Introduction to Mobile Robotics

SLAM – Landmark-based FastSLAM

Wolfram Burgard, Cyrill Stachniss, Maren Bennewitz, Diego Tipaldi, Luciano Spinello

Partial slide courtesy of Mike Montemerlo

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

The SLAM Problem

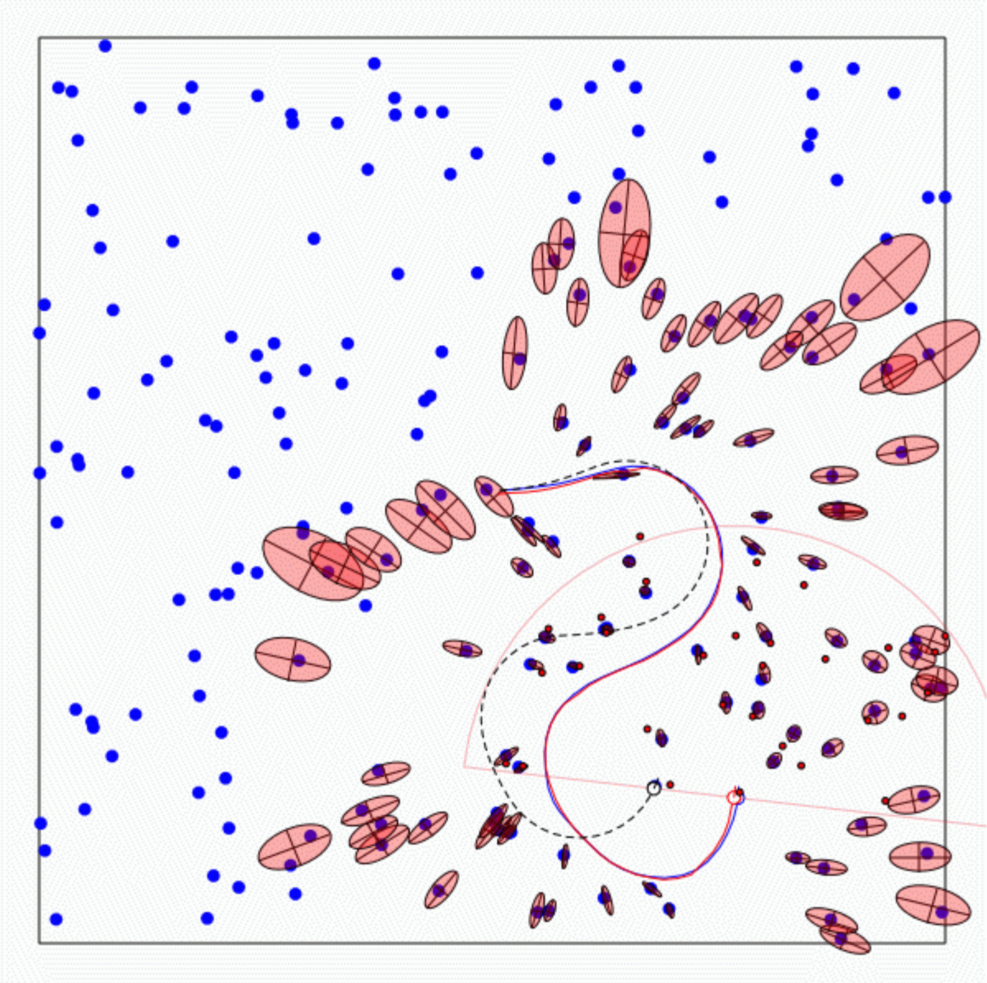
A robot moving though an unknown, static environment

Given:

- The robot’s controls
- Observations of nearby features

Estimate:

- Map of features
- Path of the robot

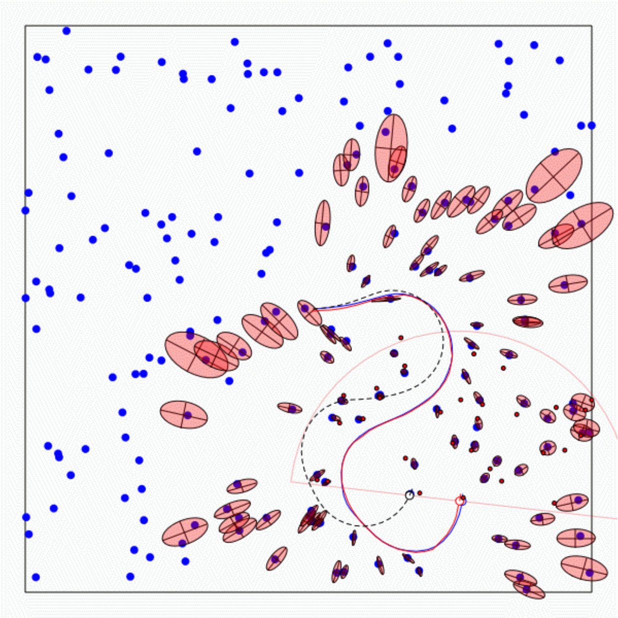


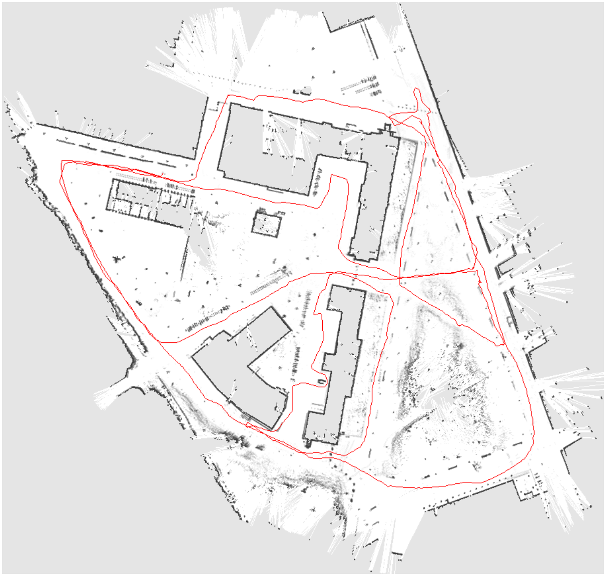
Map Representations

Typical models are:

- Feature maps
- Grid maps (occupancy or reflection probability maps)

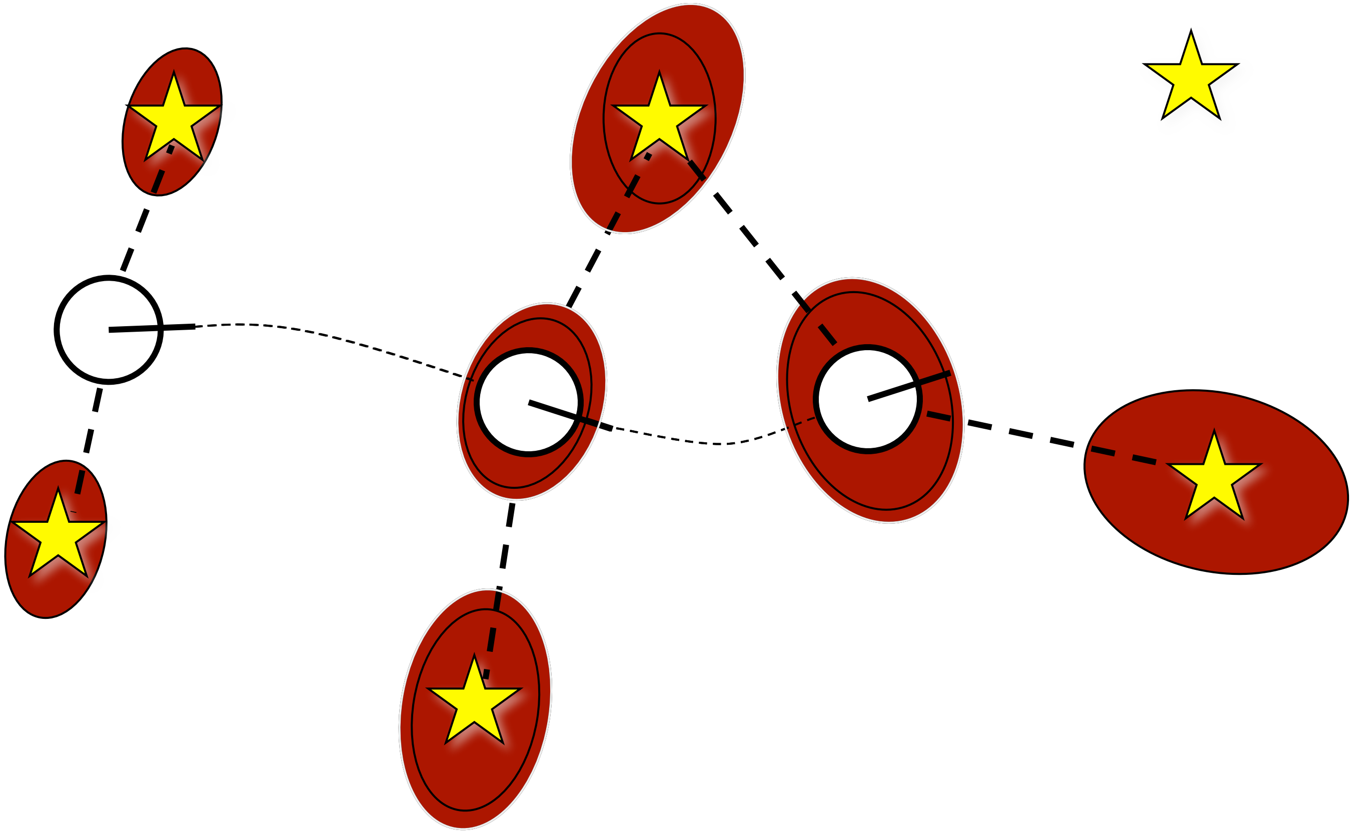
today





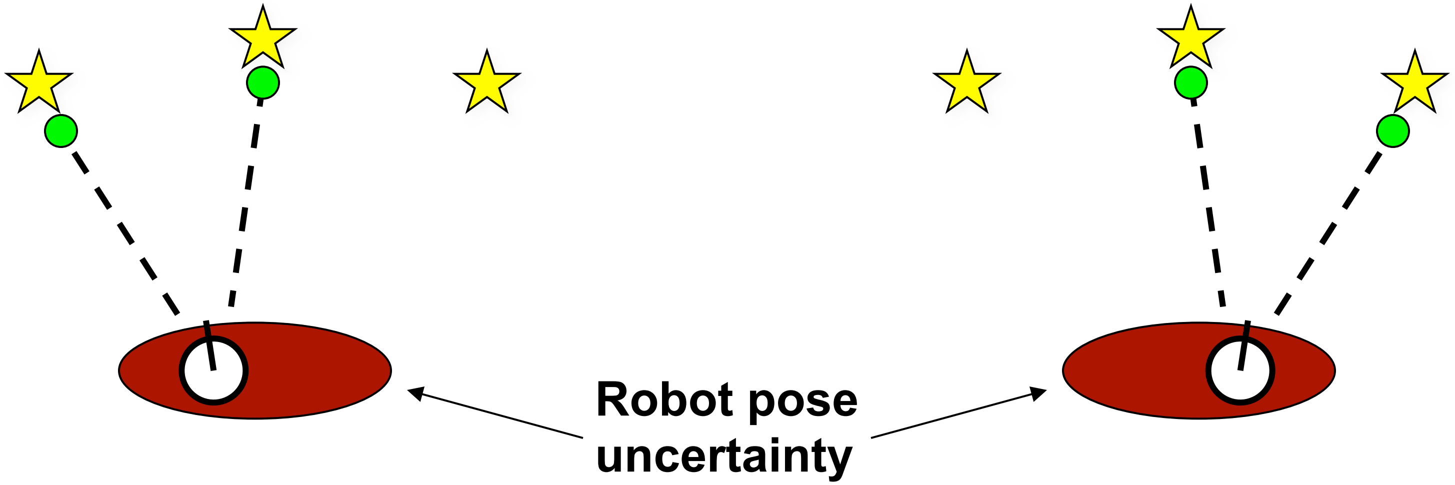
Why is SLAM a Hard Problem?

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



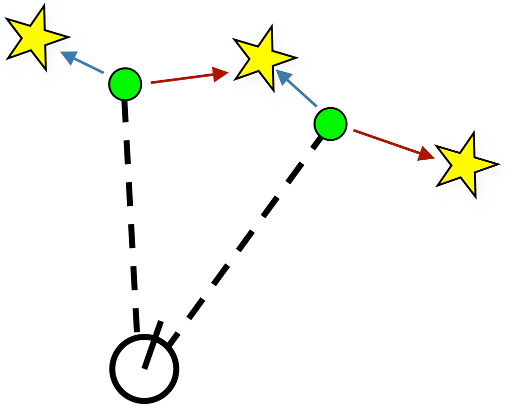
Robot path error correlates errors in the map

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”

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
  - Resampling

- 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 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.

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

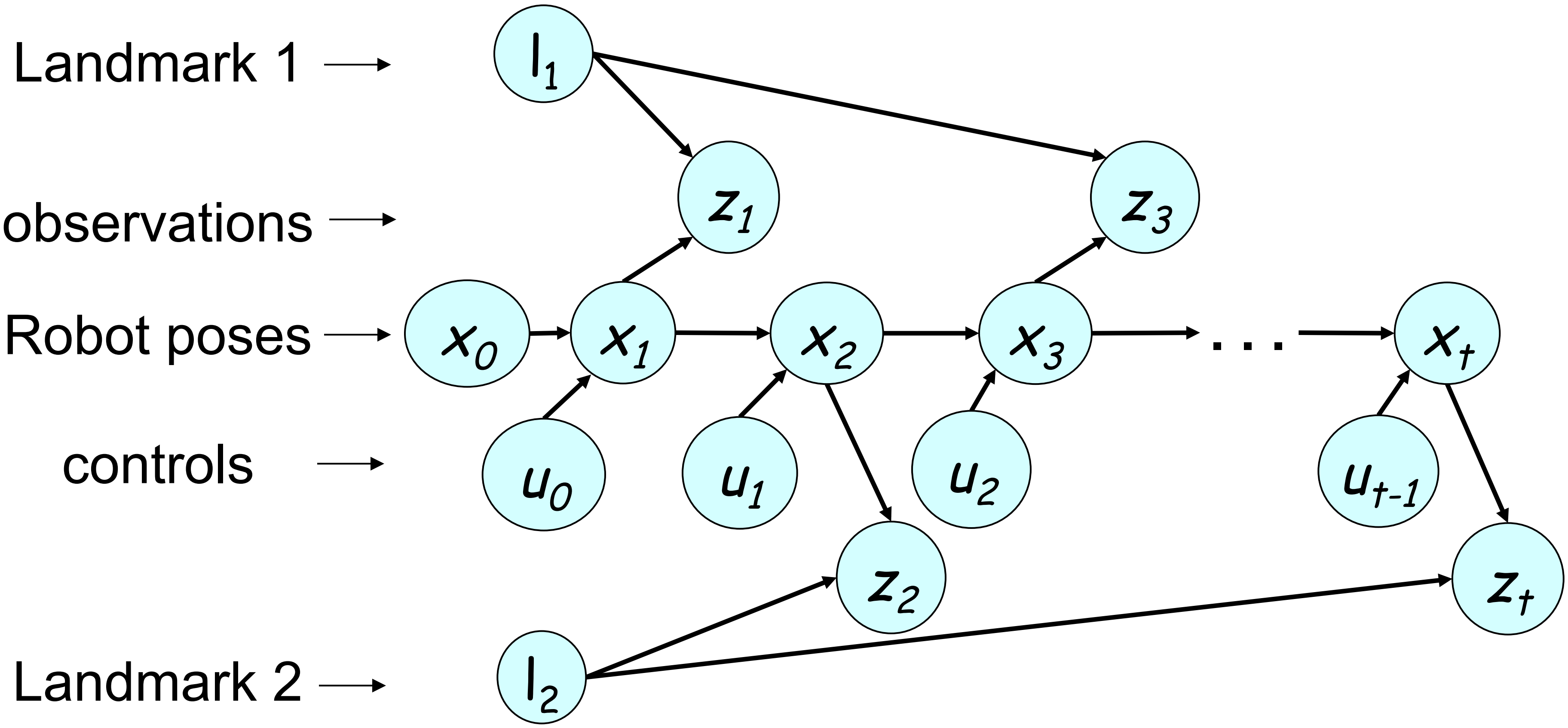
Robot path posterior

landmark positions

Does this help to solve the problem?

Factorization first introduced by Murphy in 1999

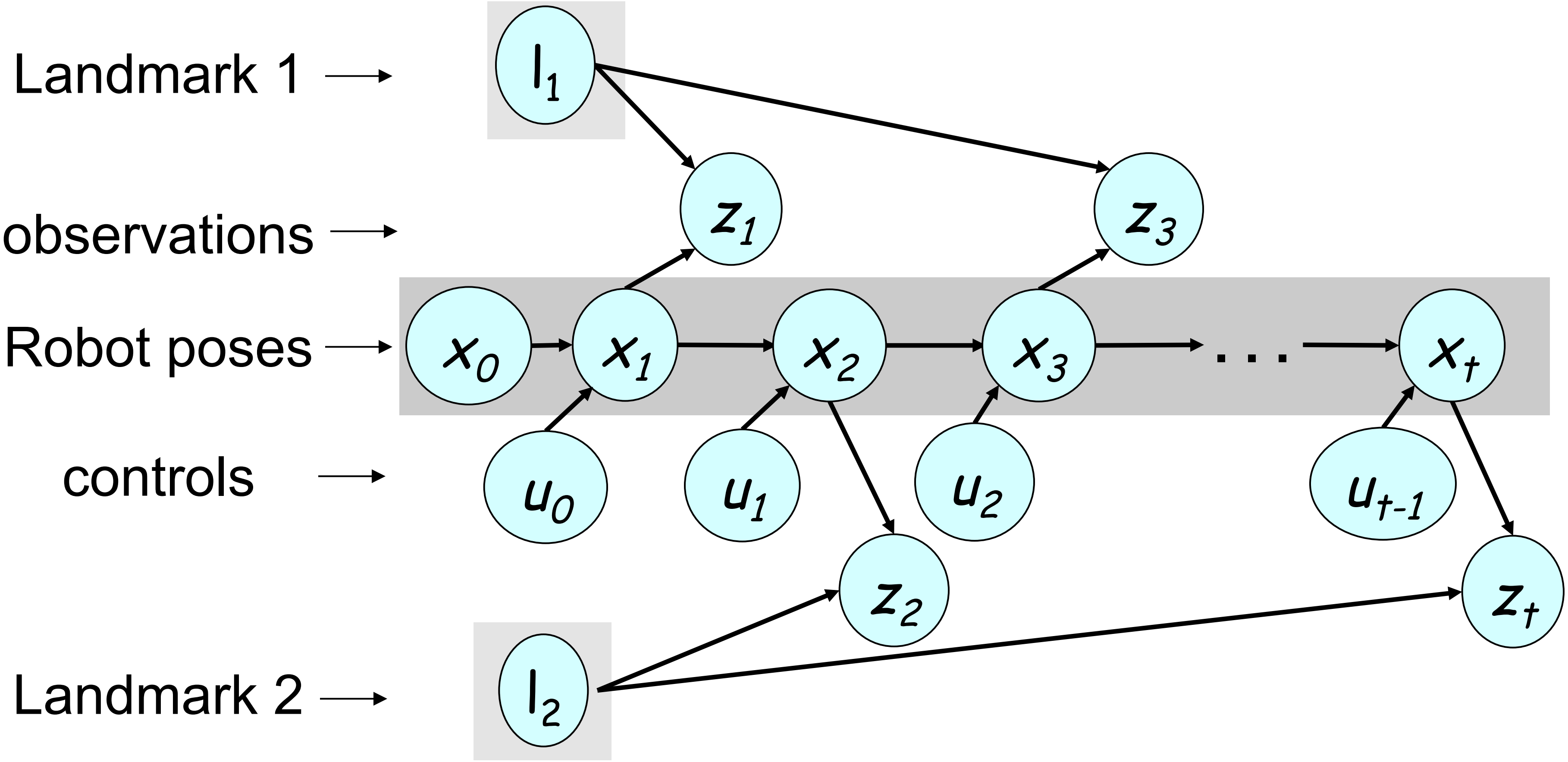
Mapping using Landmarks



Bayes Network and D-Separation (See AI or PGM course)

- X and Y are independent if d-separated by $\mathcal{V}$
- $\mathcal{V}$ d-separates X from Y if every undirected path between X and Y is **blocked** by $\mathcal{V}$
- A path is **blocked** by $\mathcal{V}$ if there is a node W on the graph such that either:
  - W has converging arrows along the path ($\rightarrow$ W $\leftarrow$) and neither W nor its descendants are observed (in $\mathcal{V}$), or
  - W does not have converging arrows along the path ($\rightarrow$ W $\rightarrow$ or $\leftarrow$ W $\rightarrow$) and W is observed (W $\in \mathcal{V}$).

Mapping using Landmarks



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

Factored Posterior

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

Robot path posterior (localization problem)

Conditionally independent landmark positions

Rao-Blackwellization

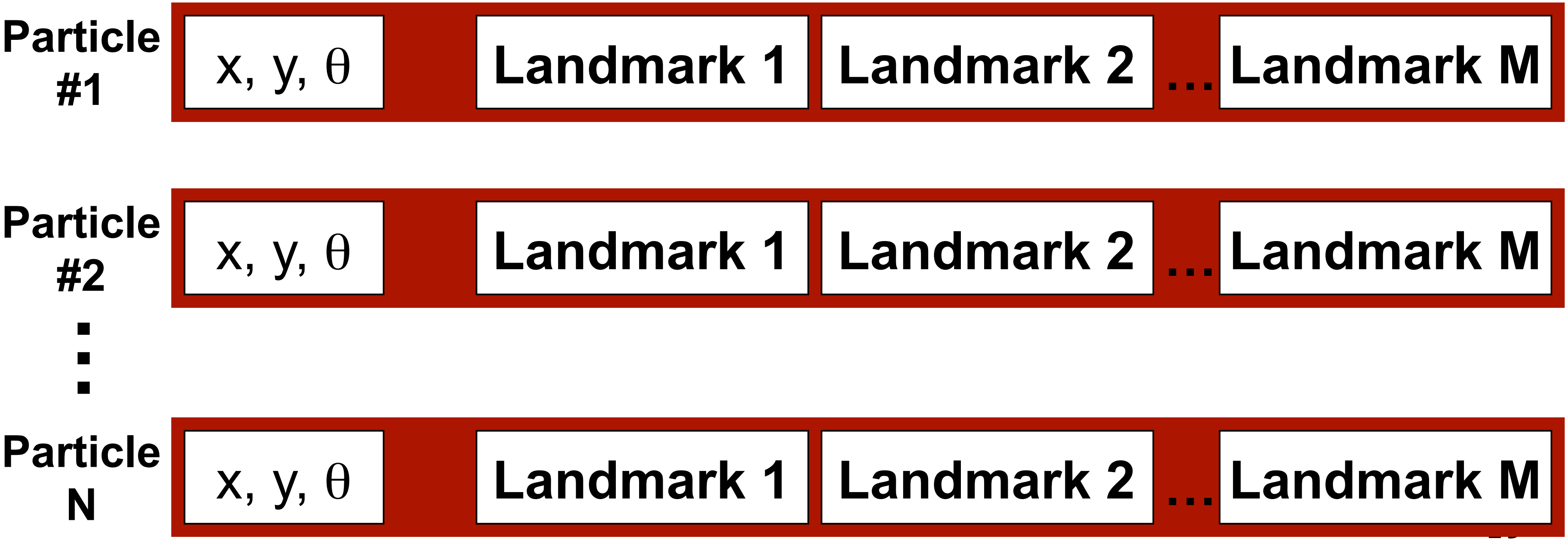
$$p(x_{1:t}, l_{1:m} \mid z_{1:t}, u_{0:t-1}) =$$

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

- This factorization is also called Rao-Blackwellization
- Given that the second term can be computed efficiently, particle filtering becomes possible!

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



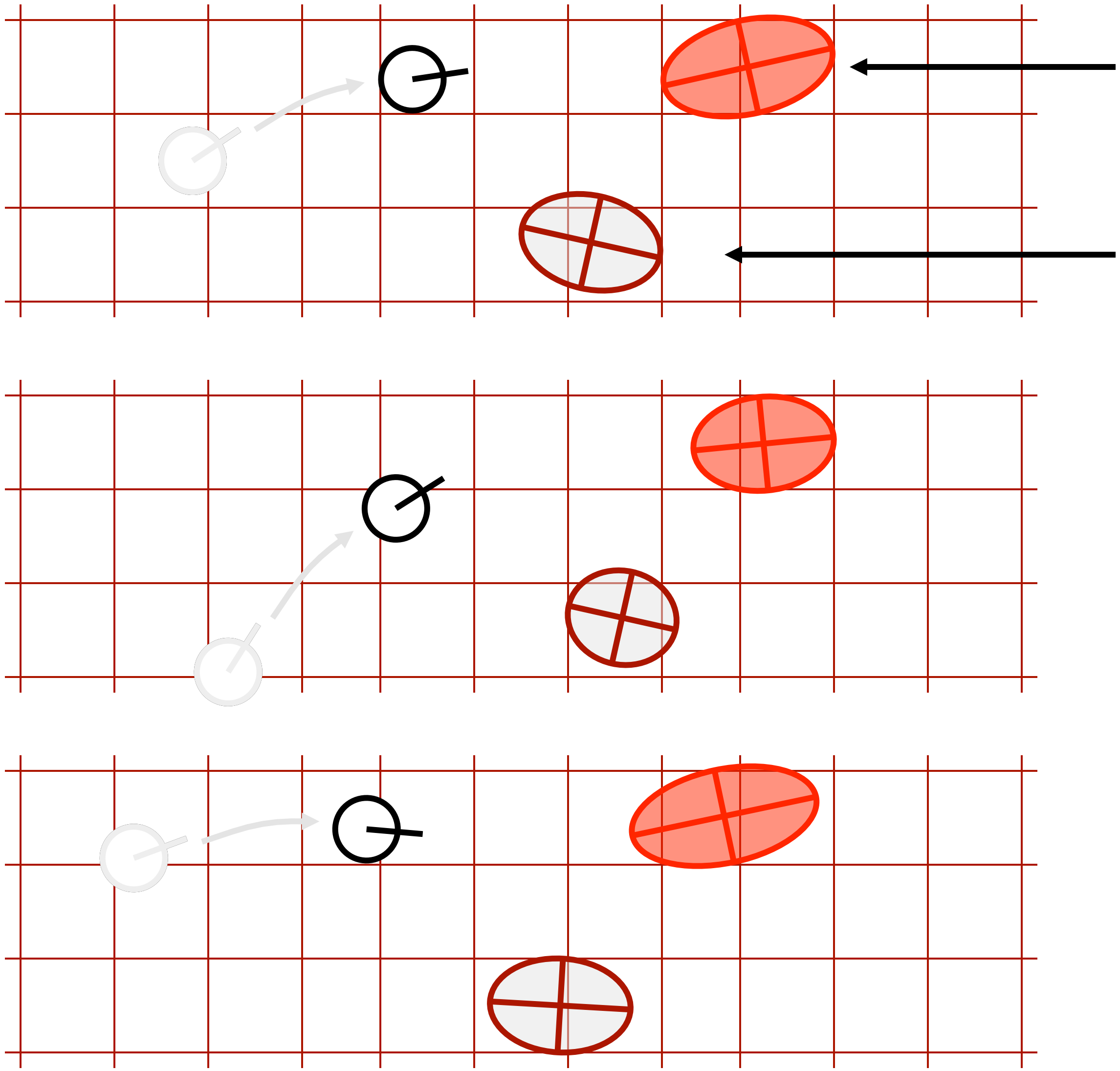
Particle #1 | x, y, θ | Landmark 1 | Landmark 2 | … | Landmark M

Particle #2 | x, y, θ | Landmark 1 | Landmark 2 | … | Landmark M

⋮

Particle N | x, y, θ | Landmark 1 | Landmark 2 | … | Landmark M

FastSLAM – Action Update



Particle #1

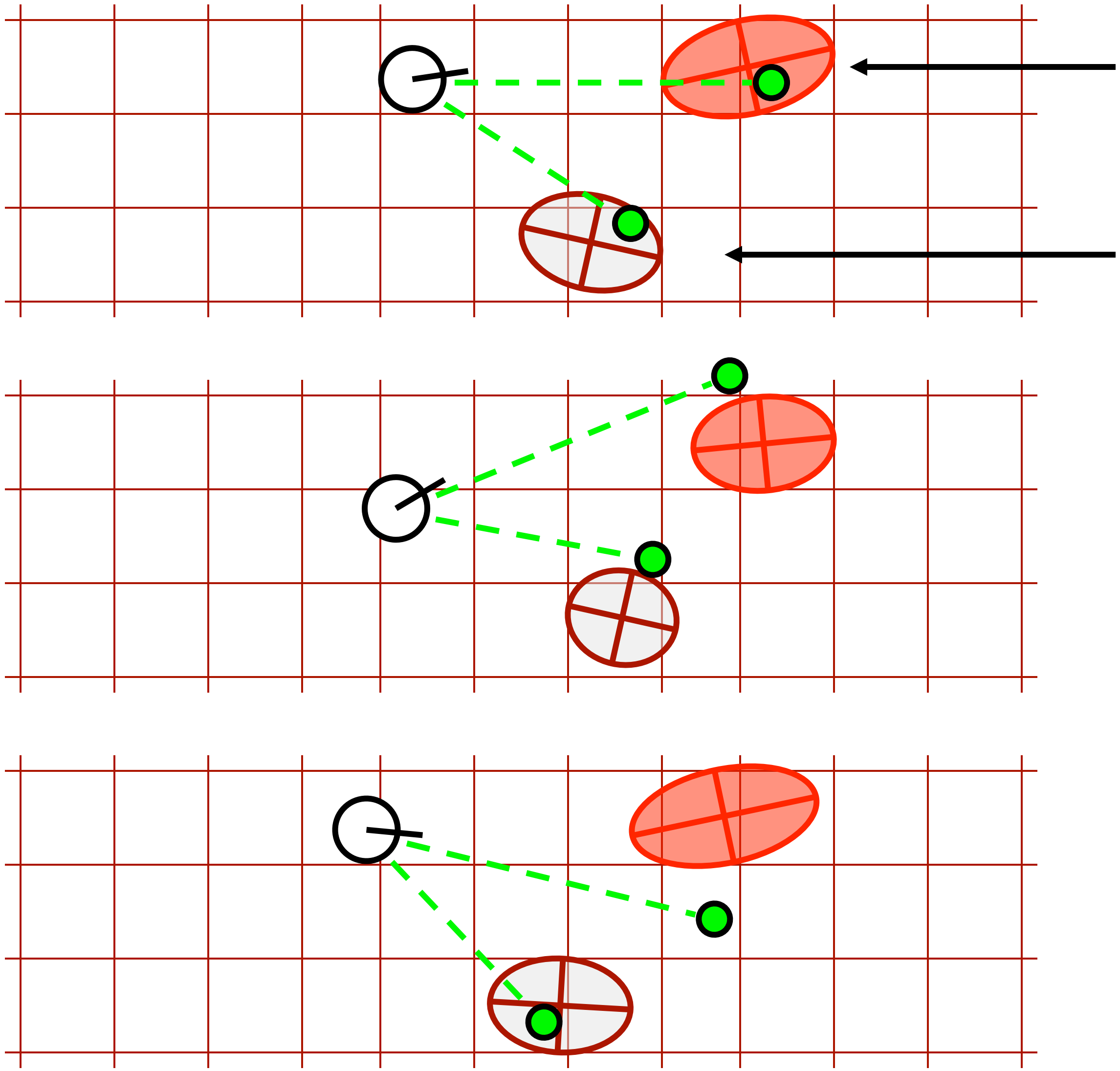
Landmark #1 Filter

Landmark #2 Filter

Particle #2

Particle #3

FastSLAM – Sensor Update



Landmark #1 Filter

Landmark #2 Filter

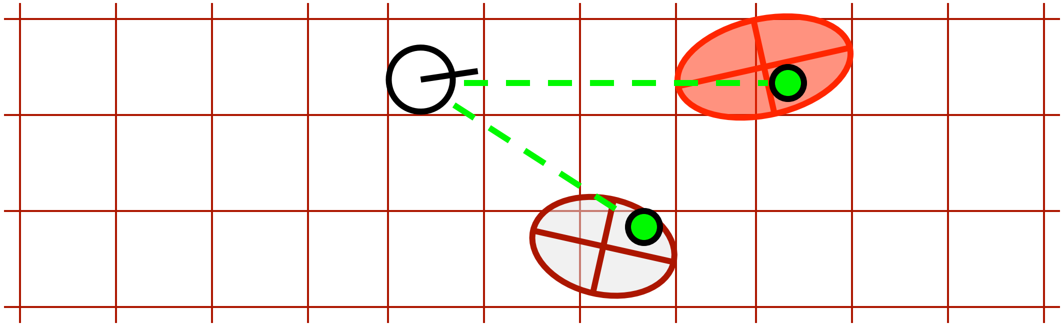
Particle #1

Particle #2

Particle #3

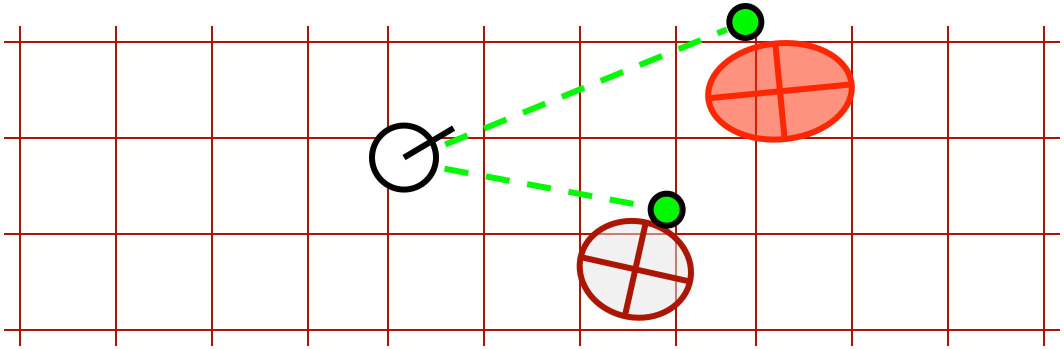
FastSLAM – Sensor Update

Particle #1



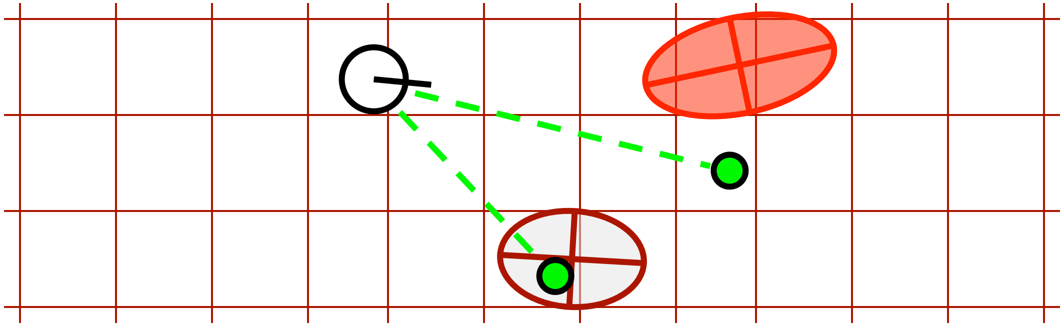
Weight = 0.8

Particle #2



Weight = 0.4

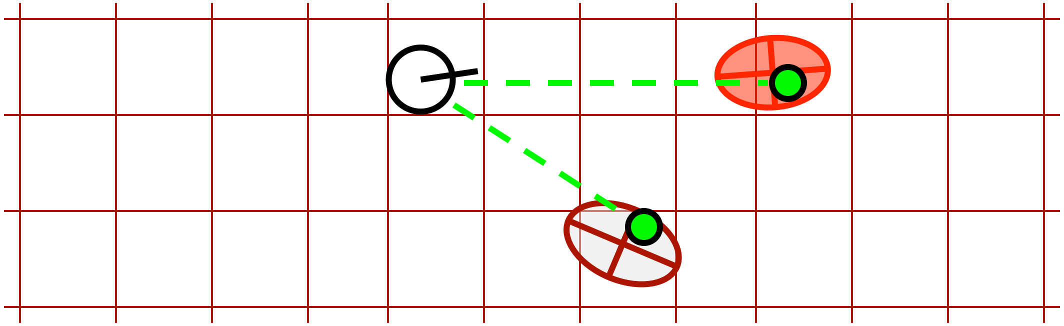
Particle #3



Weight = 0.1

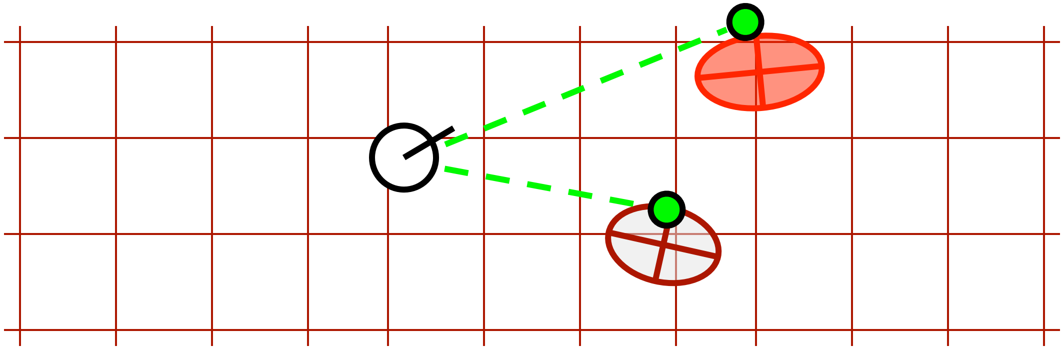
FastSLAM – Sensor Update

Particle #1



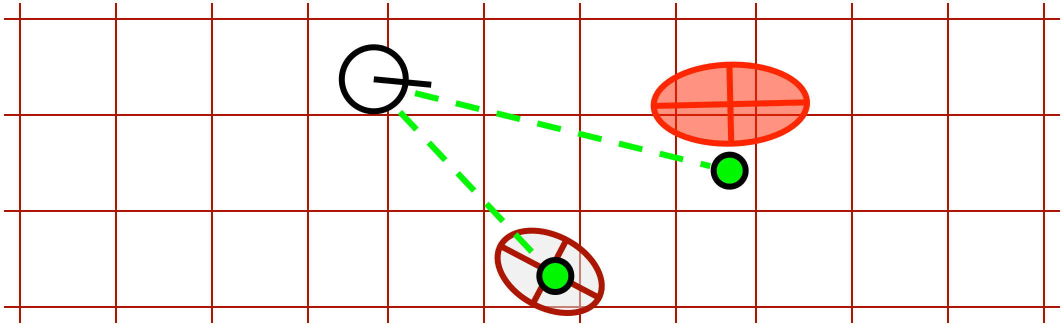
Update map of particle #1

Particle #2



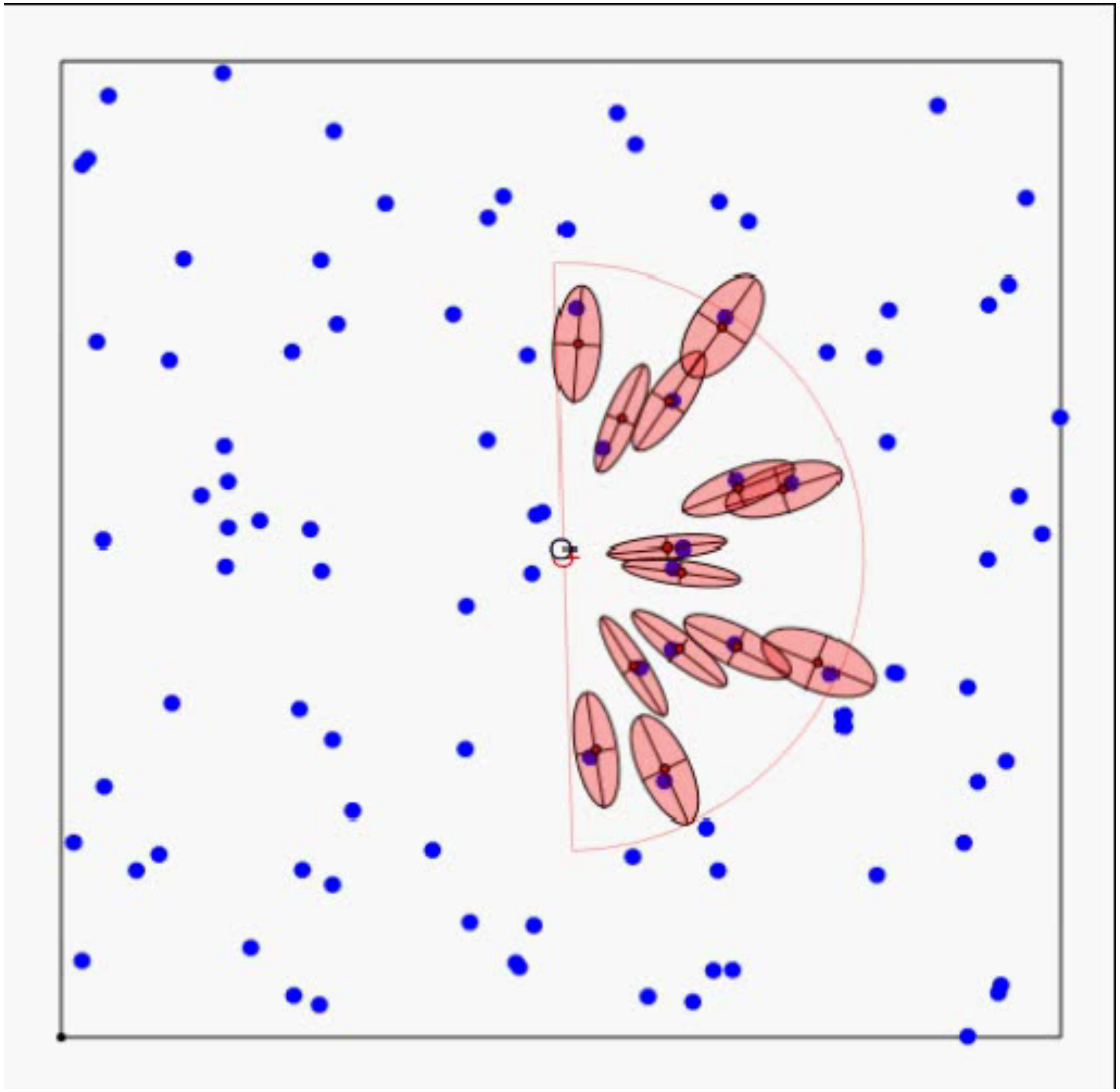
Update map of particle #2

Particle #3



Update map of particle #3

FastSLAM - 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•log(M))

Log time (per particle)

- Resample particle set

O(N•log(M))

Log time (per particle)

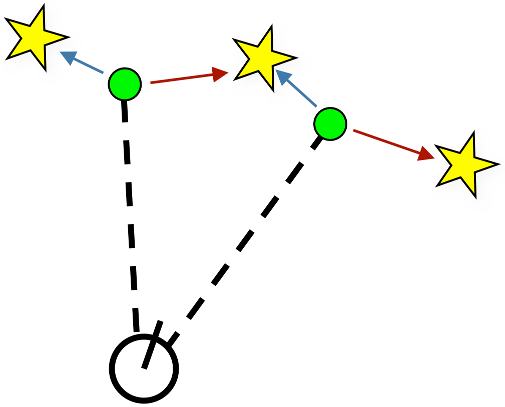
N = Number of particles
M = Number of map features

O(N•log(M))

Log time in the number of landmarks, linear in the number of particles

Data Association Problem

- Which observation belongs to which landmark?

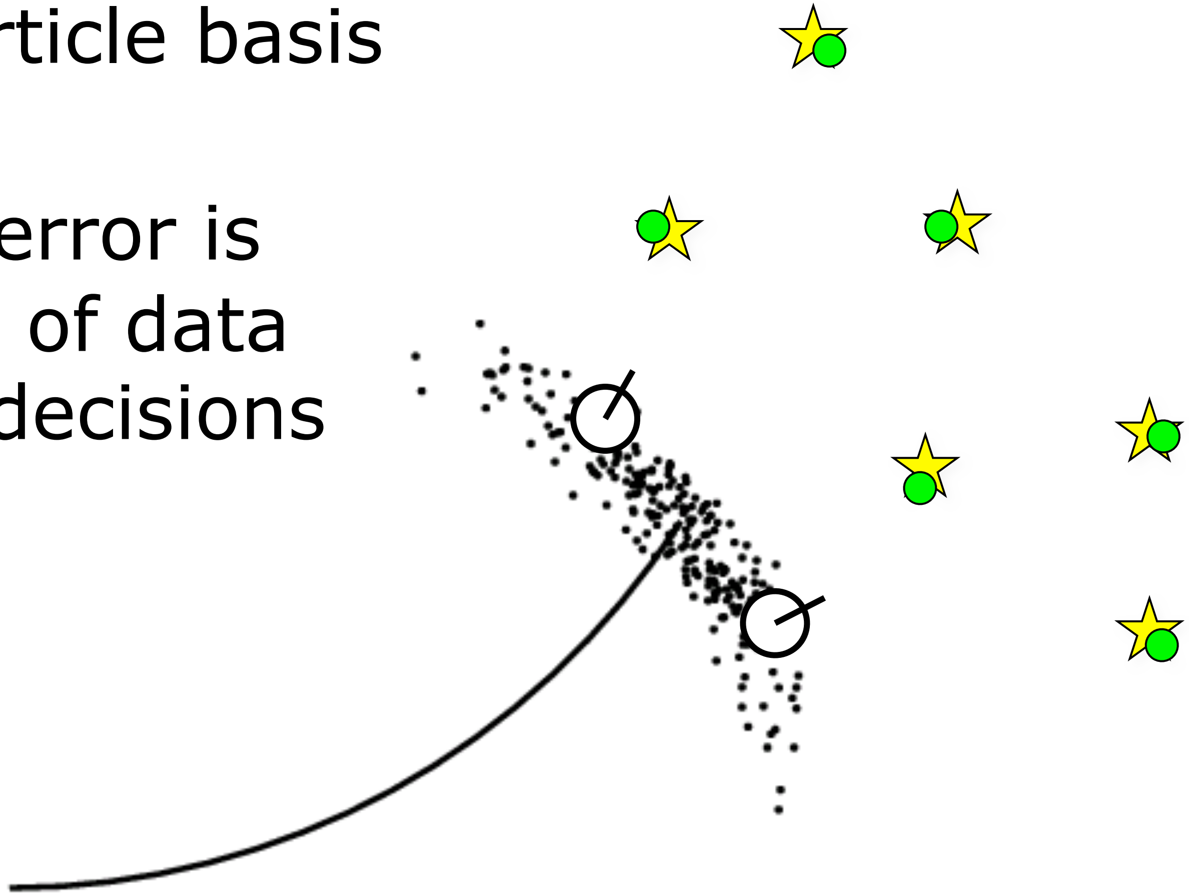


- A robust SLAM solution must consider possible data associations
- Potential data associations depend also on the pose of the robot

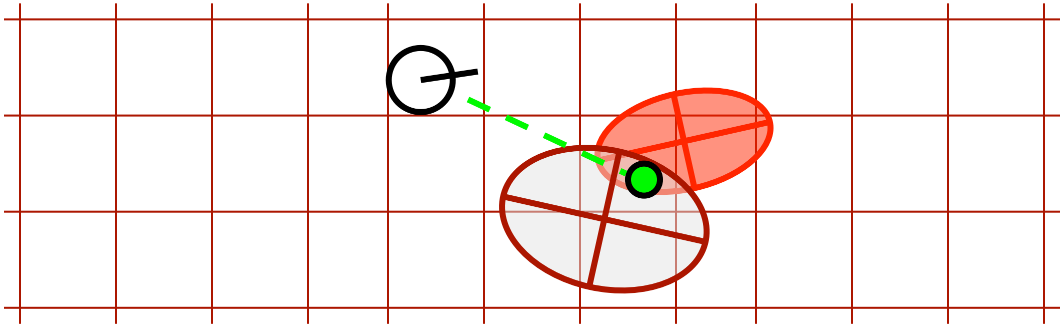
Multi-Hypothesis Data Association

- Data association is done on a per-particle basis

- Robot pose error is factored out of data association decisions



Per-Particle Data Association



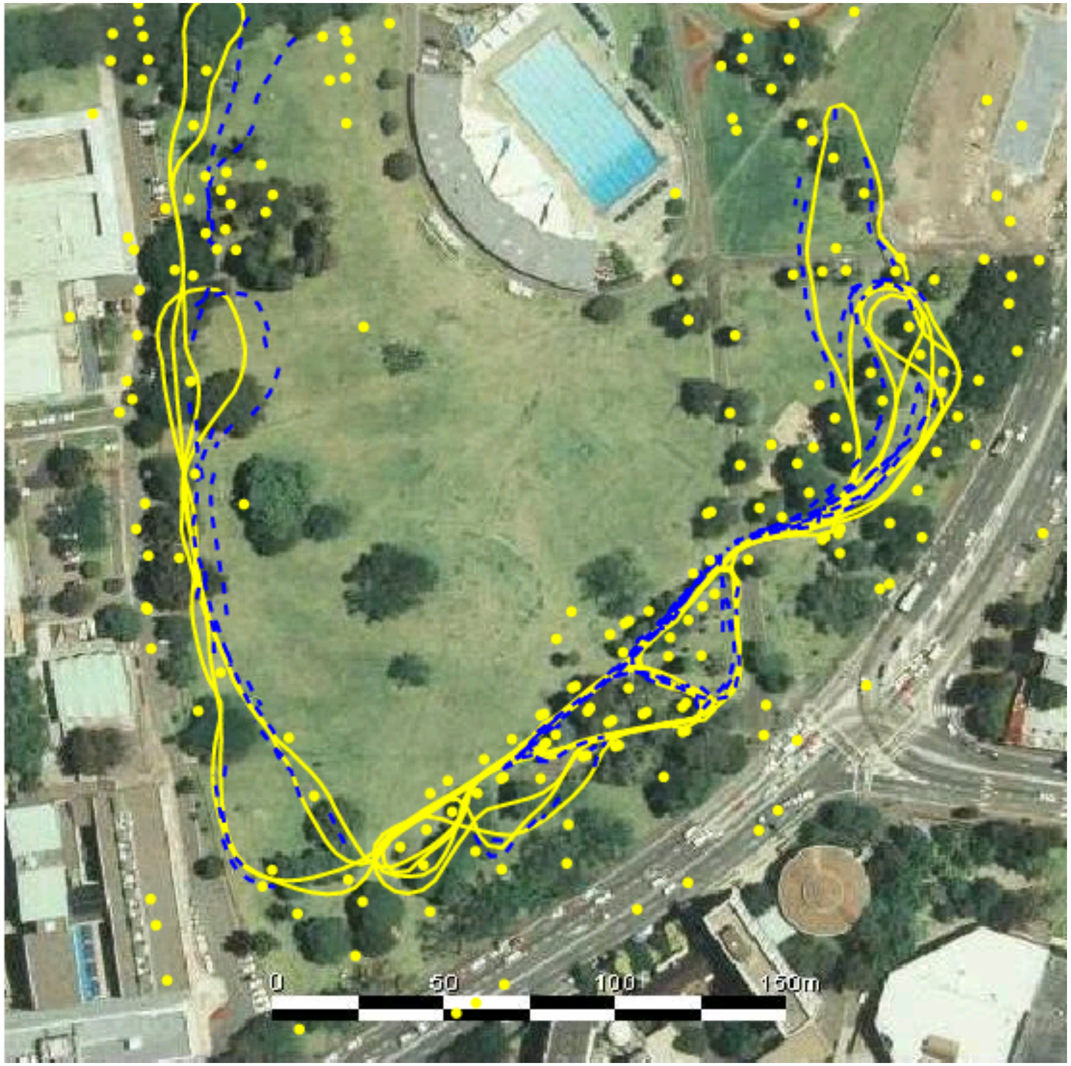
Was the observation generated by the red or the brown landmark?

P(observation|red) = 0.3   P(observation|brown) = 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



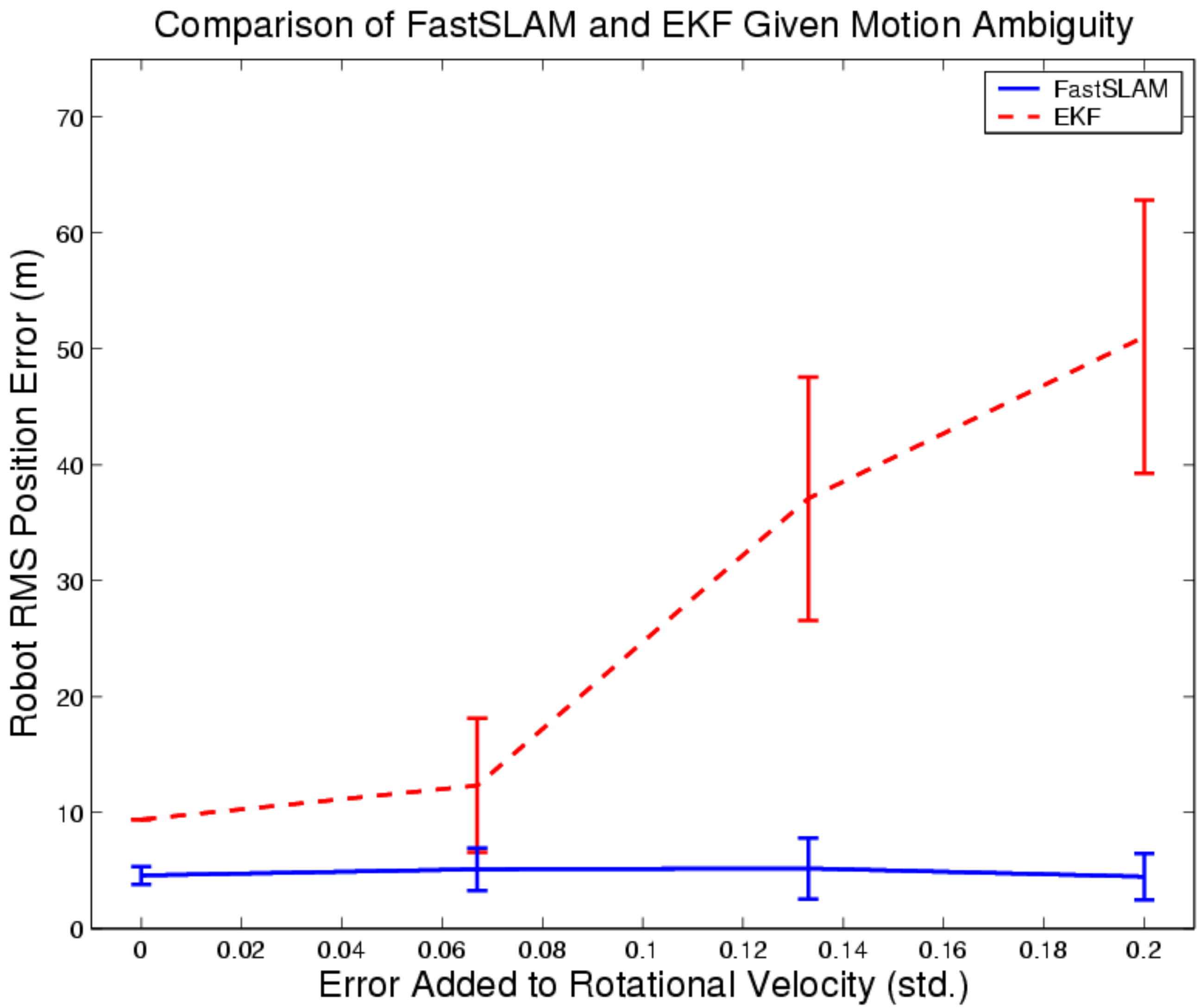
Blue = GPS
Yellow = FastSLAM

Dataset courtesy of University of Sydney

Results – Victoria Park (Video)

Dataset courtesy of University of Sydney

Results – Data Association



FastSLAM Summary

- FastSLAM factors the SLAM posterior into low-dimensional estimation problems
  - Scales to problems with over 1 million features
- FastSLAM factors robot pose uncertainty out of the data association problem
  - Robust to significant ambiguity in data association
  - Allows data association decisions to be delayed until unambiguous evidence is collected
- Advantages compared to the classical EKF approach (especially with non-linearities)
- Complexity of $O(N \log M)$