FastSLAM*

Prof. Mohan Sridharan
Chair in Robot Systems
University of Edinburgh, UK
https://homepages.inf.ed.ac.uk/msridhar/
m.sridharan@ed.ac.uk

*Revised original slides that accompany the book: PR by Thrun, Burgard and Fox.

The SLAM Problem

- 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 and egg problem: a map is needed to localize the robot and a pose estimate is needed to build a map.

The SLAM Problem

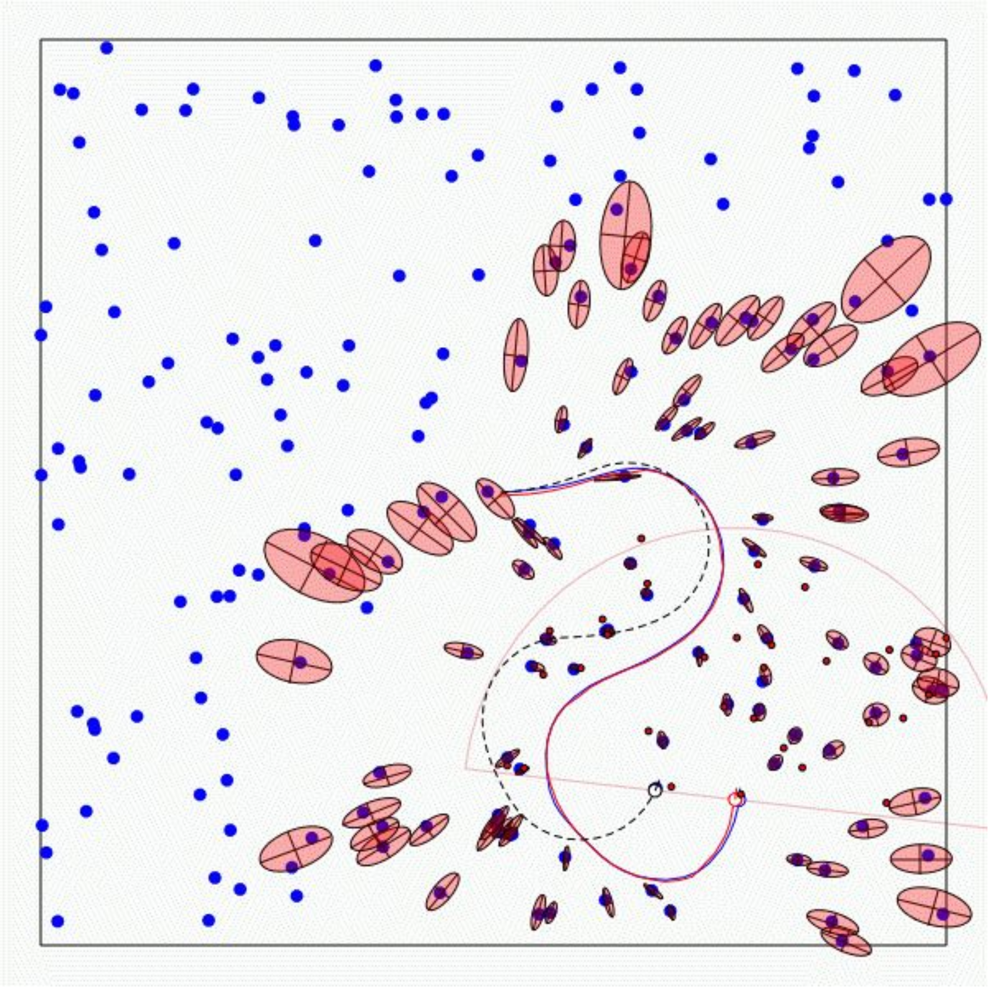
A robot moving though an unknown, static environment!

Given:

- The robot's controls.
- Observations of nearby features.

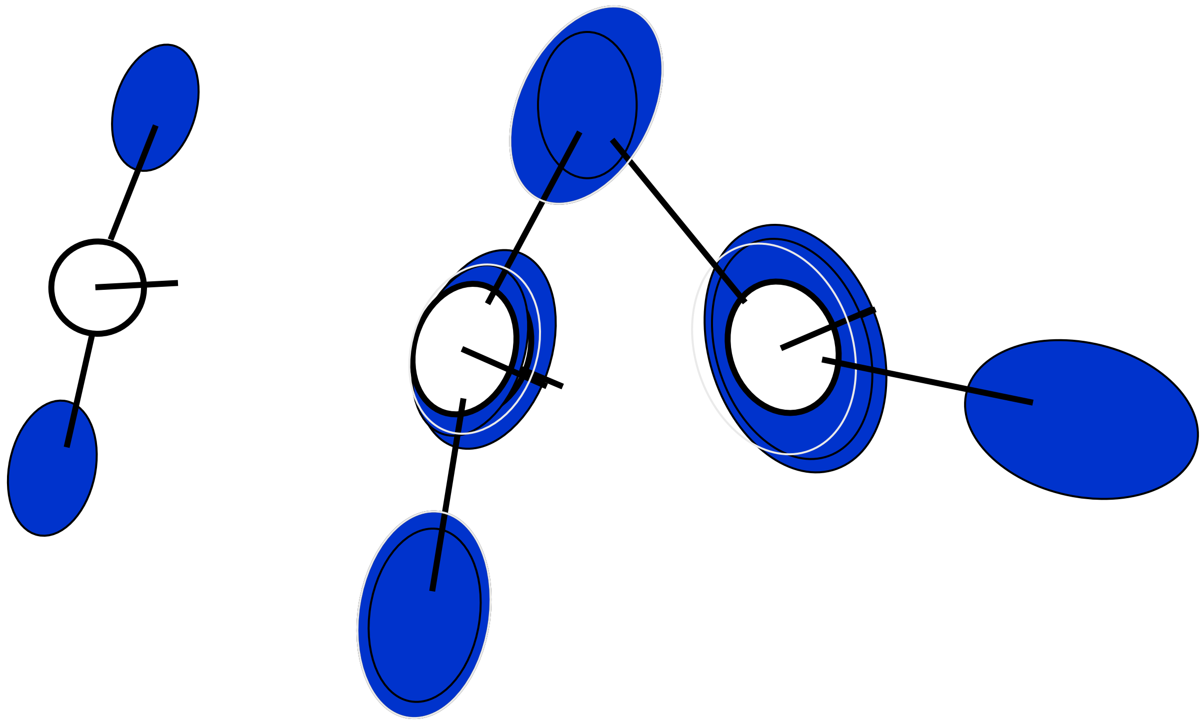
Estimate:

- Map of features.
- Path of the robot.



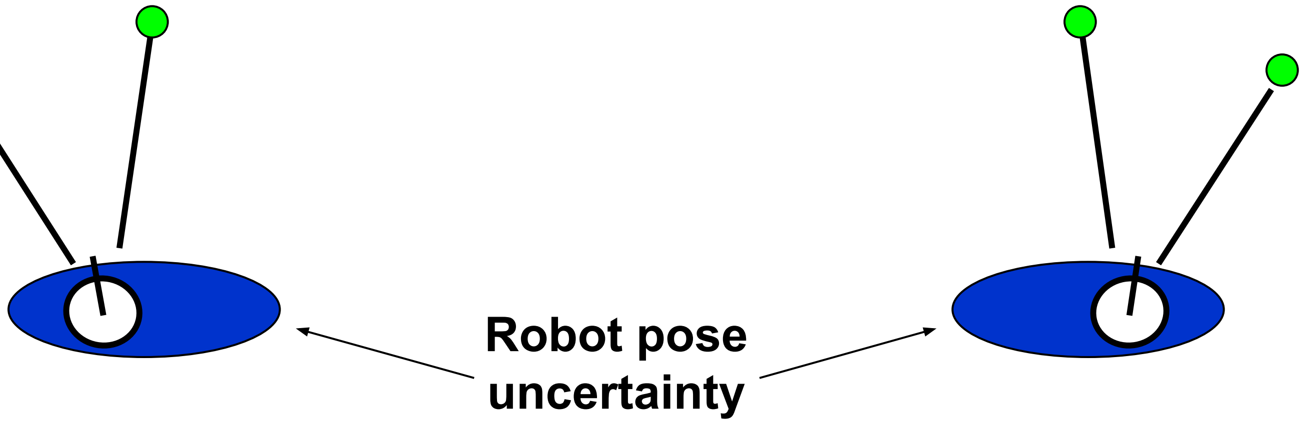
Why is SLAM a hard problem?

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



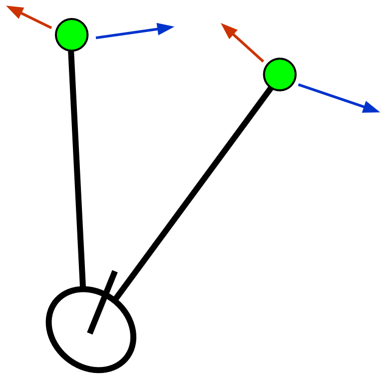
Robot path error correlates errors in the map.

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



- Data association: assignment of observations to landmarks i.e. correspondence.
- 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.
  - Perform re-sampling.
- Localization, multi-hypothesis tracking.

Localization and SLAM

- Particle filters can be used to solve both problems.

- Localization: state space $<x, y, \theta>$

- SLAM: state space $<x, y, \theta, map>$
  - for landmark maps = $<m_1, m_2, \ldots, m_N>$
  - for grid maps = $<c_{11}, c_{12}, \ldots, c_{1n}, c_{21}, \ldots, c_{nm}>$

- **Problem:** number of particles needed to model a posterior is exponential in state-space dimension!

Exploiting Dependencies

- Target:

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

- Is there a dependency between the dimensions of the state space?

- If so, can we use the dependency to solve the problem more efficiently?

Exploit Dependencies

- In the context of SLAM:
  - The map depends on the poses of the robot.
  - We know how to build a map if the position of the sensor is known.
  - *Given robot pose, we can estimate locations of all features independent of each other!*

Factored Posterior (Landmarks)

poses map observations movements

$$p(x_{1:t}, m_{1:N} \mid z_{1:t}, u_{1:t}) = p(x_{1:t} \mid z_{1:t}, u_{1:t})\, p(m_{1:N} \mid x_{1:t}, z_{1:t})$$

SLAM posterior

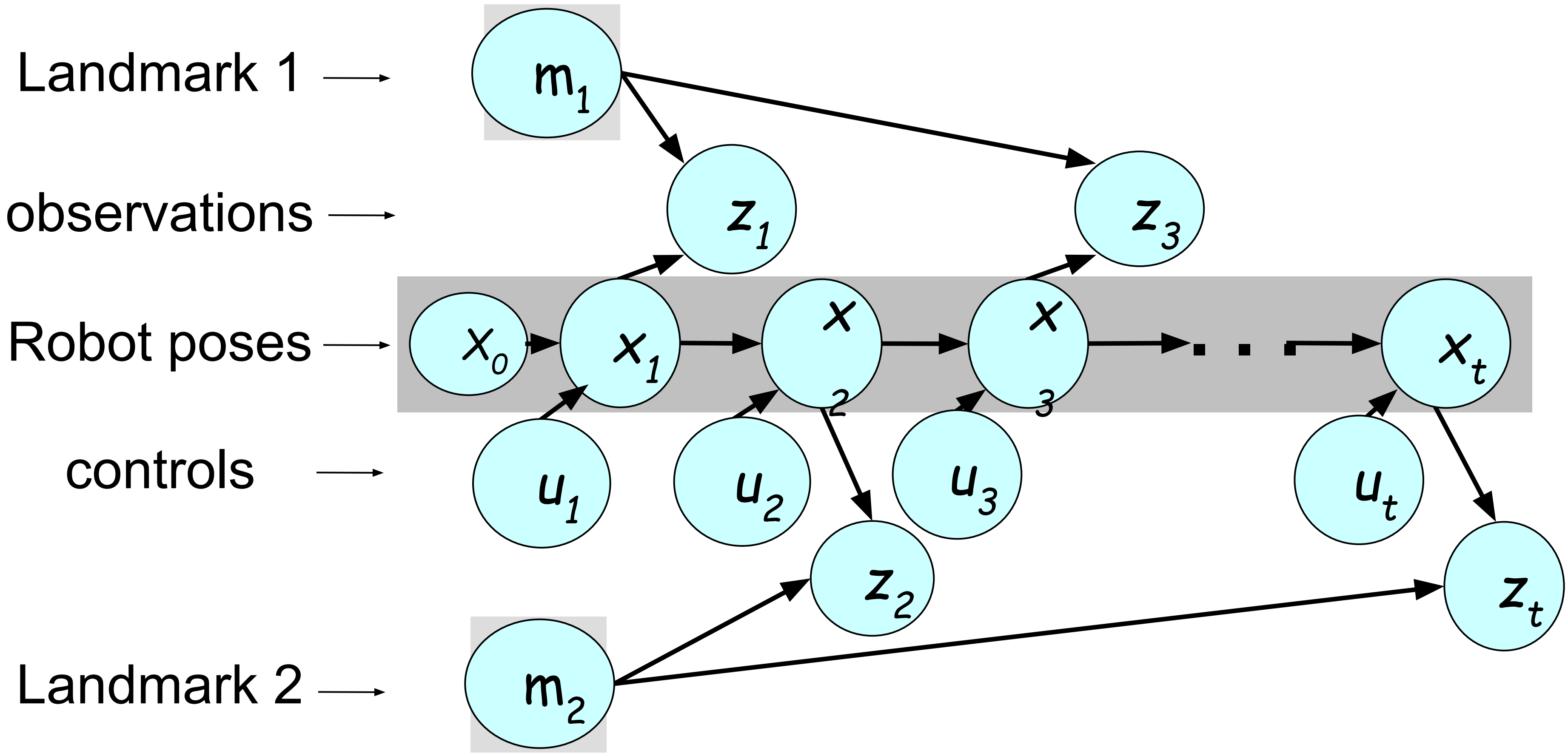
Robot path posterior

landmark positions

Does this help to solve the problem?

Factorization first introduced by Murphy in 1999

Mapping using Landmarks



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

Factored Posterior

$$p(x_{1:t}, m_{1:N} \mid z_{1:t}, u_{1:t}) = p(x_{1:t} \mid z_{1:t}, u_{1:t})\, p(m_{1:N} \mid x_{1:t}, z_{1:t})$$

$$= p(x_{1:t} \mid z_{1:t}, u_{1:t}) \prod_i p(m_i \mid x_{1:t}, z_{1:t})$$

Robot path posterior (localization problem)

Conditionally independent landmark positions

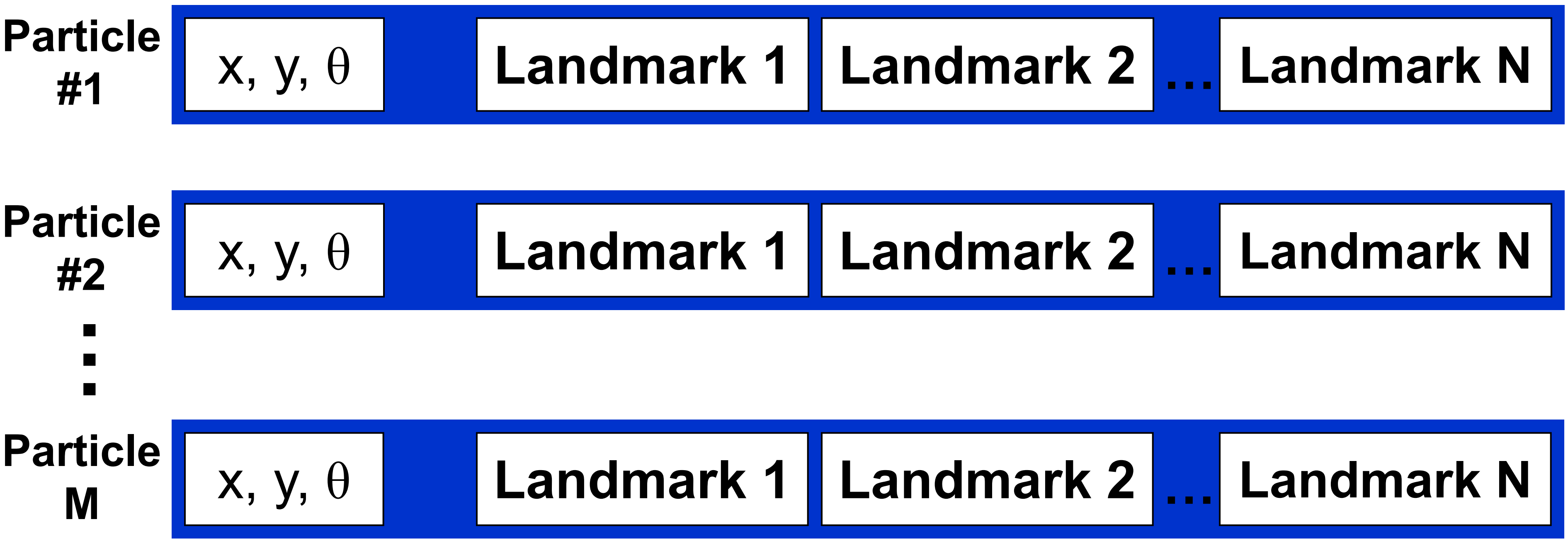
Rao-Blackwellization

$$p(x_{1:t}, m_{1:N} \mid z_{1:t}, u_{1:t}) = p(x_{1:t} \mid z_{1:t}, u_{1:t})\, p(m_{1:N} \mid x_{1:t}, z_{1:t})$$
$$= p(x_{1:t} \mid z_{1:t}, u_{1:t}) \prod_i p(m_i \mid x_{1:t}, z_{1:t})$$

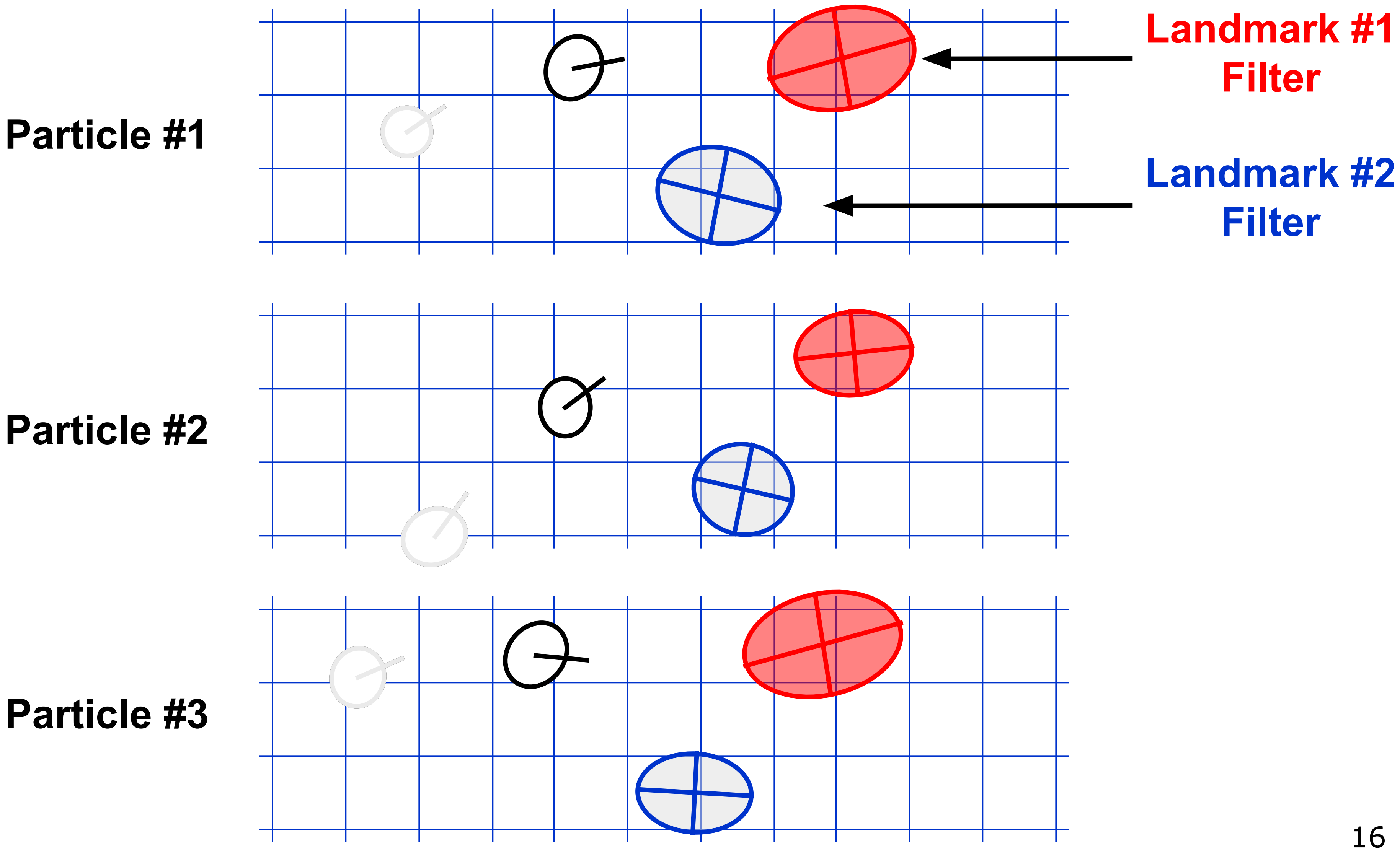
- This factorization is called Rao-Blackwellization.
- Estimate robot pose as a particle filter.
- Each particle associated with a set of Gaussians, one for each landmark position.
- Landmark positions estimated using EKFs.

FastSLAM

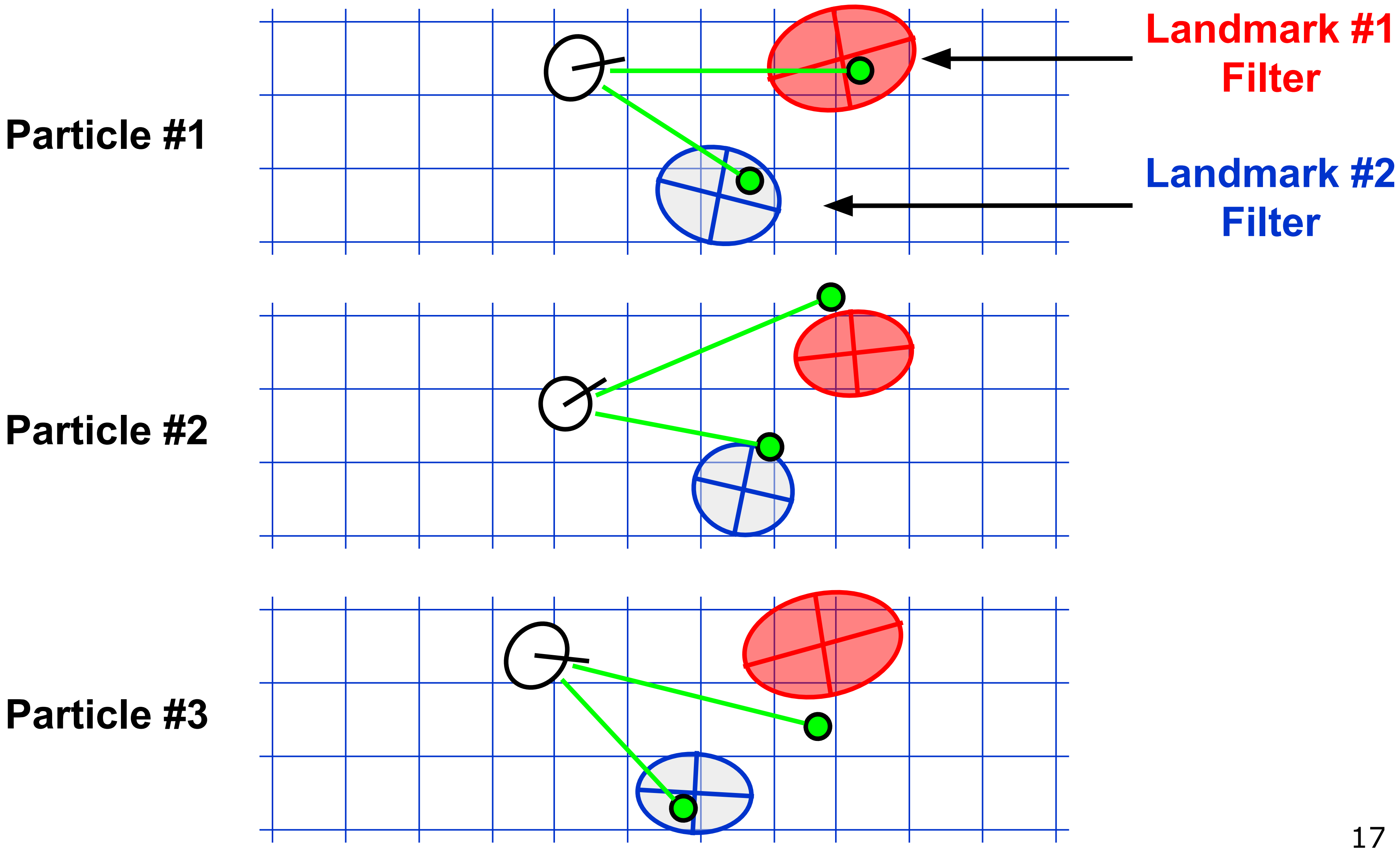
- Rao-Blackwellized particle filtering based on landmarks.
- Each landmark represented by a 2x2 EKF.
- Each particle therefore has to maintain *N* EKFs.



FastSLAM – Action Update



FastSLAM – Sensor Update

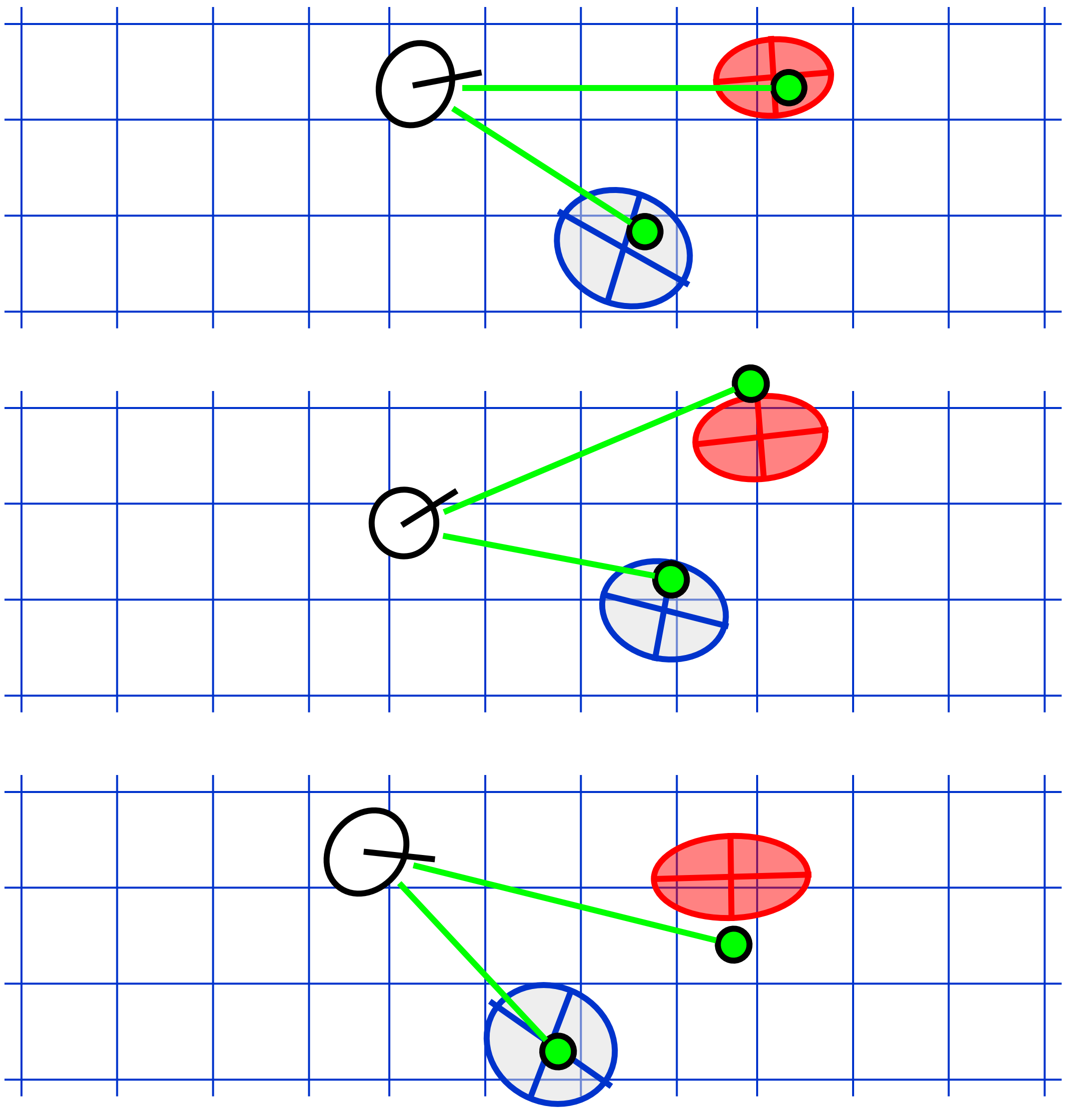


FastSLAM – Sensor Update

Particle #1

Particle #2

Particle #3



Weight = 0.8

Weight = 0.4

Weight = 0.1

Update Steps (known correspondence)

- Do for M particles:
  - Retrieve a pose from particle set.
  - Sample new pose – *notice lack of measurement update!*

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

  - Measurement update - for each observed features, identify correspondence and incorporate into appropriate EKF by updating mean and covariance.
  - Compute importance factor – include measurement in pose update.
- Resample based on importance weights.

Update Steps (known correspondence)

- Do for M particles:
  - Sample new pose – *notice lack of measurement update!*

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

  - Update posterior over observed landmark/feature (similar technique as in EKF-SLAM or even EKF).

$$p(m_{c_t} \mid x_{1:t}, z_{1:t}, c_{1:t}) = \eta \; p(z_t \mid x_t, m_{c_t}, c_t) \; p(m_{c_t} \mid x_{1:t-1}, z_{1:t-1}, c_{1:t-1})$$

  - Compute importance factor – include measurement in pose update:

$$w_t^{[k]} = \frac{p(x_t^{[k]} \mid z_{1:t}, u_{1:t}, c_{1:t})}{p(x_t^{[k]} \mid z_{1:t-1}, u_{1:t}, c_{1:t-1})} = \eta \; p(z_t \mid x_t^{[k]}, z_{1:t-1}, c_{1:t})$$

- Resample based on importance weights.

- FastSLAM 1.0 (Section 13.3).

Update Steps (FastSLAM 2.0)

- Do for N particles:
  - Obtain proposal distribution – *include measurement in computation.*

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

  - Update posterior over observed landmark/feature.

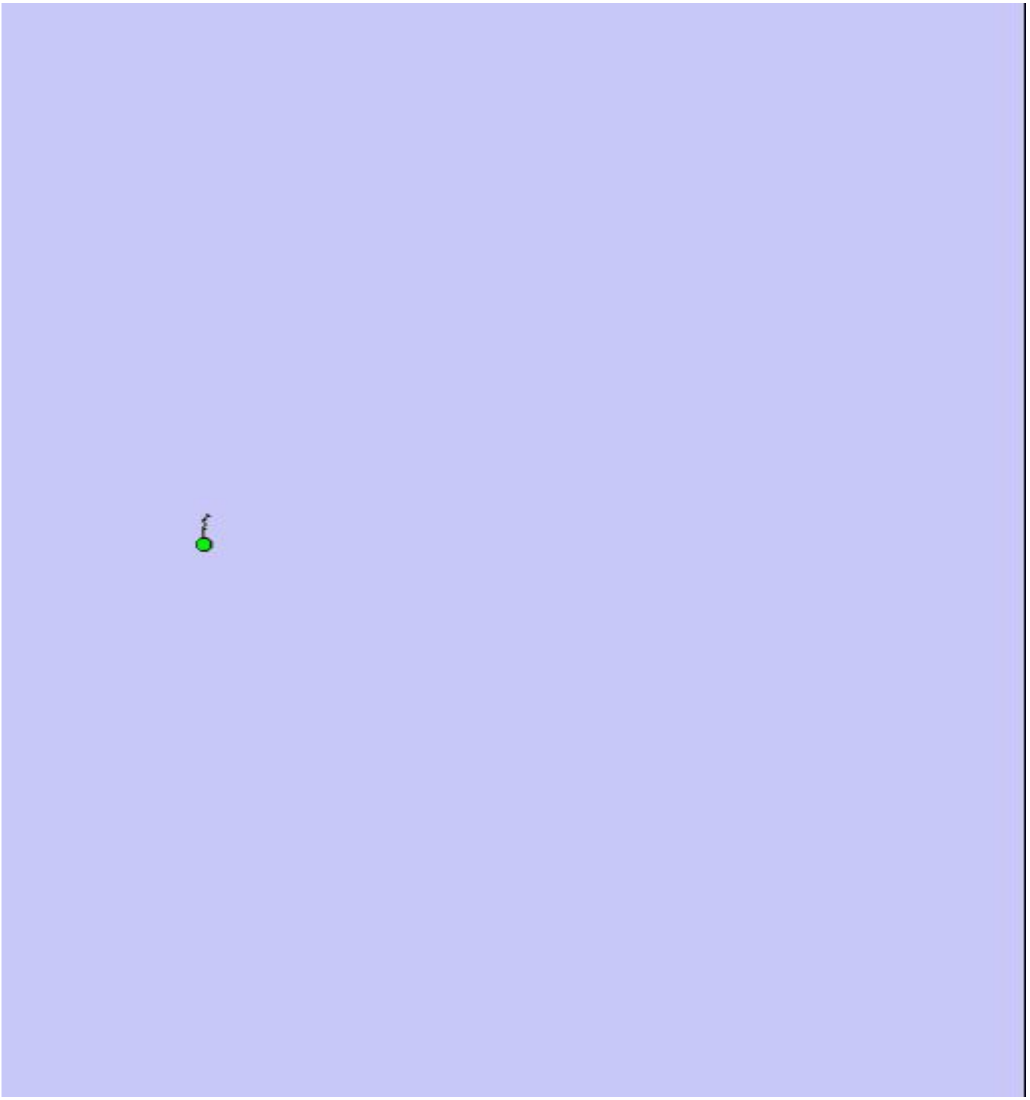
$$p(m_{c_t} \mid x_t^{[k]}, z_{1:t}, c_{1:t}) = \eta \; p(z_t \mid x_t^{[k]}, m_{c_t}, c_t) \; p(m_{c_t} \mid x_{1:t-1}^{[k]}, z_{1:t-1}, c_{1:t-1})$$

  - Compute importance factor.

$$w_t^{[k]} = \eta \; p(z_t \mid x_{1:t-1}^{[k]}, z_{1:t-1}, c_{1:t}, u_{1:t})$$

- Resample based on importance weights.

FastSLAM - Indoor (Closing the loop)



FastSLAM Complexity

- Update robot particles based on control u_{t-1}.

  $O(M)$
  Constant time per particle

- Incorporate observation z_t into Kalman filters.

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

- Resample particle set.

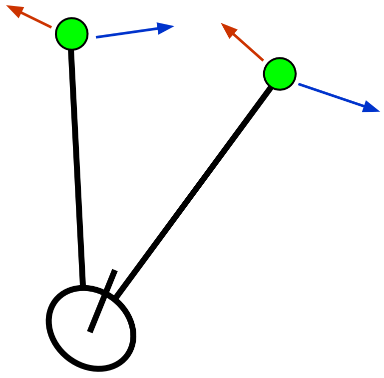
  $O(M \cdot \log(N))$
  Log time per particle

M = Number of particles
N = Number of map features

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

Data Association Problem

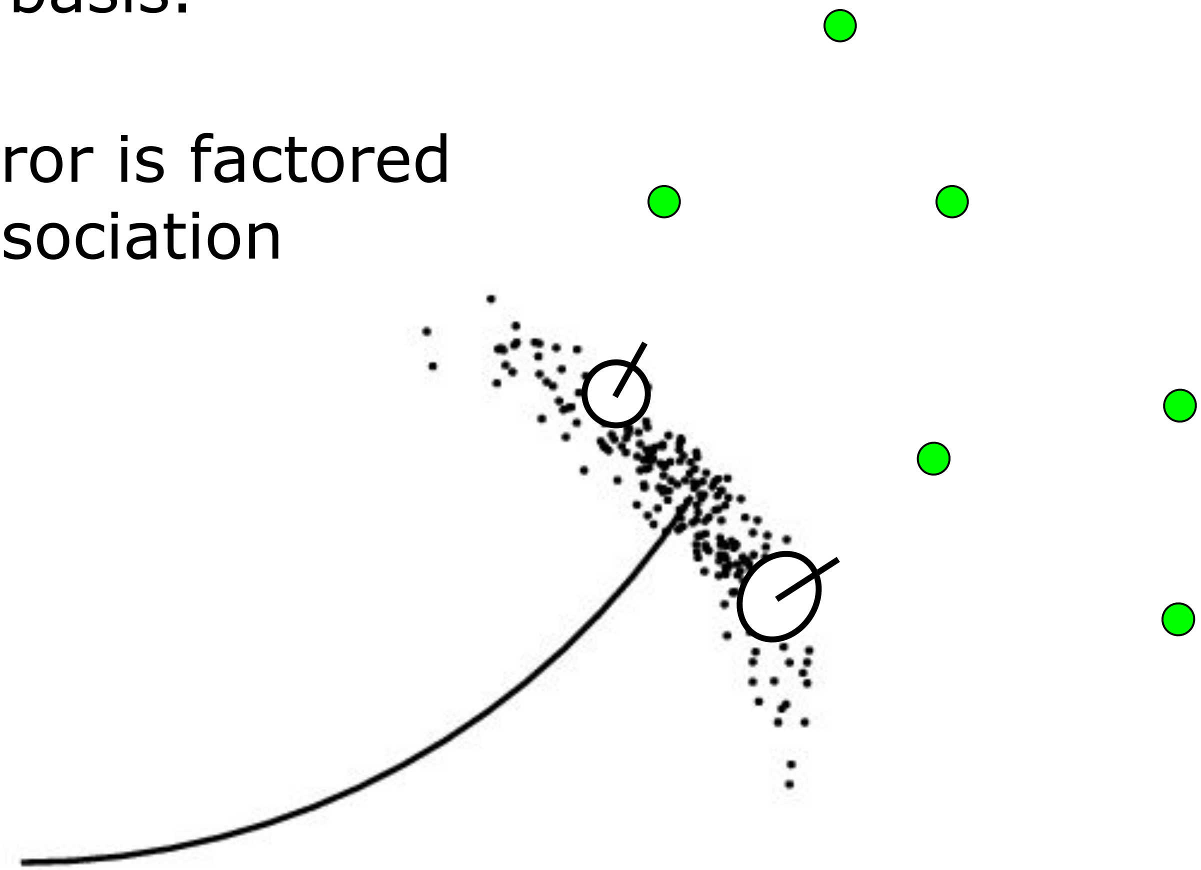
- Which observation belongs to which landmark?



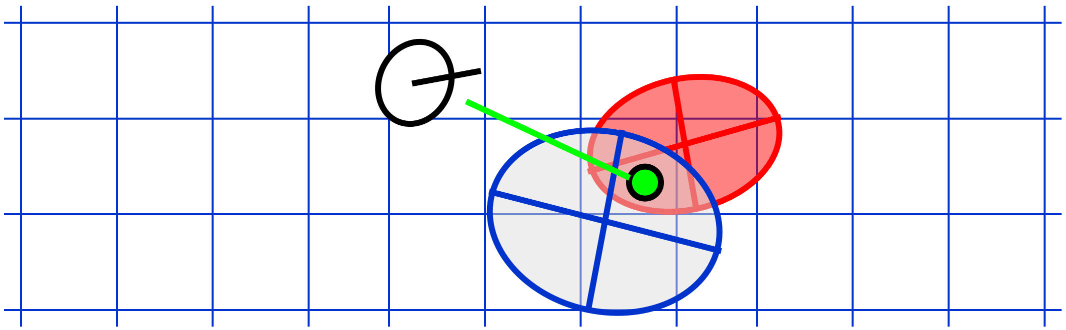
- Robust SLAM must consider possible data associations.
- Potential data associations depend also on the robot pose.

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

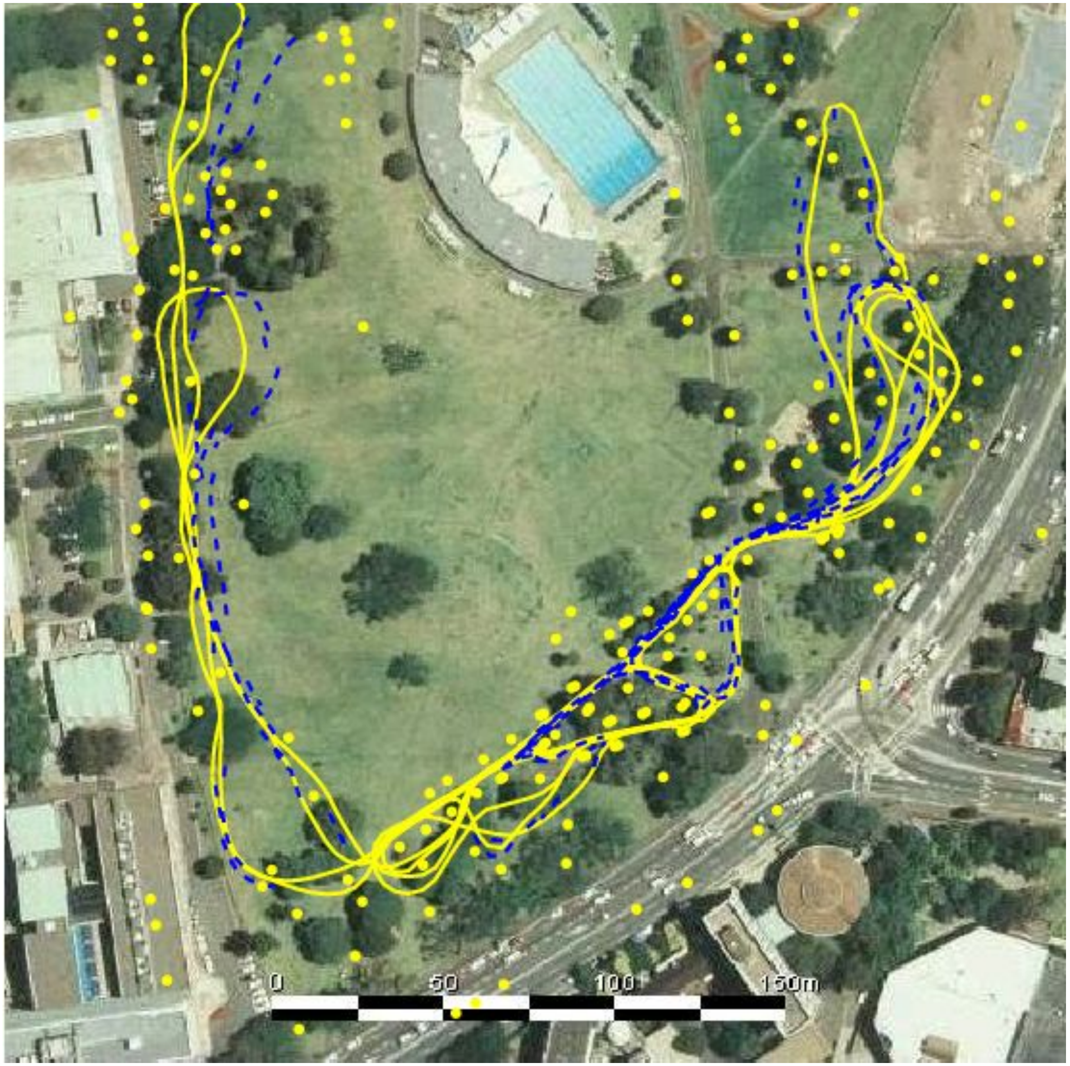
- Two options for per-particle data association:
  - Pick the most probable match.
  - Pick random association weighted by the observation likelihoods.

- If the probability is small, generate new landmark.

Results – Victoria Park

- 4 km traversed.
- < 5 m RMS position error.
- ~100 particles.

Blue = GPS
Yellow = FastSLAM



Dataset courtesy of University of Sydney

Efficiency and other Issues…

- Duplicating map corresponding to same particle.

- Evaluating measurement likelihoods for each of the N map features.
- Efficient data structures – balanced binary trees.

- Loop closure is troublesome.
- Sections 13.8 and 13.9…

- Unknown correspondence – complicated, see section 13.5, 13.6…

Grid-based SLAM

- Can we solve the SLAM problem if no pre-defined landmarks are available?
- Can we use the ideas of FastSLAM to build grid maps?
- As with landmarks, the map depends on the poses of the robot during data acquisition.
- If the poses are known, grid-based mapping is easy ("mapping with known poses").

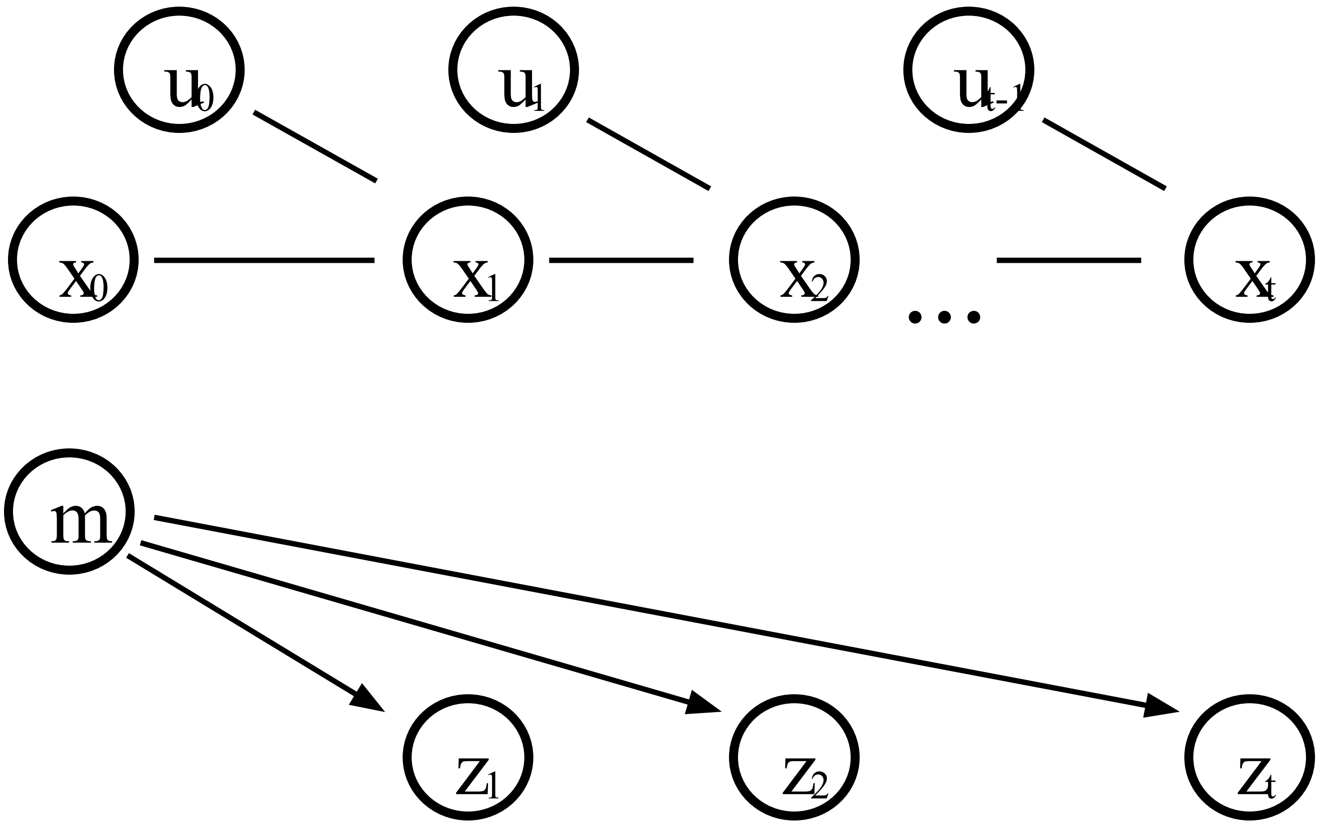
Rao-Blackwellized Mapping

- Each particle represents a possible trajectory of the robot.

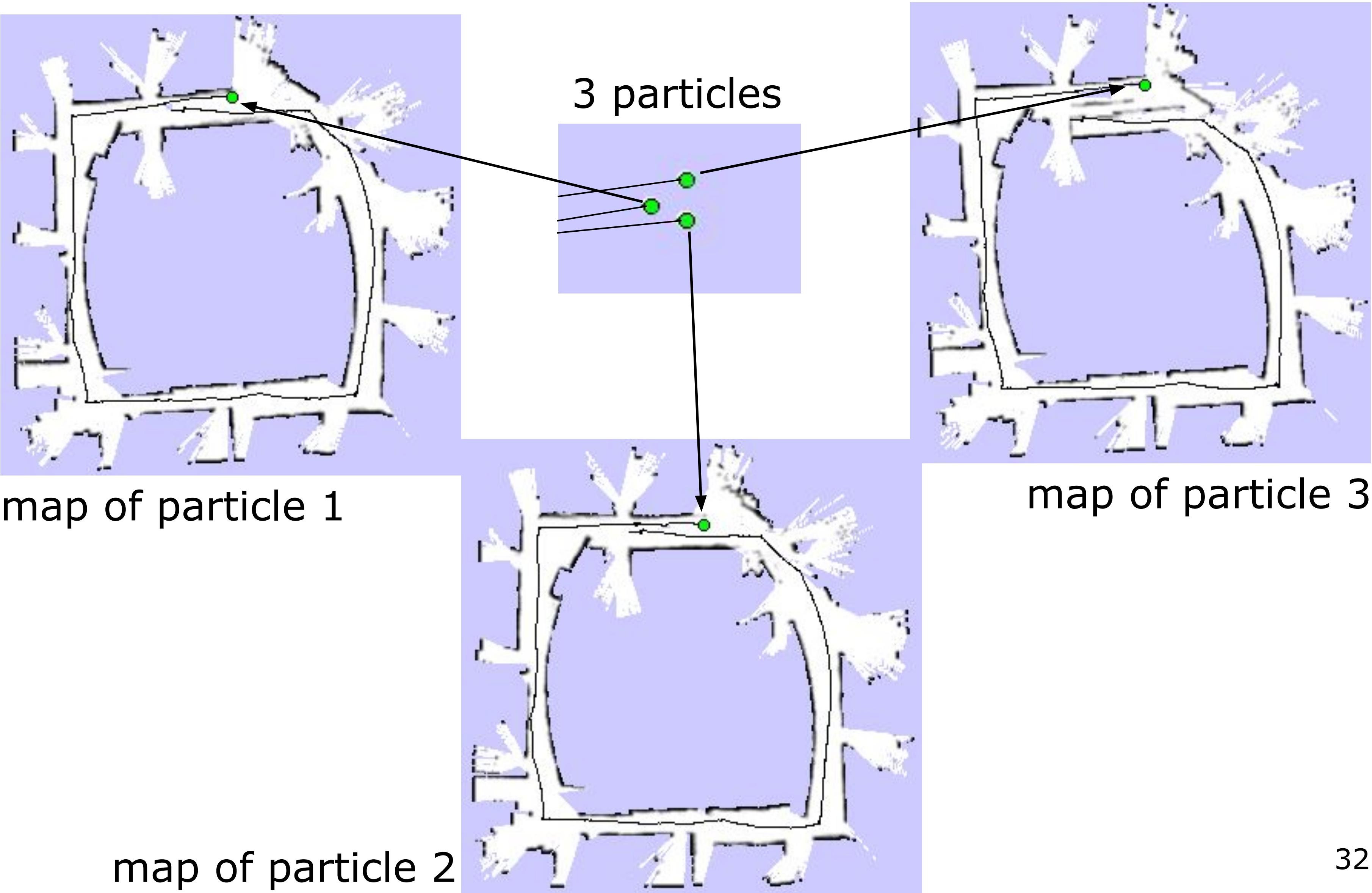
- Each particle:
  - maintains its own map.
  - updates it using “mapping with known poses”.

- Each particle’s probability is proportional to the likelihood of the observations relative to its own map.

A Graphical Model of Rao-Blackwellized Mapping



Particle Filter Example



Problem

- Each map is quite big in case of grid maps!
- Need to keep the number of particles small ☹

- **Solution**:
  Compute better proposal distributions!

- **Idea**:
  Improve the pose estimate **before** applying the particle filter.

Pose Correction Using Scan Matching

Maximize the likelihood of the i^{th} pose and map relative to the $(i-1)^{th}$ pose and map

$$\hat{x}_t = \underset{x_t}{\arg\max} \left\{ p(z_t \mid x_t, \hat{m}_{t-1}) \cdot p(x_t \mid u_{t-1}, \hat{x}_{t-1}) \right\}$$

current measurement

robot
motion

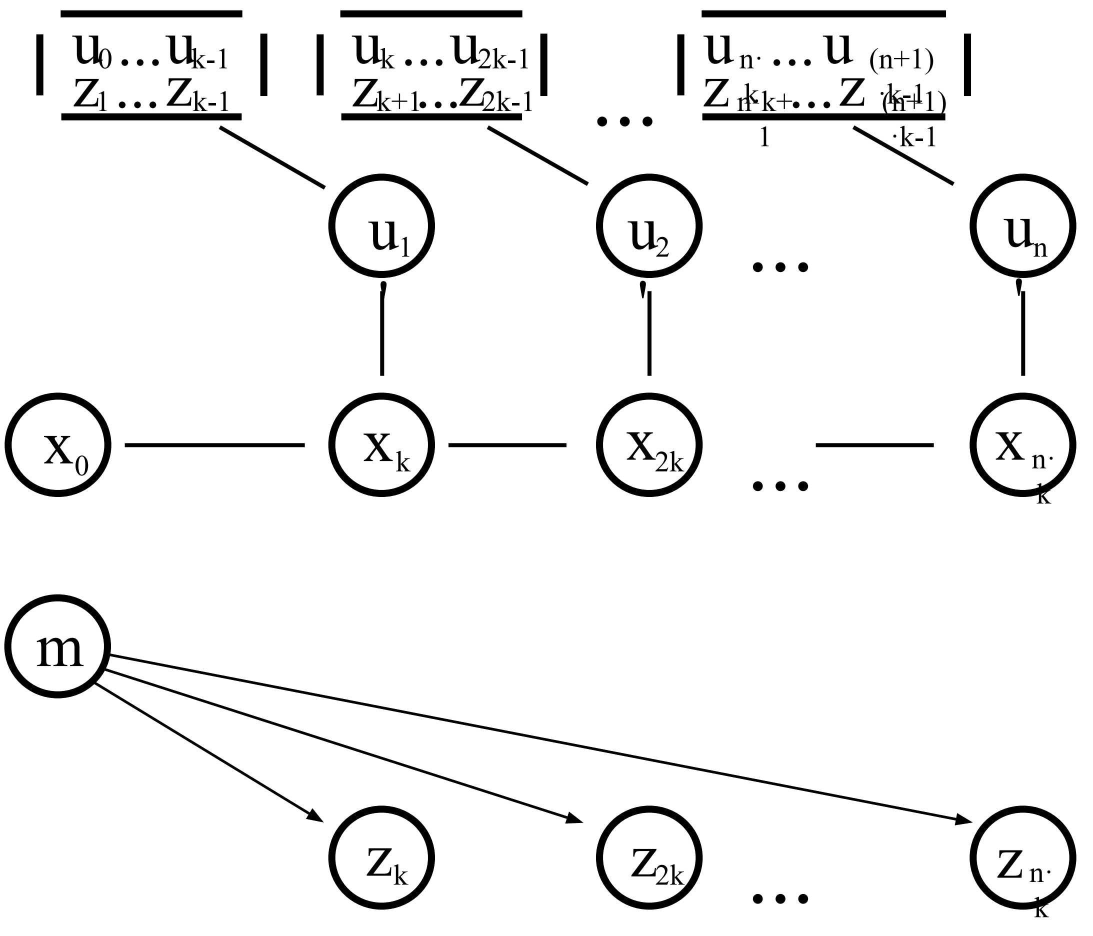
map constructed so far

FastSLAM with Improved Odometry

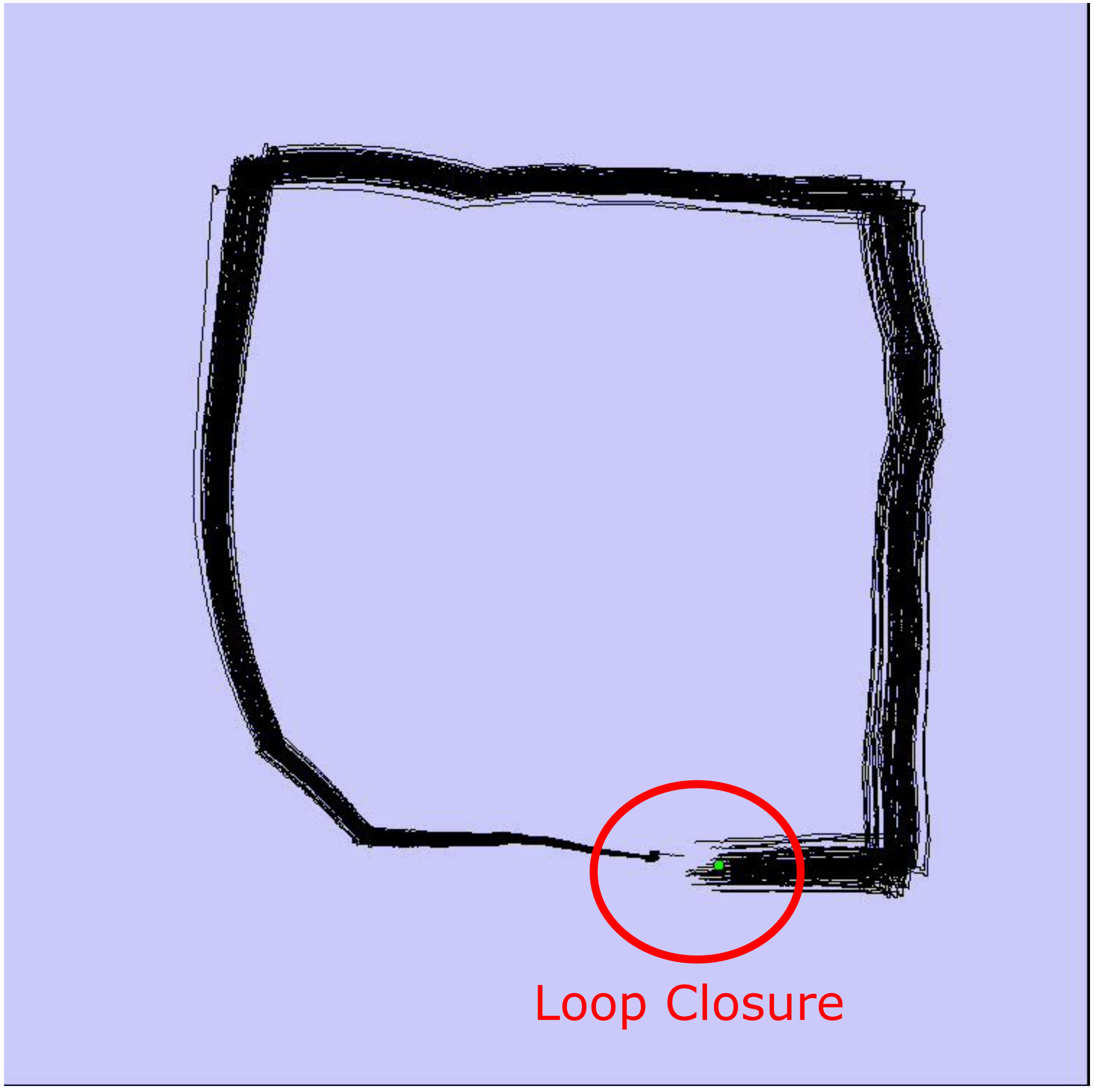
- Scan-matching provides a **locally consistent** pose correction.
- Pre-correct short odometry sequences using scan-matching and use them as input to FastSLAM.
- Fewer particles are needed, since the error in the input in smaller.

[Haehnel et al., 2003]

Graphical Model for Mapping with Improved Odometry



FastSLAM with Scan-Matching



Comparison to Standard FastSLAM

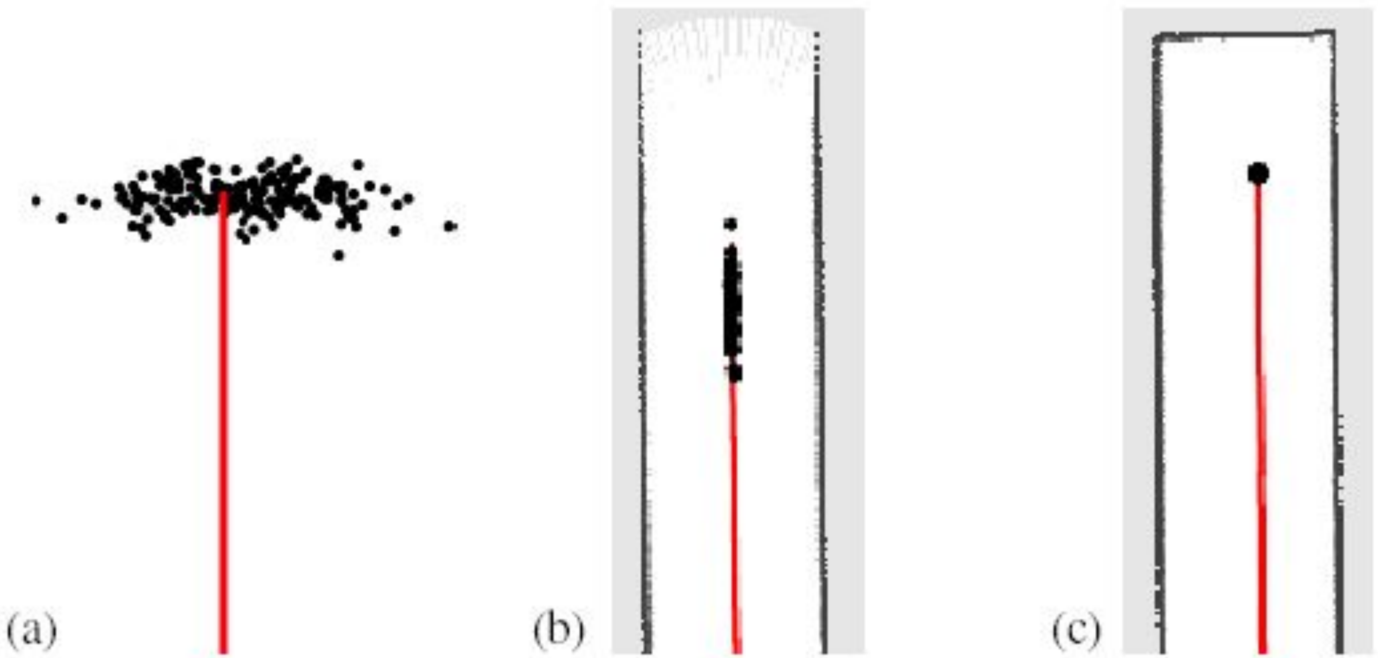
- Same observation models.
- Odometry instead of scan matching as input.
- Number of particles varying from 500 to 2000.
- Typical result (video).

Further Improvements

- Improved proposal distributions will lead to more accurate maps.
- They can be achieved by adapting the proposal distribution according to the most recent observations.
- Selective re-sampling steps can further improve the accuracy.

Improved Proposal

- The proposal adapts to the structure of the environment.
- Known measurements taken into account.



Selective Re-sampling

- During re-sampling important samples might get lost (particle depletion problem).
- In case of suboptimal proposal distributions re-sampling is necessary to achieve convergence.
- Key question: When should we re-sample?

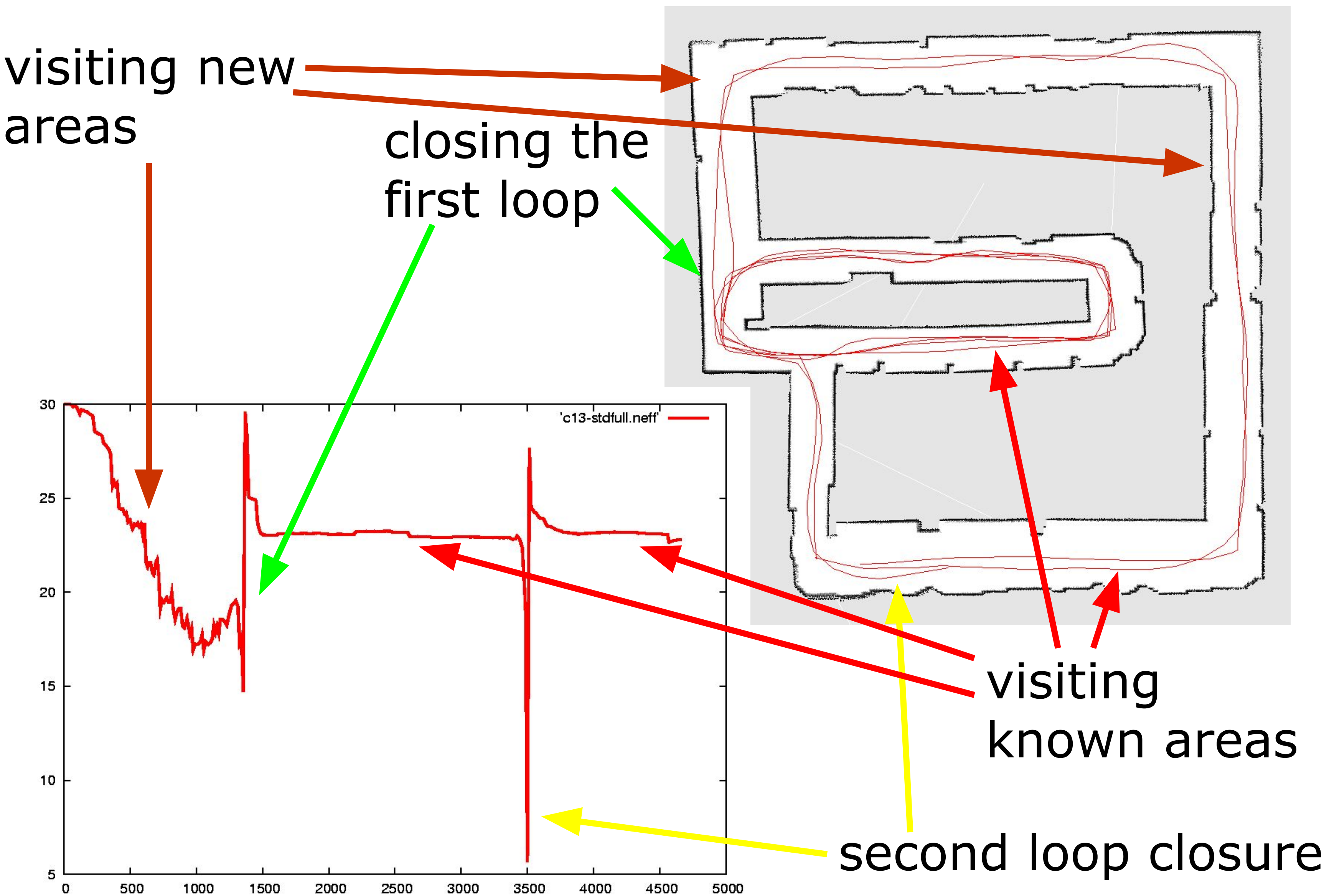
Number of Effective Particles

$$n_{eff} = \frac{1}{\sum_i \left(w_t^{(i)}\right)^2}$$

- Empirical measure of how well the goal distribution is approximated by samples drawn from the proposal.
- n_{eff} describes “the variance of the particle weights”.
- n_{eff} is maximal for equal weights. In this case, the distribution is close to the proposal.

- Only re-sample when n_{eff} drops below a given threshold (n/2)
  See [Doucet, ’98; Arulampalam, ’01]

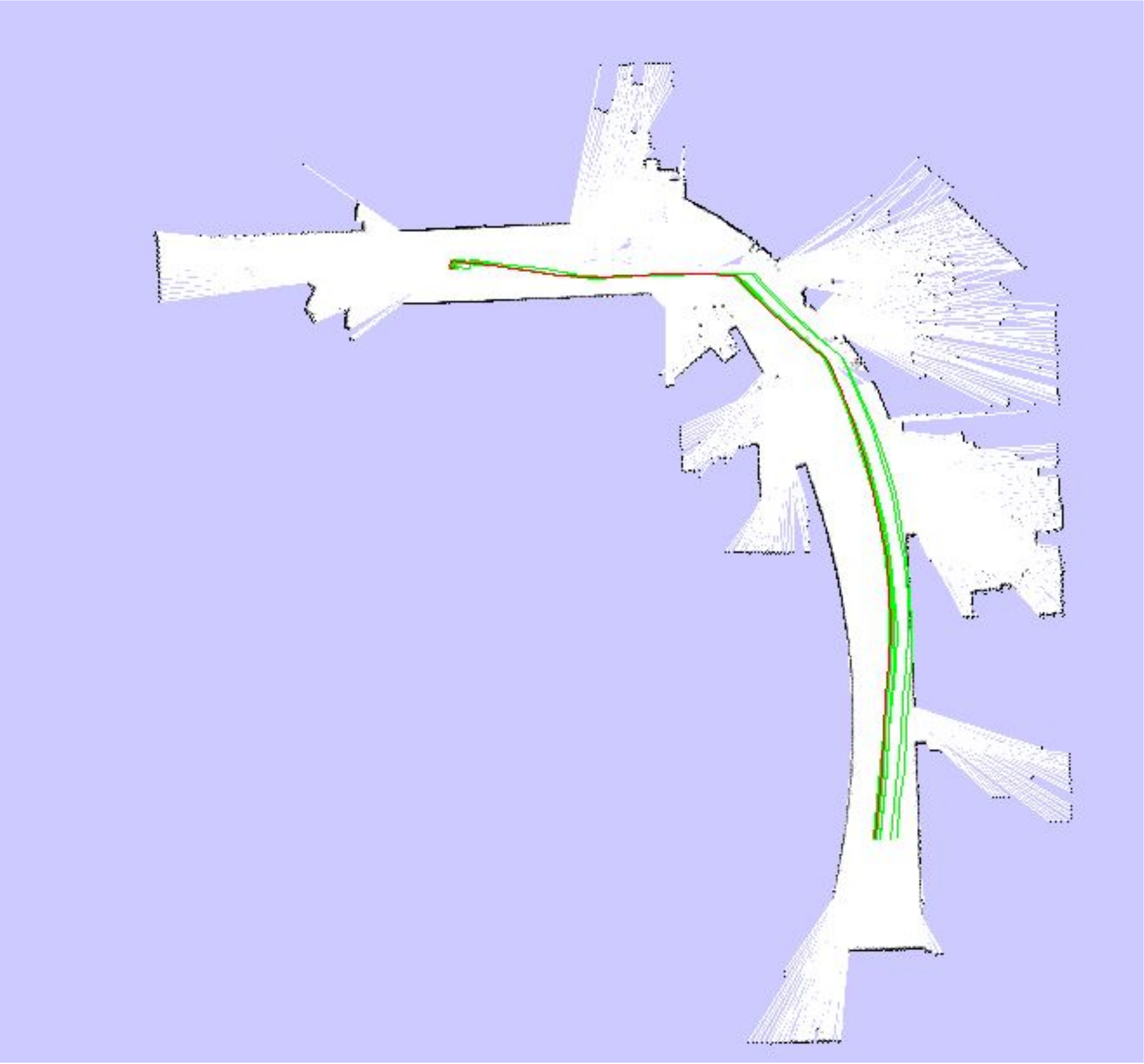
Typical Evolution of n_{eff}



Intel Lab

- **15 particles**
- four times faster than real-time P4, 2.8GHz
- 5cm resolution during scan matching
- 1cm resolution in final map

Intel Lab

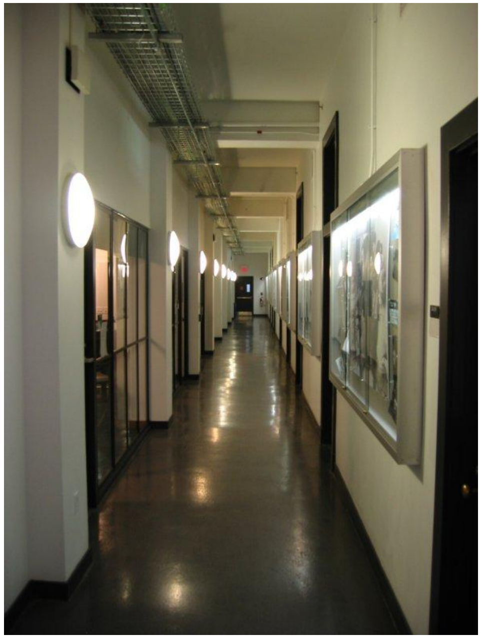


- **15 particles**
- Compared to FastSLAM with Scan-Matching, the particles are propagated closer to the true distribution

Outdoor Campus Map

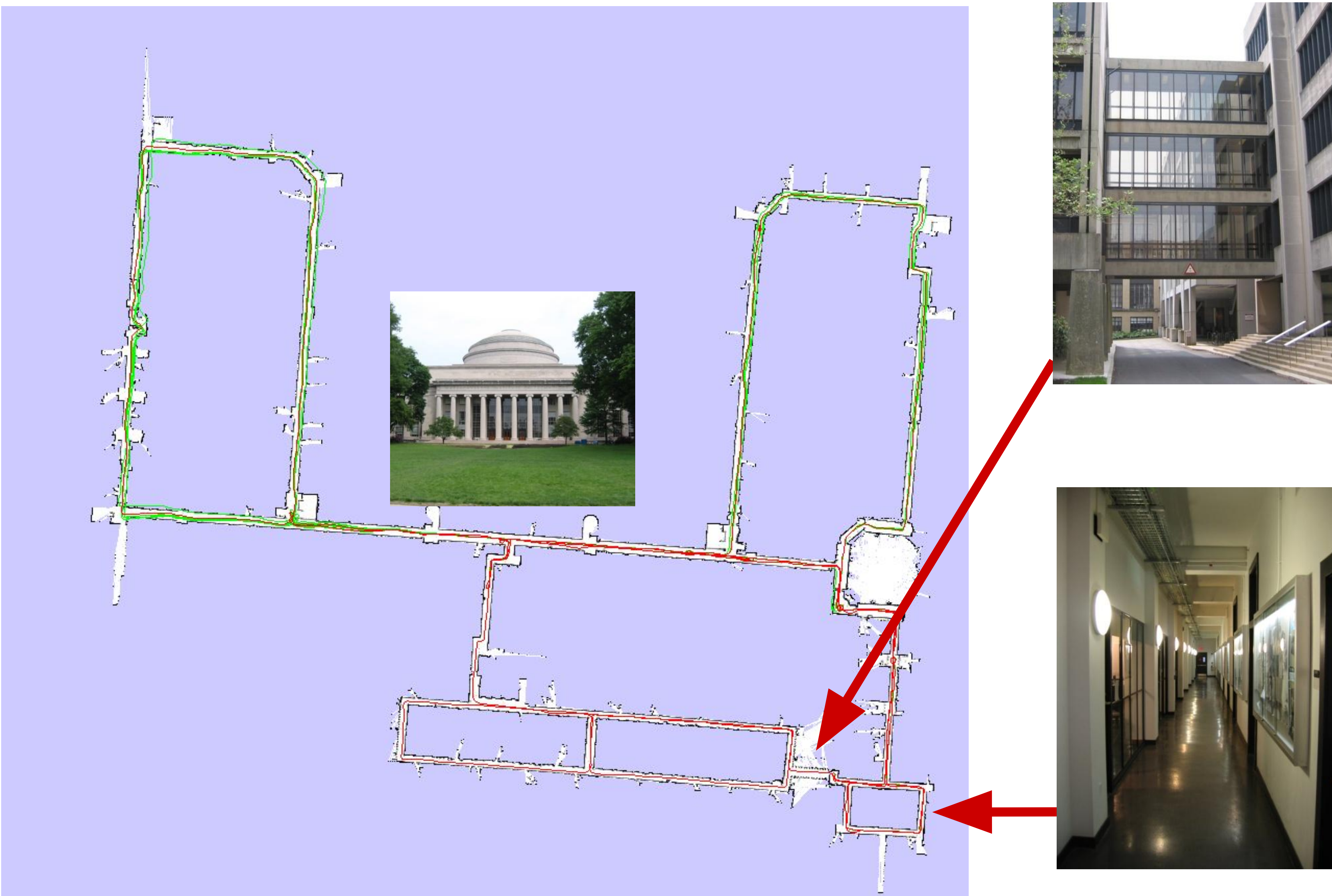
- **30 particles**
- 250x250m^2
- 1.088 miles (odometry)
- 20cm resolution during scan matching
- 30cm resolution in final map

MIT Killian Court



- The **"infinite-corridor-dataset"** at MIT.

MIT Killian Court



Conclusion

- The ideas of FastSLAM can also be applied in the context of grid maps.
- Utilizing accurate sensor observation leads to good proposals and highly efficient filters.
- It is similar to scan-matching on a per-particle basis.
- The number of necessary particles and re-sampling steps can seriously be reduced.
- Improved versions of grid-based FastSLAM can handle larger environments than naïve implementations in "real time" since they need order of magnitude fewer samples.