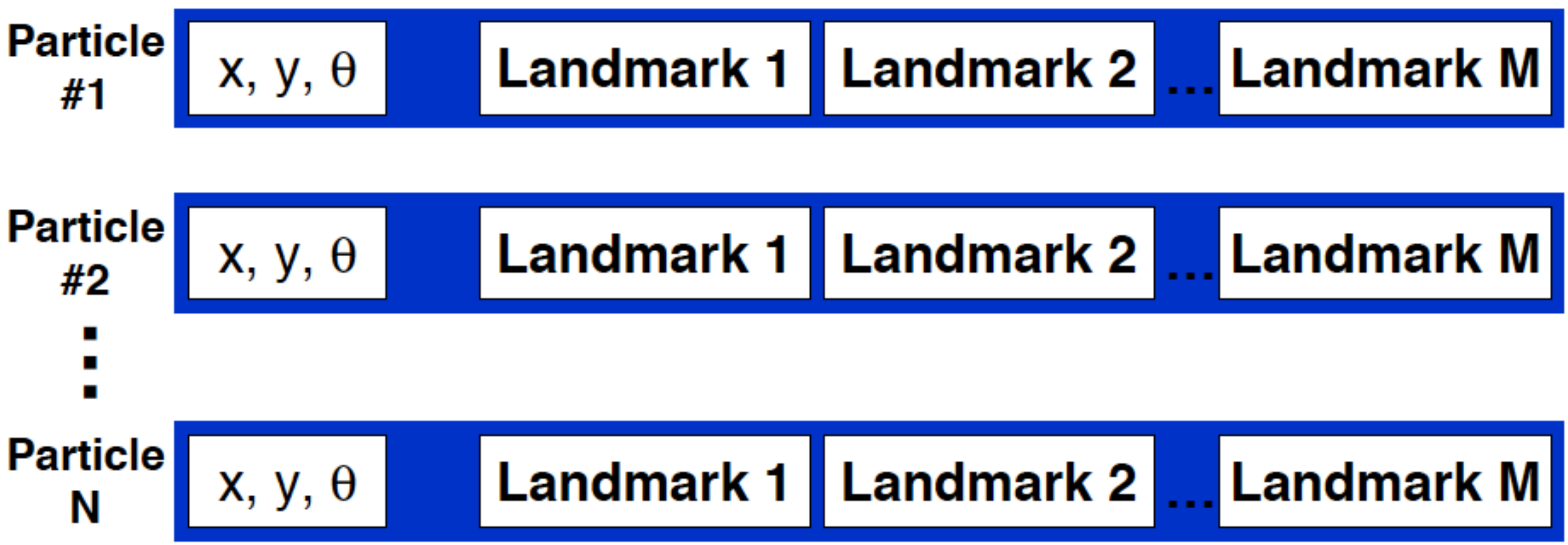
$$p(x_{1:t} \mid z_{1:t}, u_{0:t-1}) \cdot \prod_{i=1}^{M} p(l_i \mid x_{1:t}, z_{1:t})$$

Particle filter represents this distribution

Each of these terms is handled by an EKF
FOR EACH PARTICLE

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



From Burgard et al slides

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

N = Number of particles
M = Number of map features

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

From Burgard et al slides

Key Steps of FastSLAM 1.0

- Extend the path posterior by sampling a new pose for each sample

$$x_t^{[k]} \sim p(x_t \mid x_{t-1}^{[k]}, u_t)$$

- Compute particle weight

exp. observation

$$w^{[k]} = |2\pi Q|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(z_t - \hat{z}^{[k]})^T Q^{-1} (z_t - \hat{z}^{[k]})\right\}$$

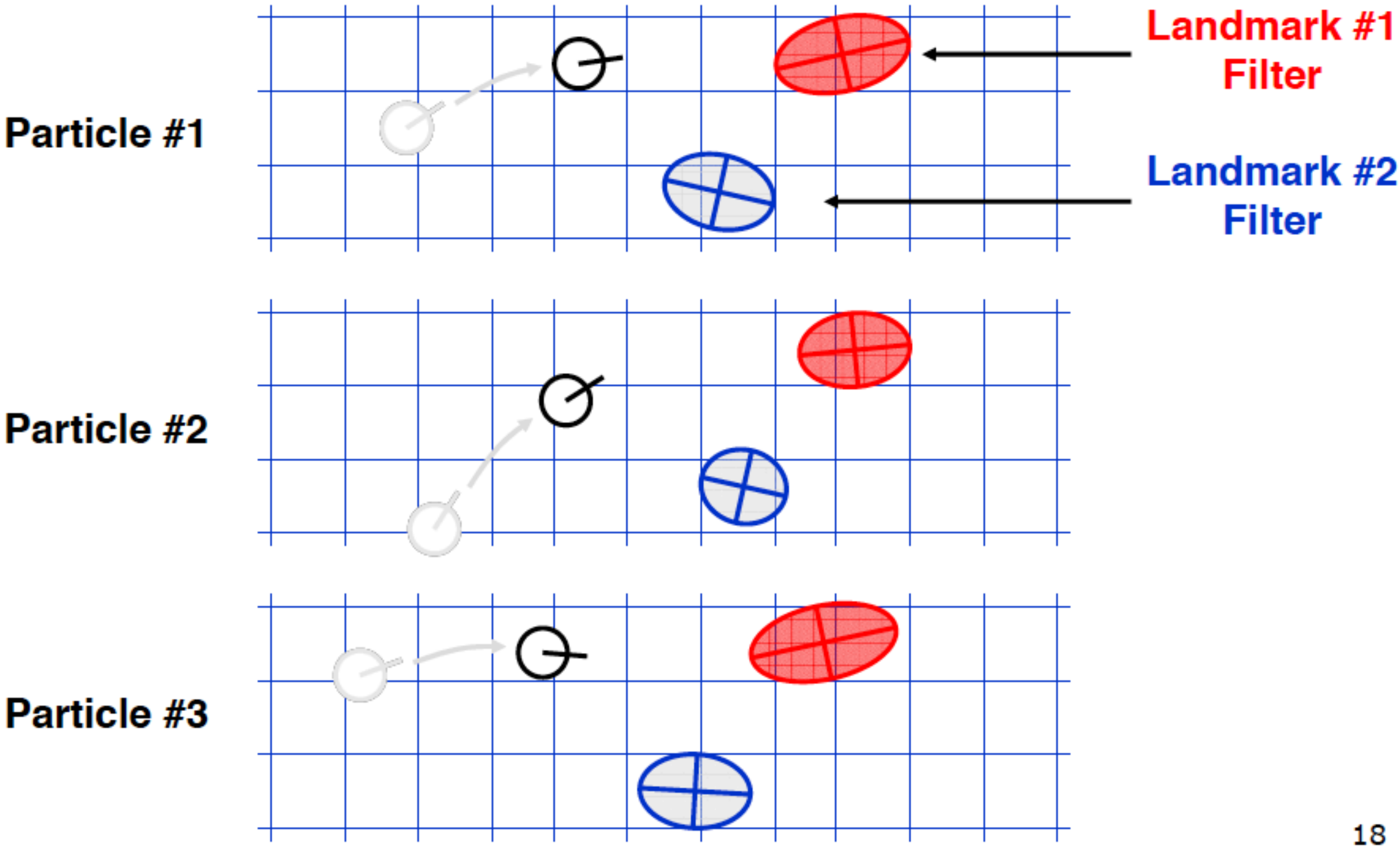
innovation covariance

- Update belief of observed landmarks (EKF update rule)
- Resample

Courtesy: C. Stachniss

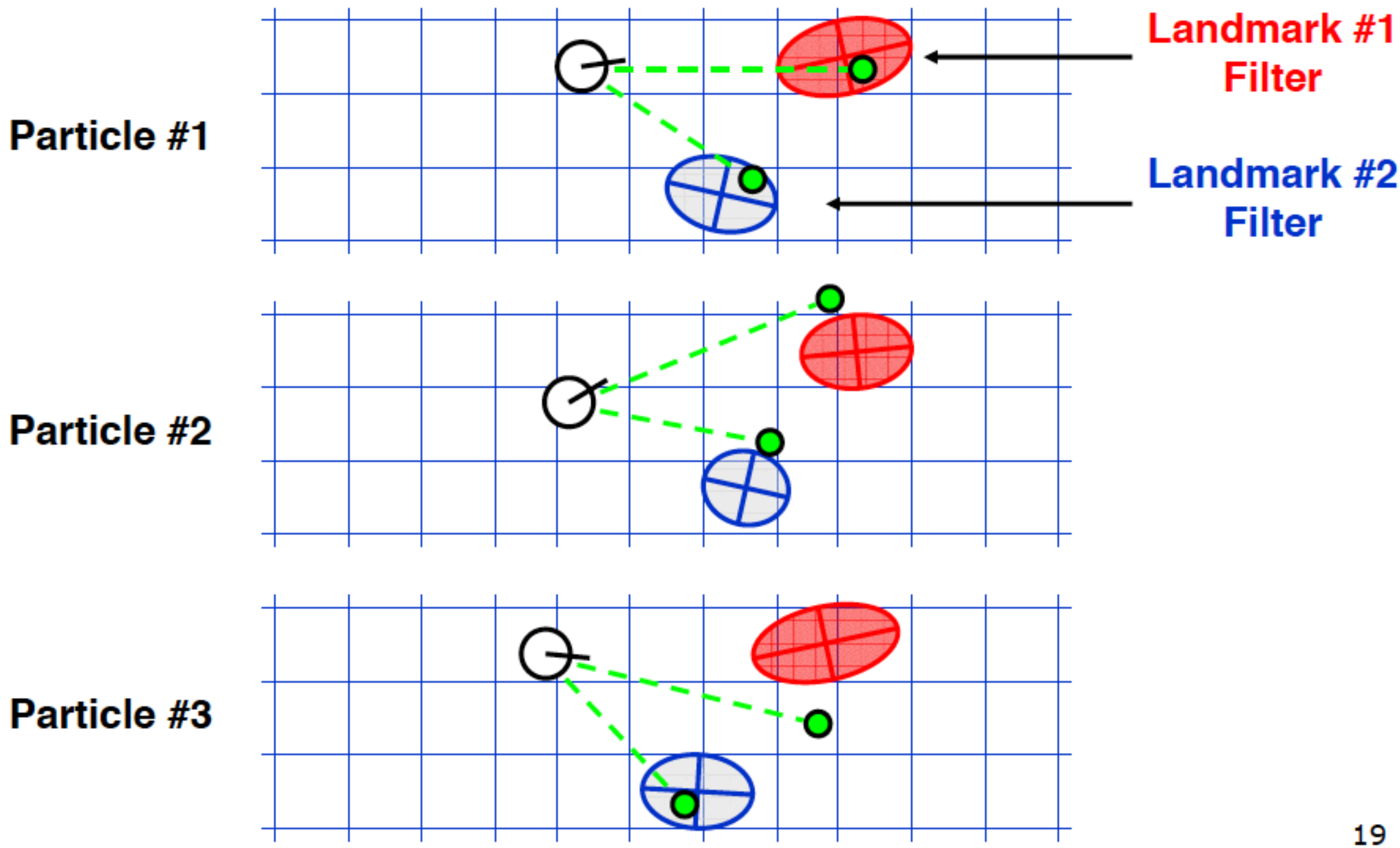
Unknown author, UW Slides

FastSLAM – Action Update



From Burgard et al slides

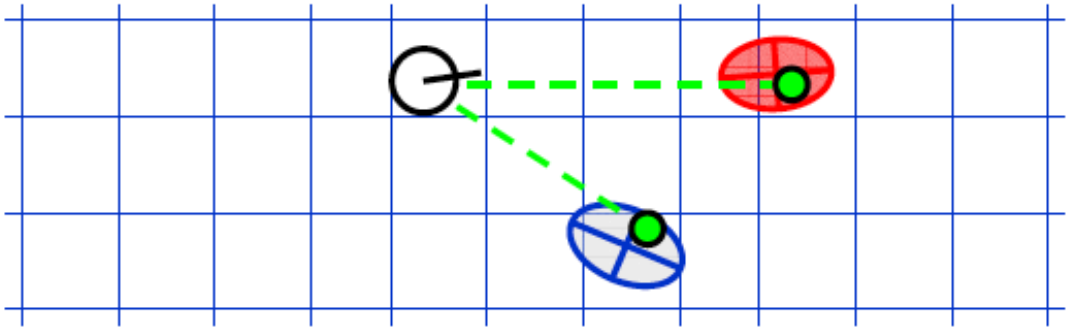
FastSLAM – Sensor Update



From Burgard et al slides

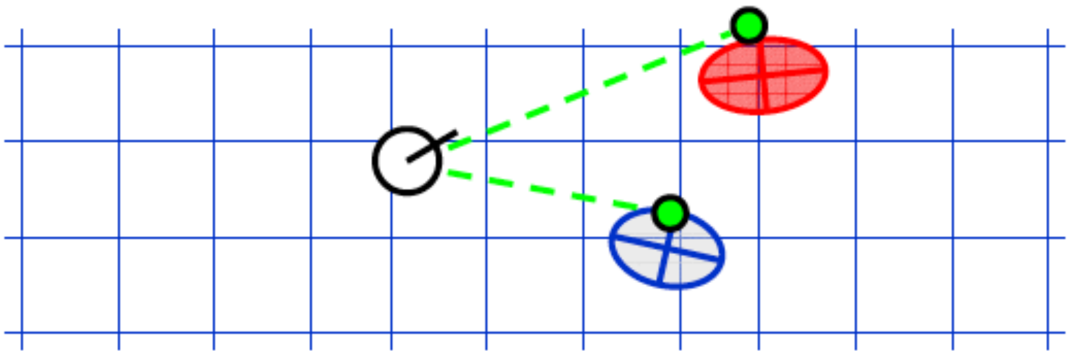
FastSLAM – Sensor Update

Particle #1



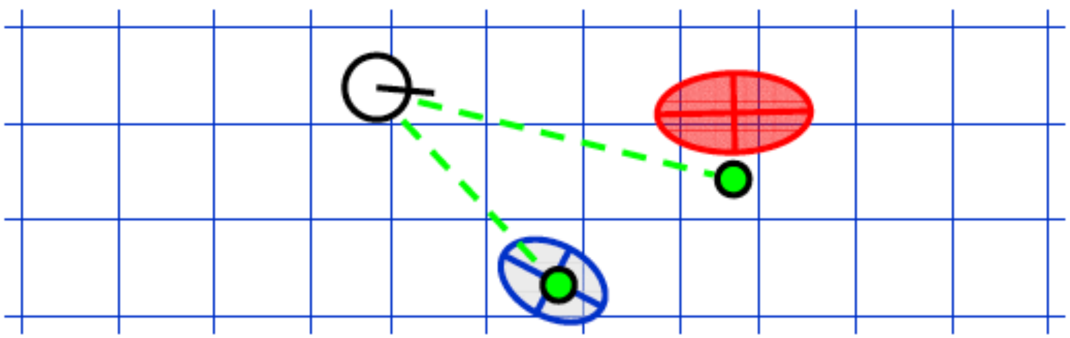
Weight = 0.8

Particle #2



Weight = 0.4

Particle #3



Weight = 0.1

From Burgard et al slides

Cum grano salis

Implementation Hint

- Alan Oursland has a Java implementation
  - http://www.oursland.net/projects/fastslam/
- He reports having a hard time getting it to work, until Dieter Fox helped him tune the Kalman Filter.
  - The observation covariance *R* must be very large, so observations can match far-away landmarks.
- This is an example of how personal experience is important to replicating ideas.

From Burgard et al slides

FastSLAM Complexity

Cum grano salis

- 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

N = Number of particles
M = Number of map features

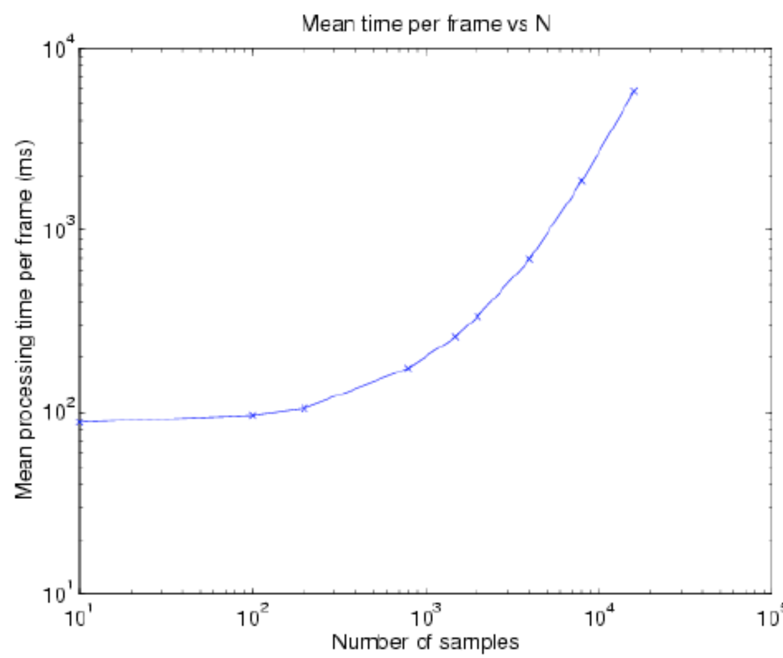
$O(N \cdot \log(M))$
Log time per particle

From Burgard et al slides

The grain of salt..

Another Practical Note

- In theory, FastSLAM should scale well: $O(KN \log M)$, where
  - N is the number of particles
  - K is the number of landmarks observed
  - M is the number of landmarks in the map
- But, in practice . . .



Robert Sim, http://www.cs.ubc.ca/~simra/lci/fastslam/nonlinear.html

From Kuipers' slides

More salt….

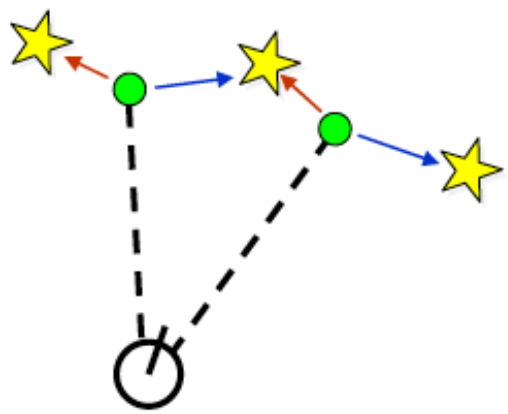
Why doesn't FastSLAM scale?

- At each frame:
  - *K* SIFT features are added to the kd-tree, and
  - *NK* landmarks are added to the FastSLAM tree.
- Memory fragmentation:
  - In time, nearby SIFT features are separated in memory, so CPU cache miss rate goes up.
  - For large maps, page fault rate will also increase.
- So the problem is the memory hierarchy, due to failure of locality.

From Kuipers' slides

Data Association Problem

- Which observation belongs to which landmark?

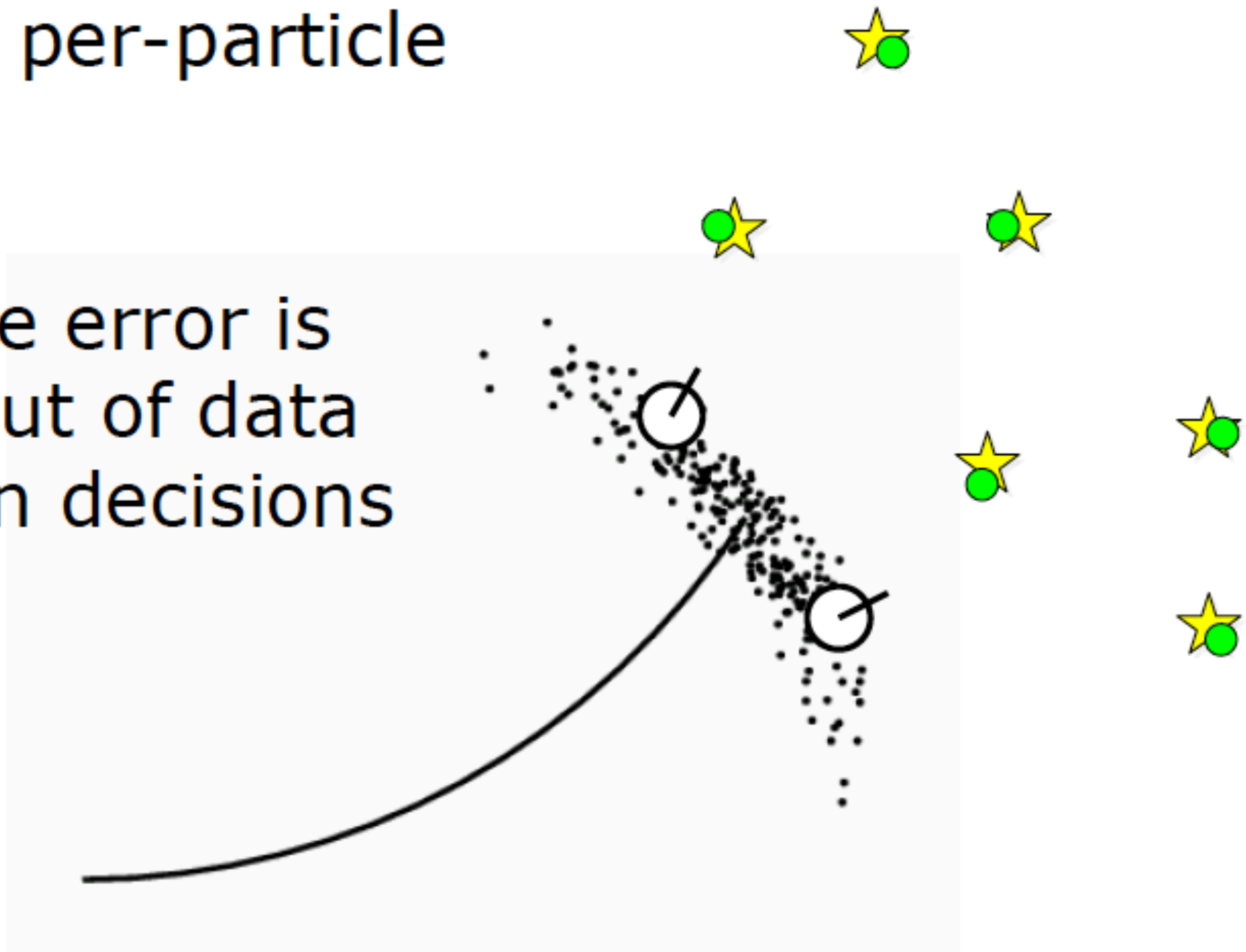


- A robust SLAM must consider possible data associations
- Potential data associations depend also on the pose of the robot

From Burgard et al slides

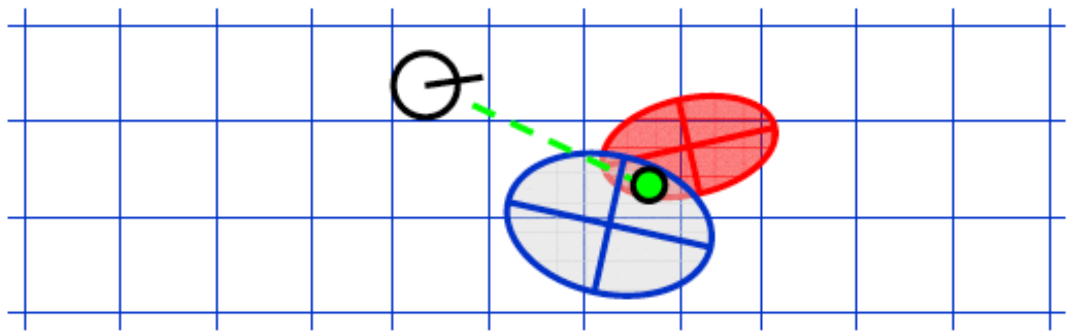
Multi-Hypothesis Data Association

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



From Burgard et al slides

Per-Particle Data Association



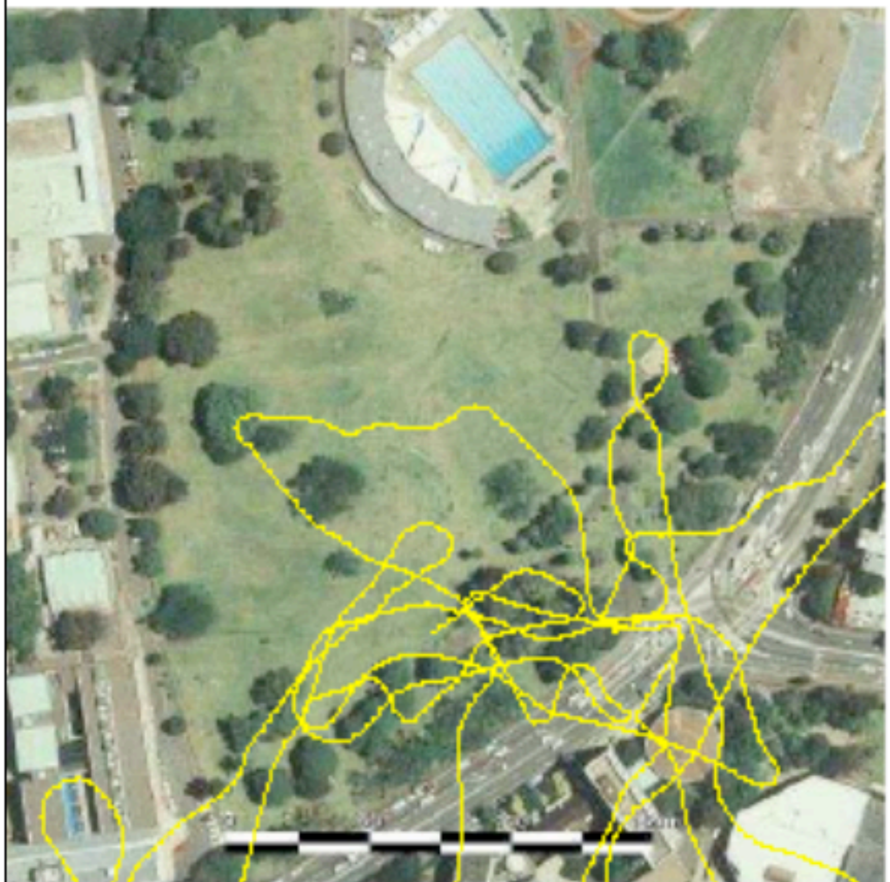
Was the observation generated by the red or the blue landmark?

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

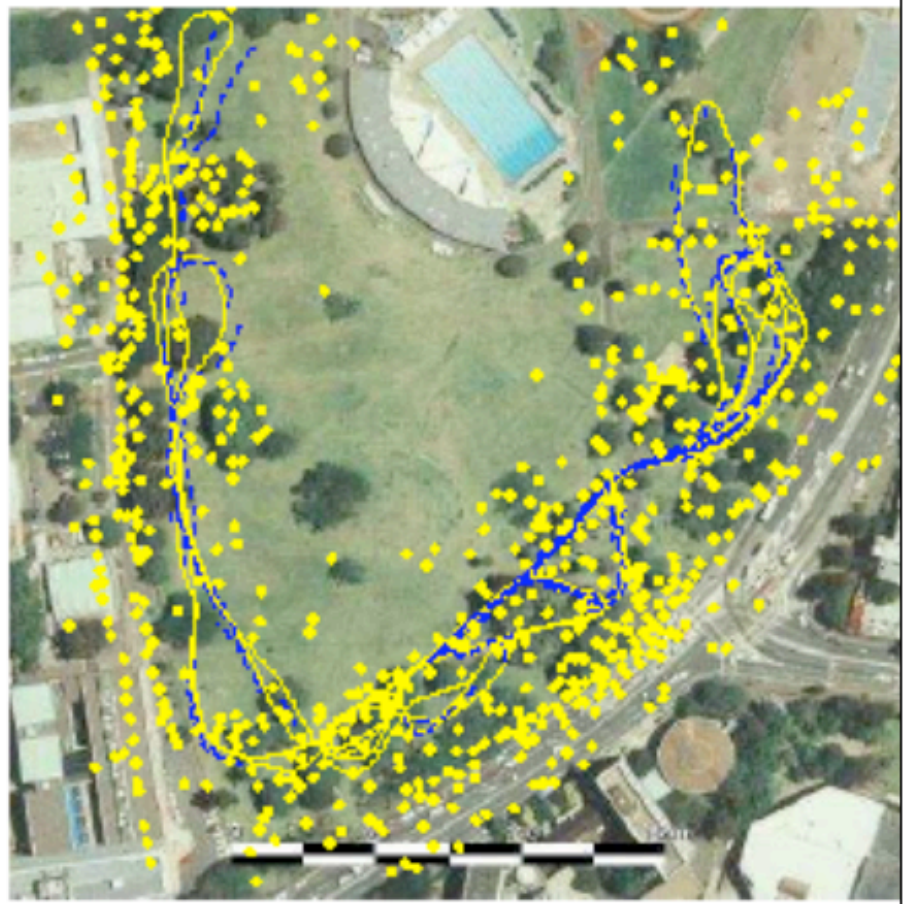
- Two options for per-particle data association
  - Pick the most probable match
  - Pick an random association weighted by the observation likelihoods
- If the probability is too low, generate a new landmark

From Burgard et al slides

FastSLAM in Victoria Park



with raw odometry

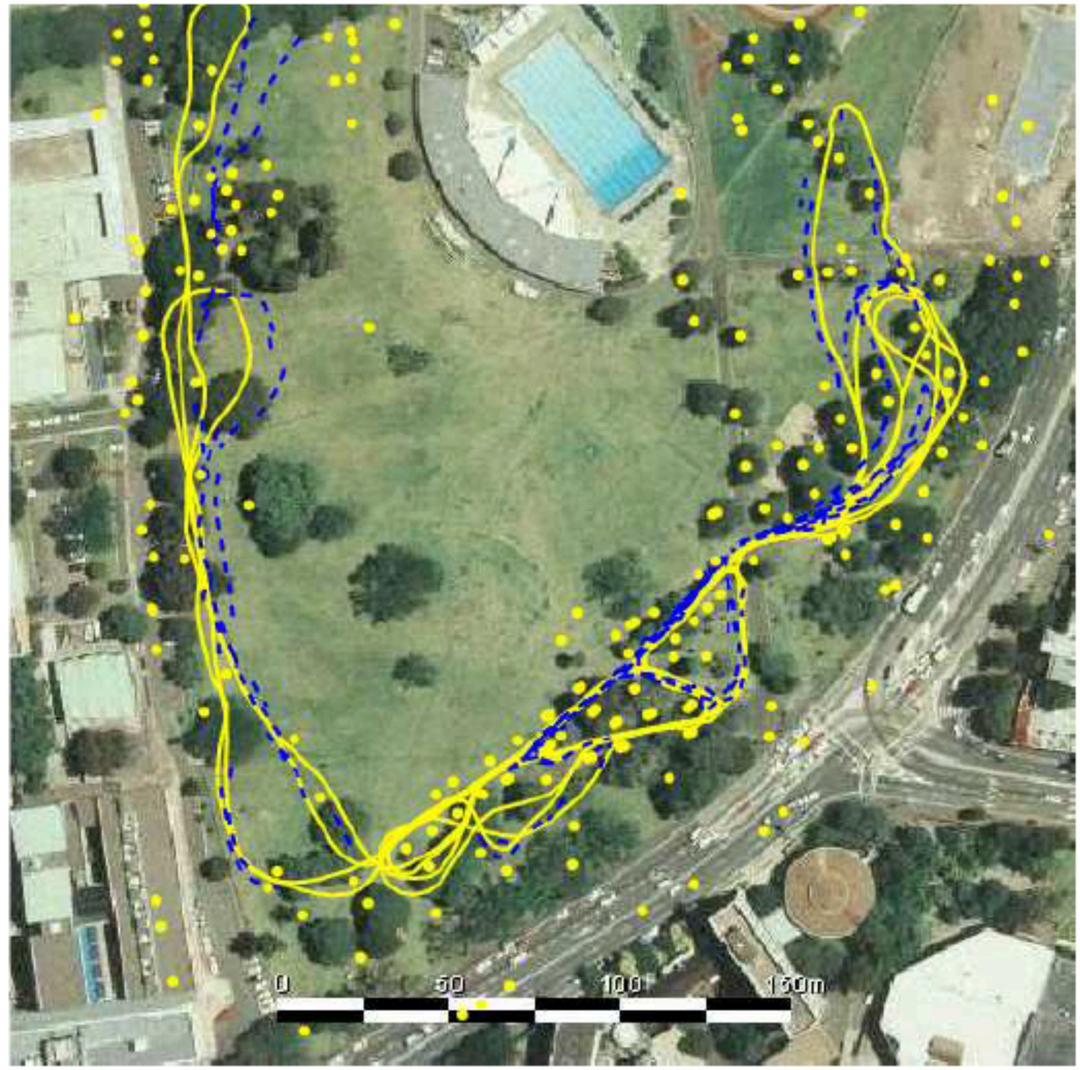


FastSLAM 2.0

From Kuipers' slides

Results – Victoria Park

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



Blue = GPS
Yellow = FastSLAM

Dataset courtesy of University of Sydney

From Burgard et al slides

FastSLAM 1.0

- FastSLAM 1.0 uses the motion model as the proposal distribution

$$x_t^{[k]} \sim p(x_t \mid x_{t-1}^{[k]}, u_t)$$

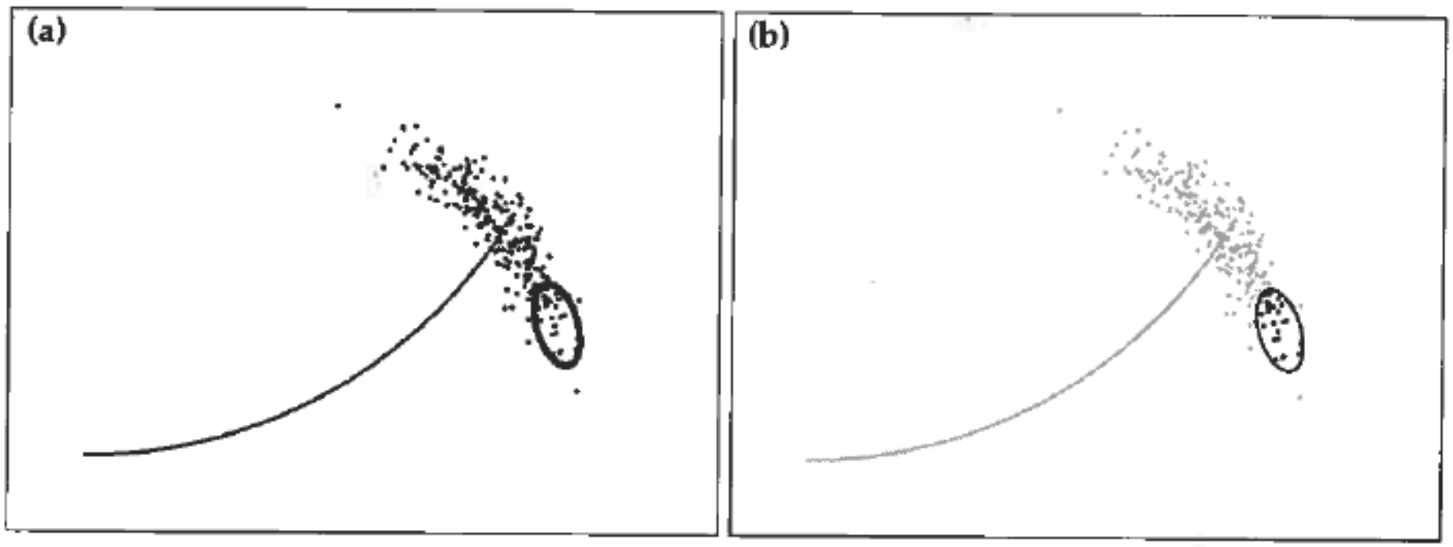
- **Is there a better distribution to sample from?**

[Montemerlo et al., 2002]

Courtesy: C. Stachniss

Weakness of FastSLAM 1.0

- Proposal Distribution
- Importance weighting



Unknown author, UW Slides

FastSLAM 1.0 to FastSLAM 2.0

- FastSLAM 1.0 uses the motion model as the proposal distribution

$$x_t^{[k]} \sim p(x_t \mid x_{t-1}^{[k]}, u_t)$$

- FastSLAM 2.0 **considers also the measurements during sampling**
- Especially useful if an accurate sensor is used (compared to the motion noise)

[Montemerlo et al., 2003]

Courtesy: C. Stachniss

Unknown author, UW Slides

FastSLAM 2.0 (Informally)

- FastSLAM 2.0 samples from

$$x_t^{[k]} \sim p(x_t \mid x_{1:t-1}^{[k]}, u_{1:t}, z_{1:t})$$

- Results in a more peaked proposal distribution
- Less particles are required
- More robust and accurate
- But more complex…

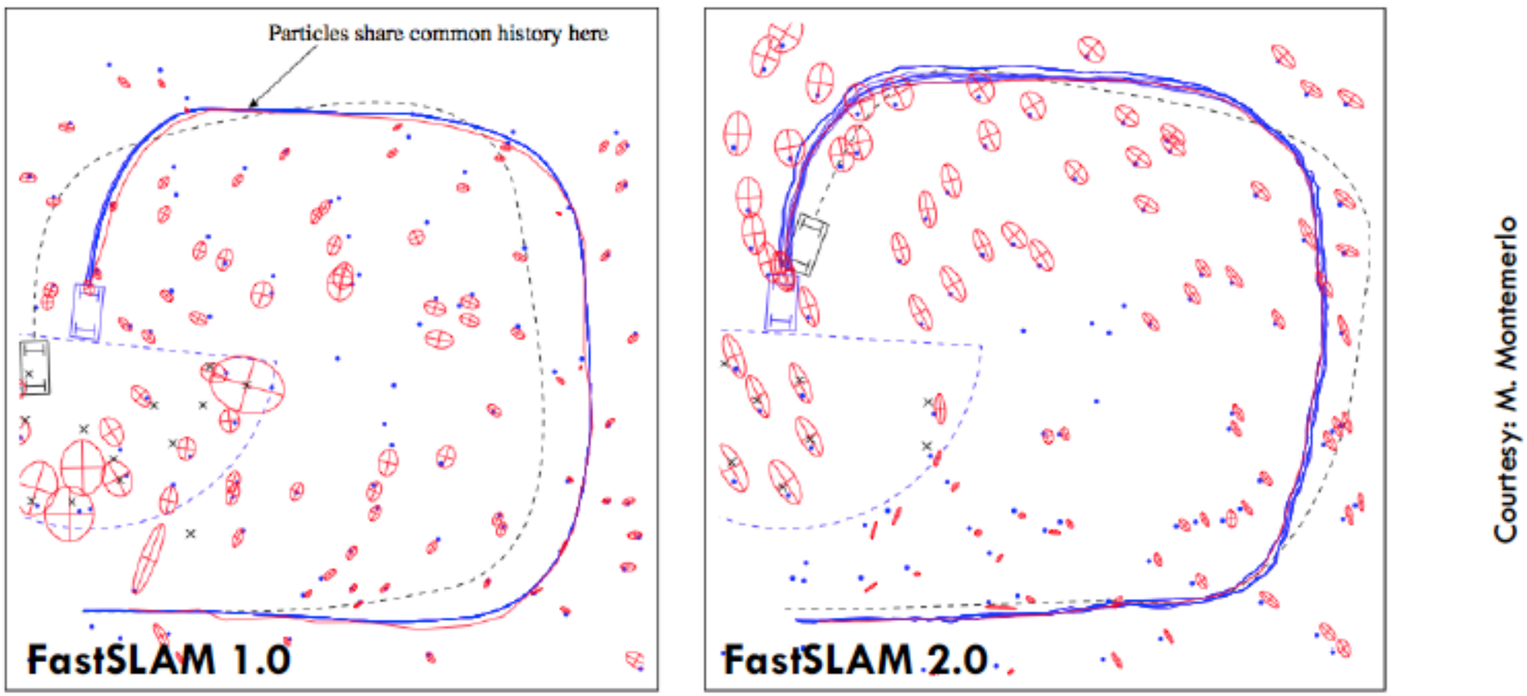
[Montemerlo et al., 2003]

Courtesy: C. Stachniss

Unknown author, UW Slides

FastSLAM Problems

- How to determine the sample size?
- Particle deprivation, especially when closing (multiple) loops



Unknown author, UW Slides