Aalto University
School of Electrical
Engineering

Probabilistic Robotics
Bayes Filter Implementations
FastSLAM

ELEC-E8111 Autonomous Mobile Robots

Arto Visala

7. 3.2018

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

The SLAM Problem

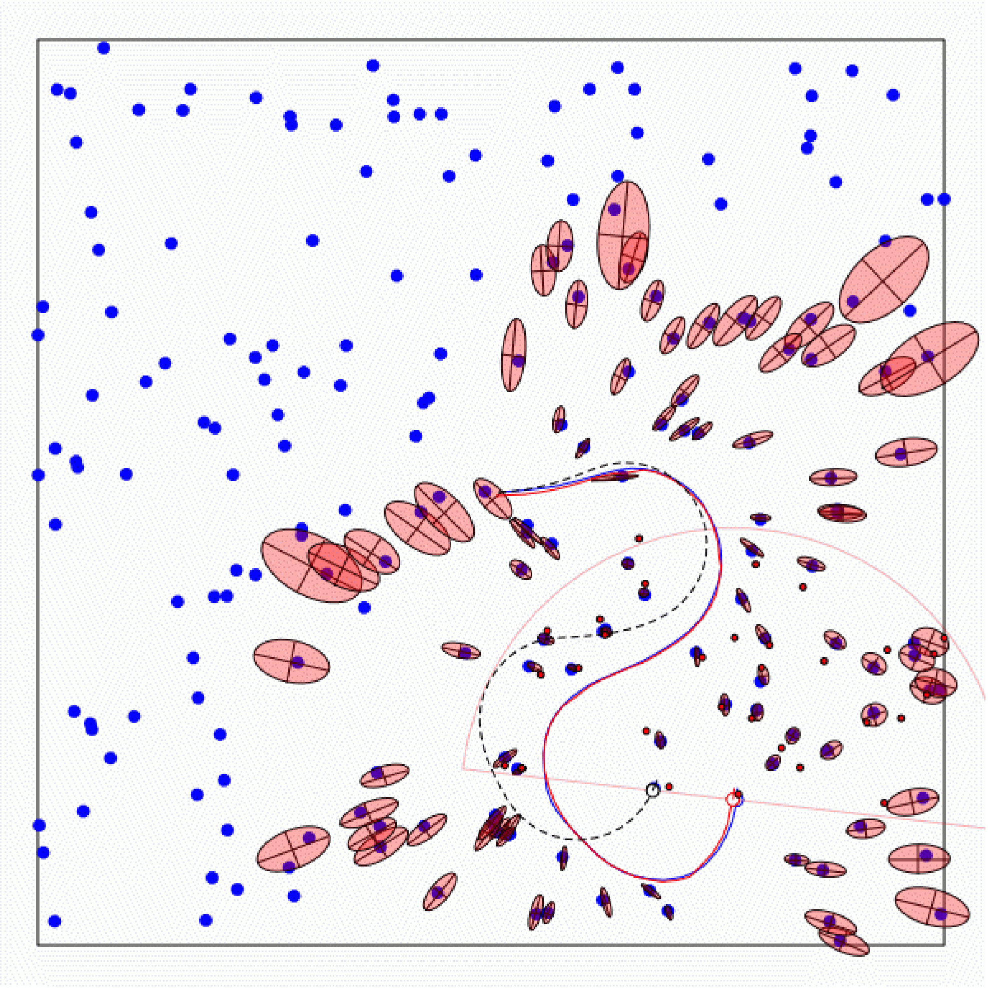
A robot moving though an unknown, static environment

Given:

- The robot's controls
- Observations of nearby features

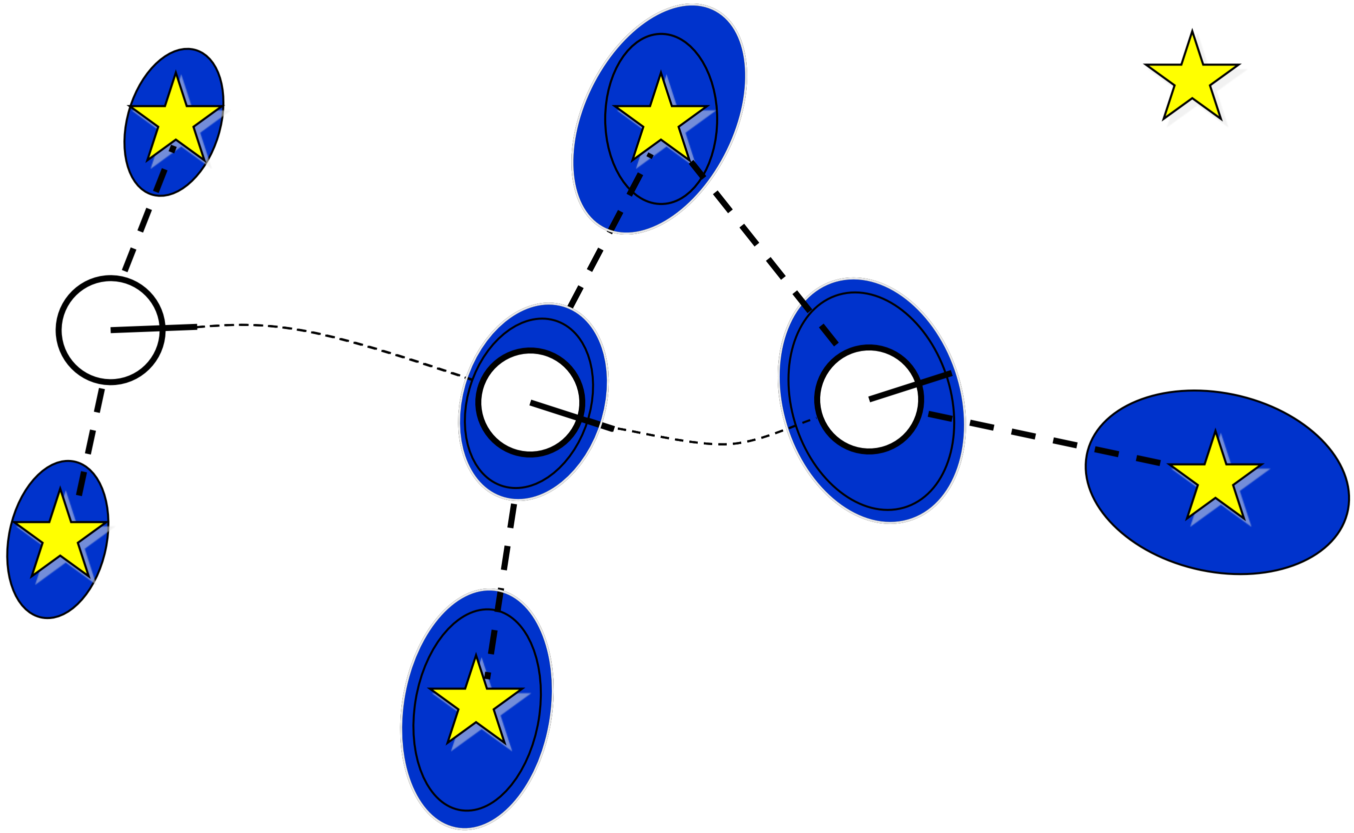
Estimate:

- Map of features
- Path of the robot



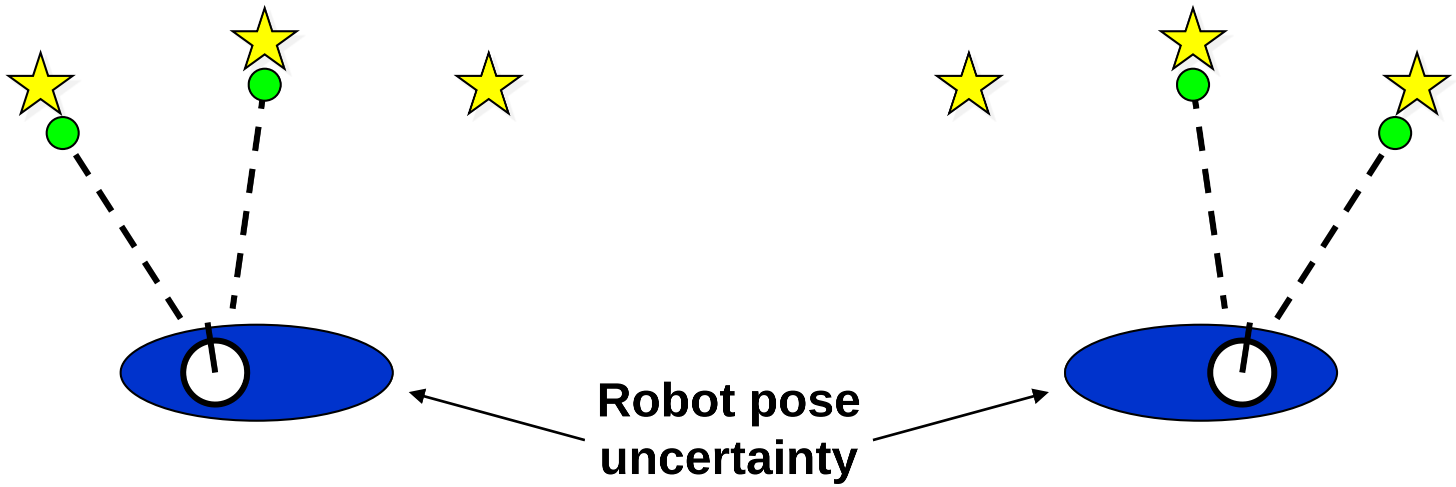
Why is SLAM a hard problem?

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



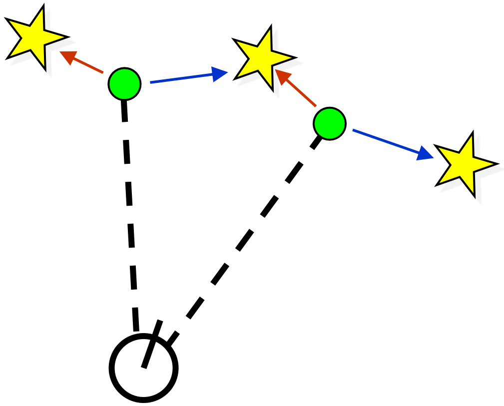
Robot path error correlates errors in the map

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



- 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, ...

Particle Filter algorithm 1/3

Algorithm Particle_filter($\mathcal{X}_{t-1}, u_t, z_t$):

$\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$

for $m = 1$ *to* M *do*

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

  $w_t^{[m]} = p(z_t \mid x_t^{[m]})$

  $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$

endfor

for $m = 1$ *to* M *do*

  draw i *with probability* $\propto w_t^{[i]}$

  add $x_t^{[i]}$ *to* $\mathcal{X}_t$

endfor

return $\mathcal{X}_t$

Particle Filter algorithm 2/3

1. Line 4 generates a hypothetical state $x_t^{[m]}$ for time t based on the particle $x_{t-1}^{[m]}$ and the control u_t. The resulting sample is indexed by m, indicating that it is generated from the m-th particle in $\mathcal{X}_{t-1}$. This step involves sampling from the state transition distribution $p(x_t \mid u_t, x_{t-1})$. To implement this step, one needs to be able to sample from this distribution. The set of particles obtained after M iterations is the filter's representation of $\overline{bel}(x_t)$.

2. Line 5 calculates for each particle $x_t^{[m]}$ the so-called *importance factor*, denoted $w_t^{[m]}$. Importance factors are used to incorporate the measurement z_t into the particle set. The importance, thus, is the probability of the measurement z_t under the particle $x_t^{[m]}$, given by $w_t^{[m]} = p(z_t \mid x_t^{[m]})$. If we interpret $w_t^{[m]}$ as the *weight* of a particle, the set of weighted particles represents (in approximation) the Bayes filter posterior $bel(x_t)$.

Particle Filter algorithm 3/3

3. The real "trick" of the particle filter algorithm occurs in lines 8 through 11 in Table 4.3. These lines implemented what is known as *resampling* or *importance sampling*. The algorithm draws with replacement M particles from the temporary set $\bar{\mathcal{X}}_t$. The probability of drawing each particle is given by its importance weight. Resampling transforms a particle set of M particles into another particle set of the same size. By incorporating the importance weights into the resampling process, the distribution of the particles change: Whereas before the resampling step, they were distributed according to $\overline{bel}(x_t)$, after the resampling they are distributed (approximately) according to the posterior $bel(x_t) = \eta\, p(z_t \mid x_t^{[m]})\overline{bel}(x_t)$. In fact, the resulting sample set usually possesses many duplicates, since particles are drawn with replacement. More important are the particles *not* contained in $\mathcal{X}_t$: Those tend to be the particles with lower importance weights.

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

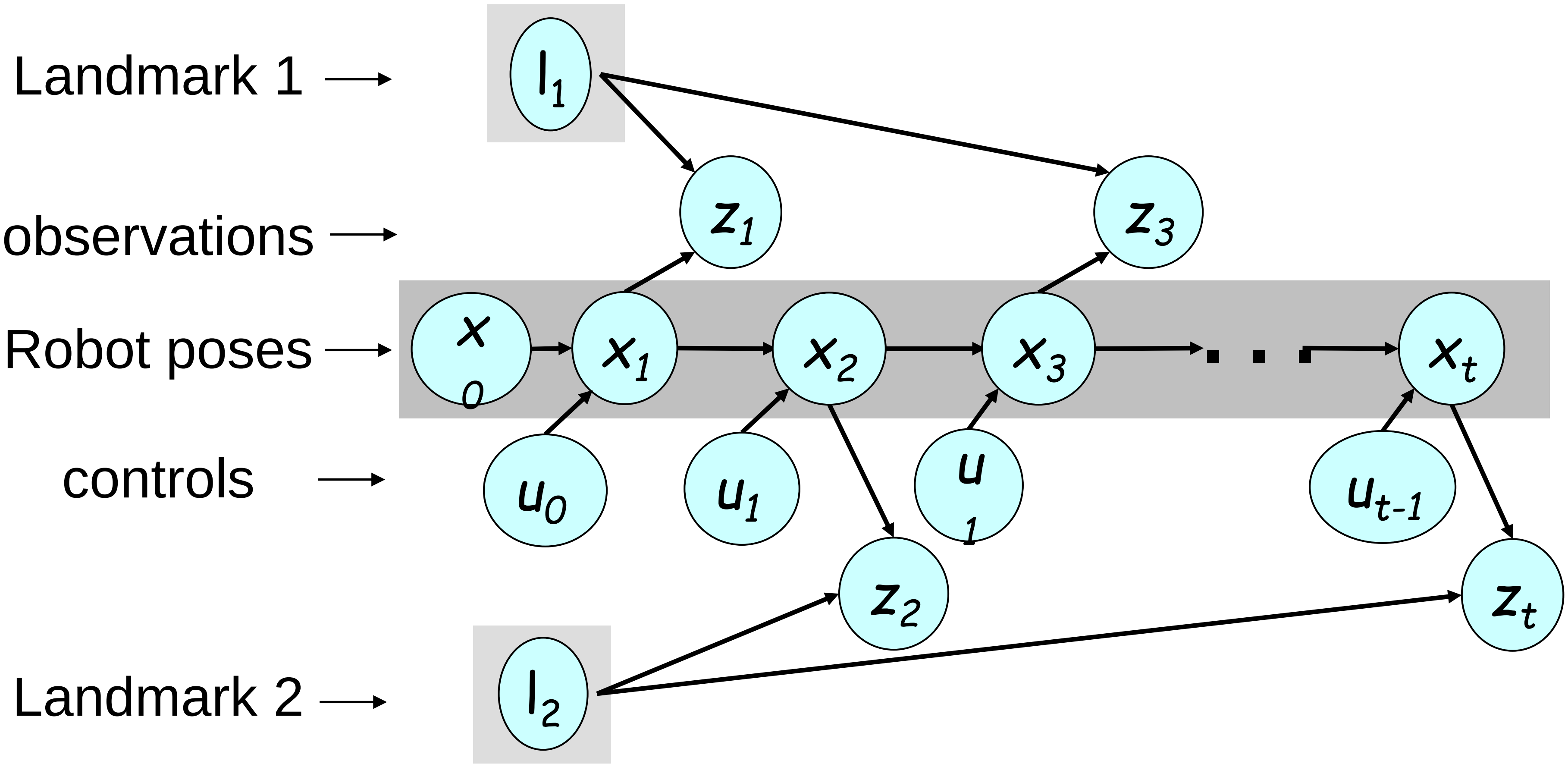
Robot path posterior

landmark positions

Does this help to solve the problem?

Factorization first introduced by Murphy in 1999

Mapping using Landmarks



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

Factored Posterior

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

Robot path posterior (localization problem)

Conditionally independent landmark positions

Rao-Blackwellization

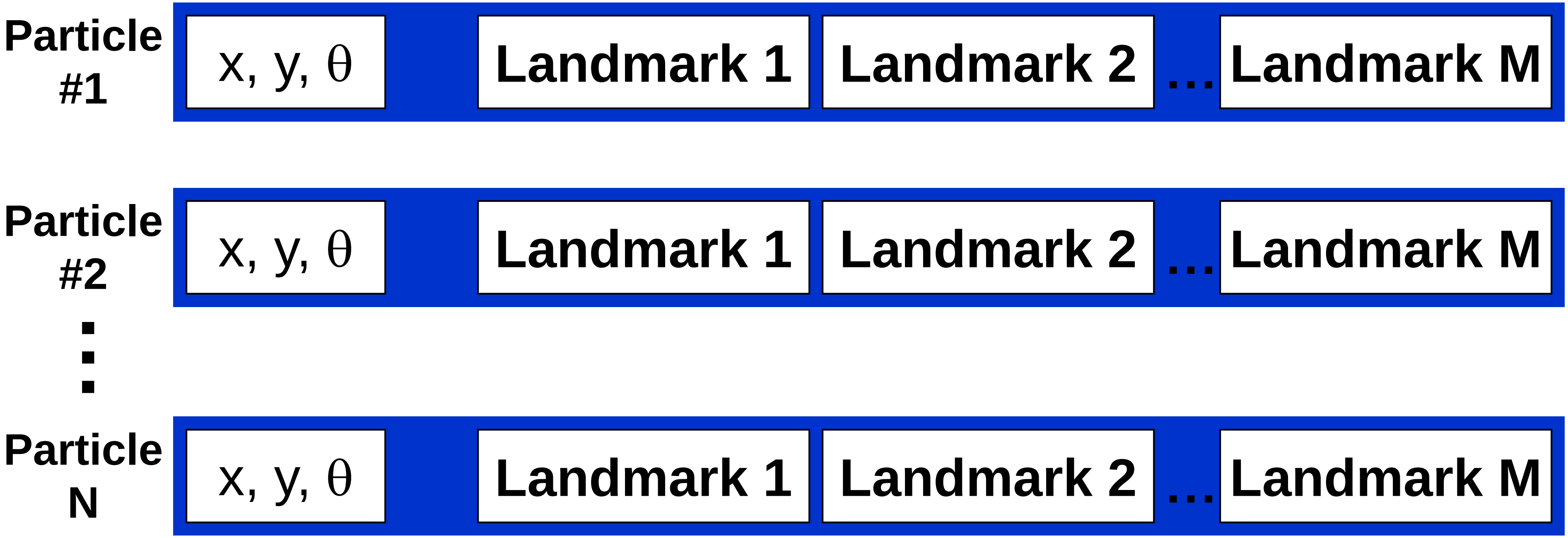
$$p(x_{1:t}, l_{1:m} \mid z_{1:t}, u_{0:t-1}) =$$

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

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

FastSLAM

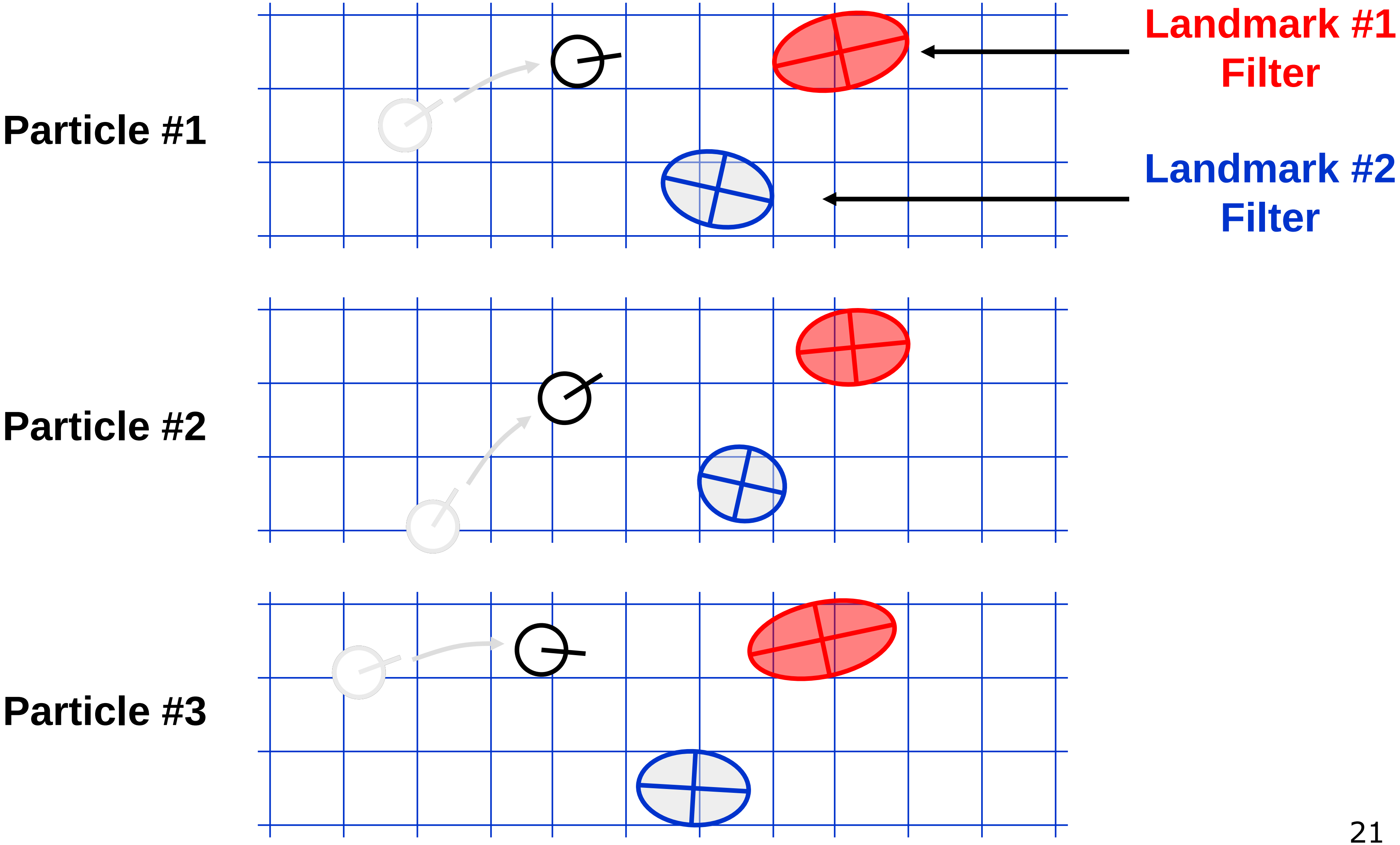
- Rao-Blackwellized particle filtering based on landmarks [Montemerlo et al., 2002]
- Each landmark is represented by a 2x2 Extended Kalman Filter (EKF)
- Each particle therefore has to maintain *M* EKFs



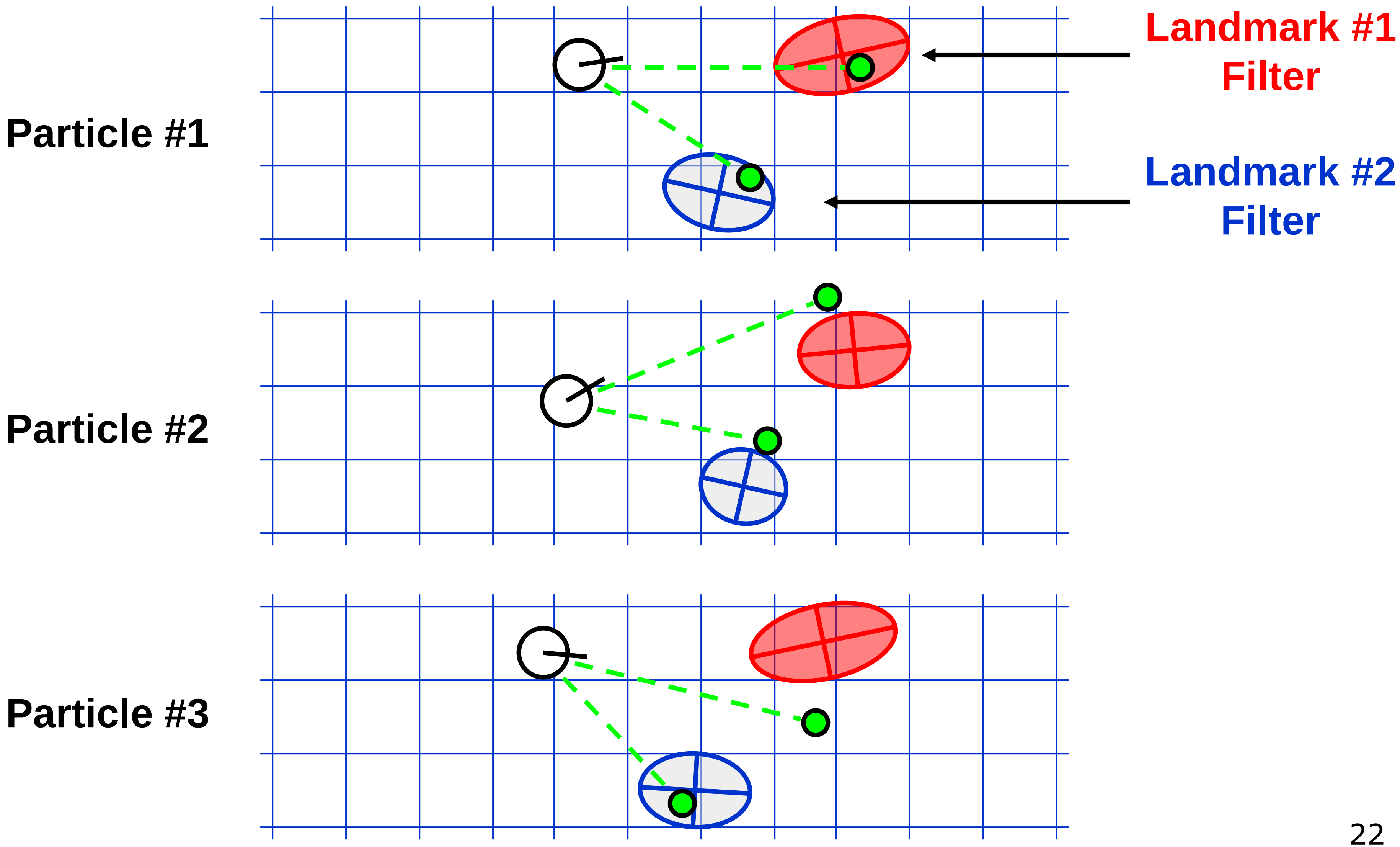
FastSLAM algorithm, feature = landmark

- Do the following M times:
  - **Retrieval.** Retrieve a pose $x_{t-1}^{[k]}$ from the particle set Y_{t-1}.
  - **Prediction.** Sample a new pose $x_t^{[k]} \sim p(x_t \mid x_{t-1}^{[k]}, u_t)$.
  - **Measurement update.** For each observed feature z_t^i identify the correspondence j for the measurement z_t^i, and incorporate the measurement z_t^i into the corresponding EKF, by updating the mean $\mu_{j,t}^{[k]}$ and covariance $\Sigma_{j,t}^{[k]}$.
  - **Importance weight.** Calculate the importance weight $w^{[k]}$ for the new particle.
- **Resampling.** Sample, with replacement, M particles, where each particle is sampled with a probability proportional to $w^{[k]}$.

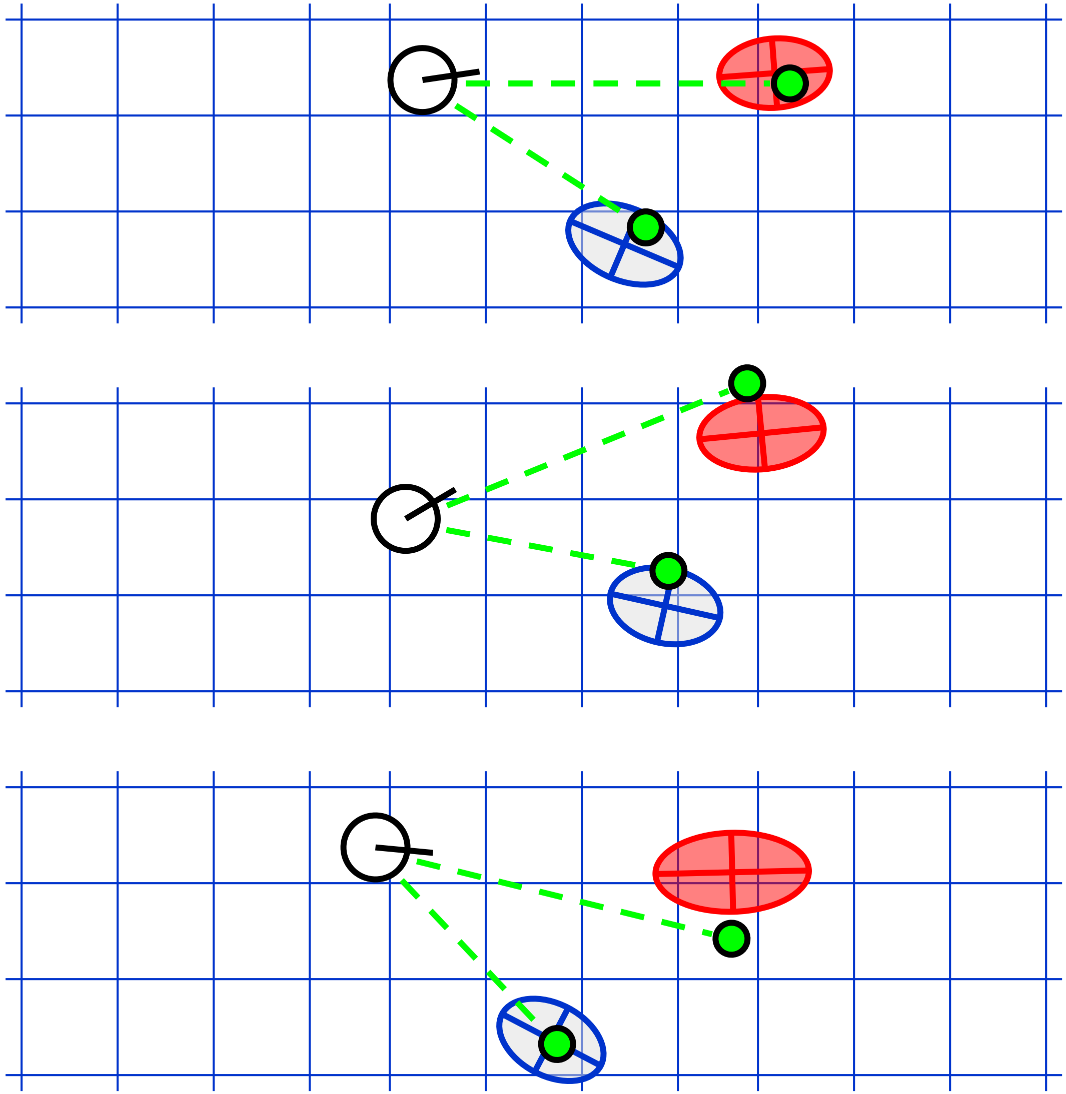
FastSLAM – Action Update



FastSLAM – Sensor Update



FastSLAM – Sensor Update



Particle #1 **Weight = 0.8**

Particle #2 **Weight = 0.4**

Particle #3 **Weight = 0.1**

FastSLAM 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

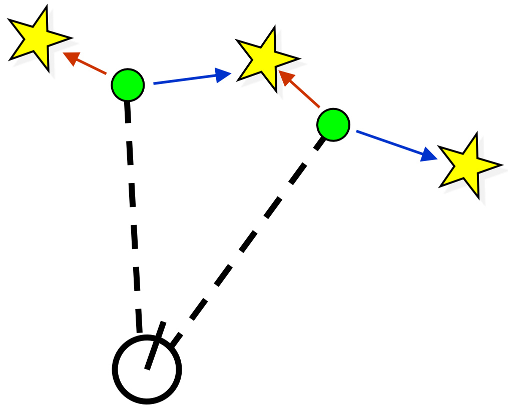
  $O(N \cdot \log(M))$
  Log time per particle

N = Number of particles
M = Number of map features

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

Data Association Problem

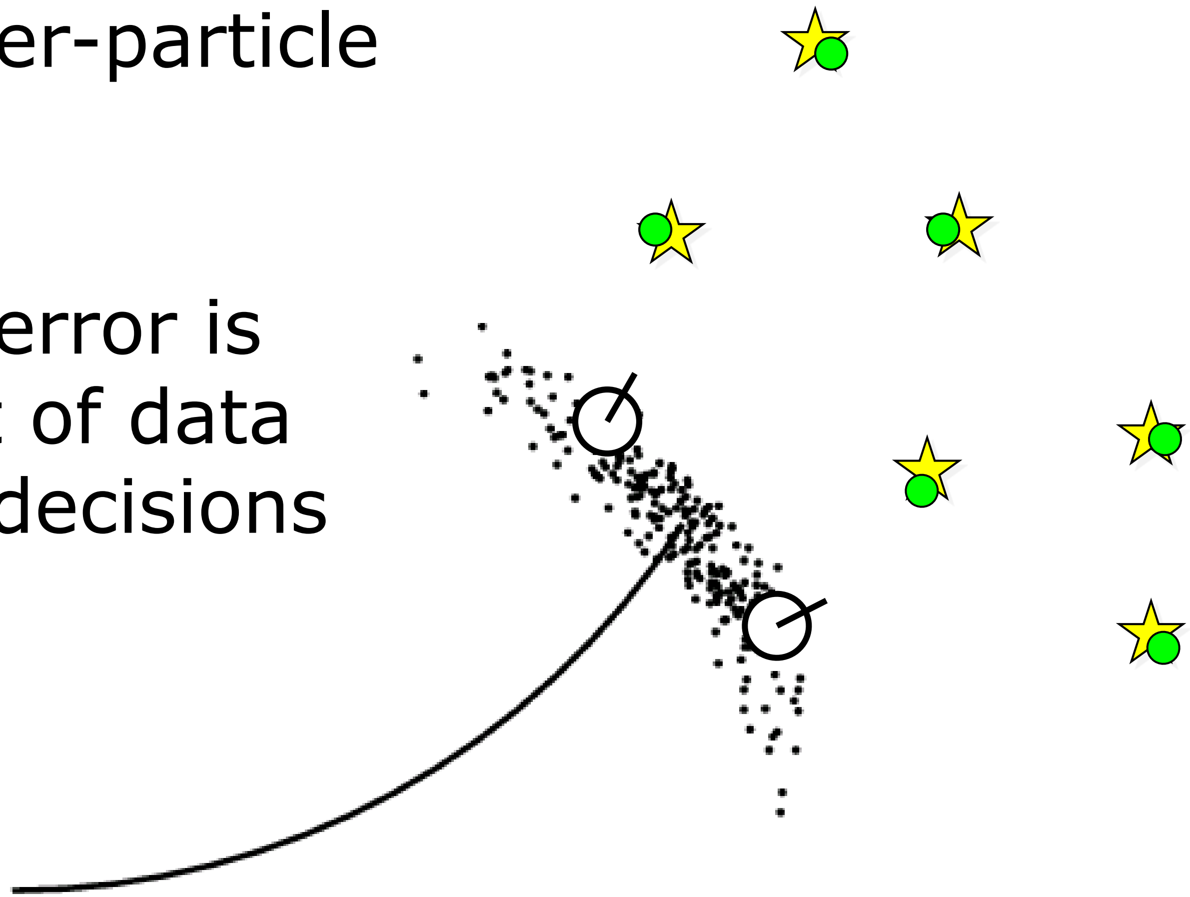
- Which observation belongs to which landmark?



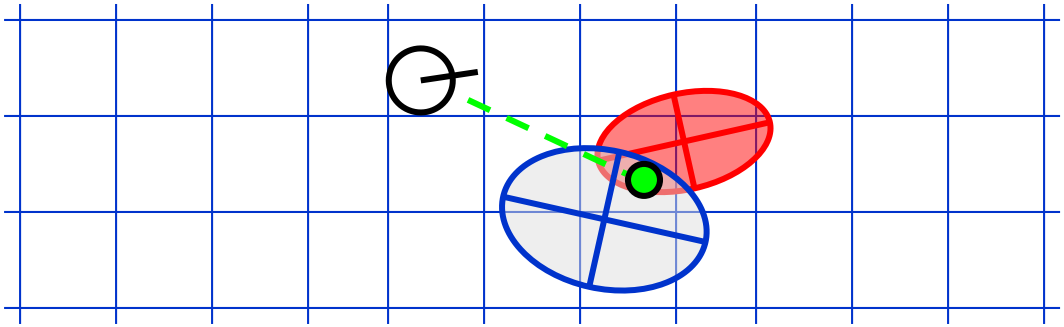
- A robust SLAM must consider possible data associations
- Potential data associations depend also on the pose of the robot

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



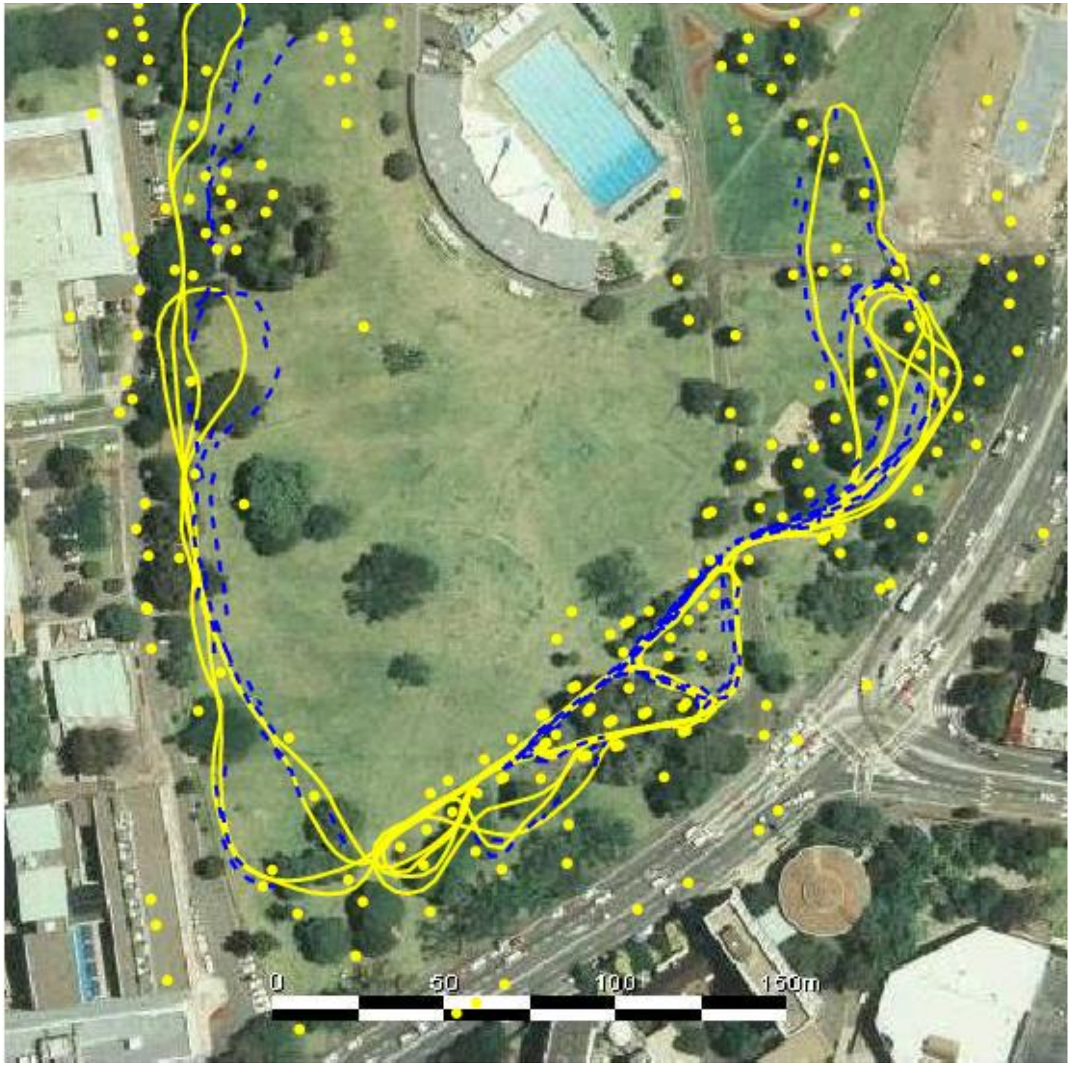
Was the observation generated by the red or the blue landmark?

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

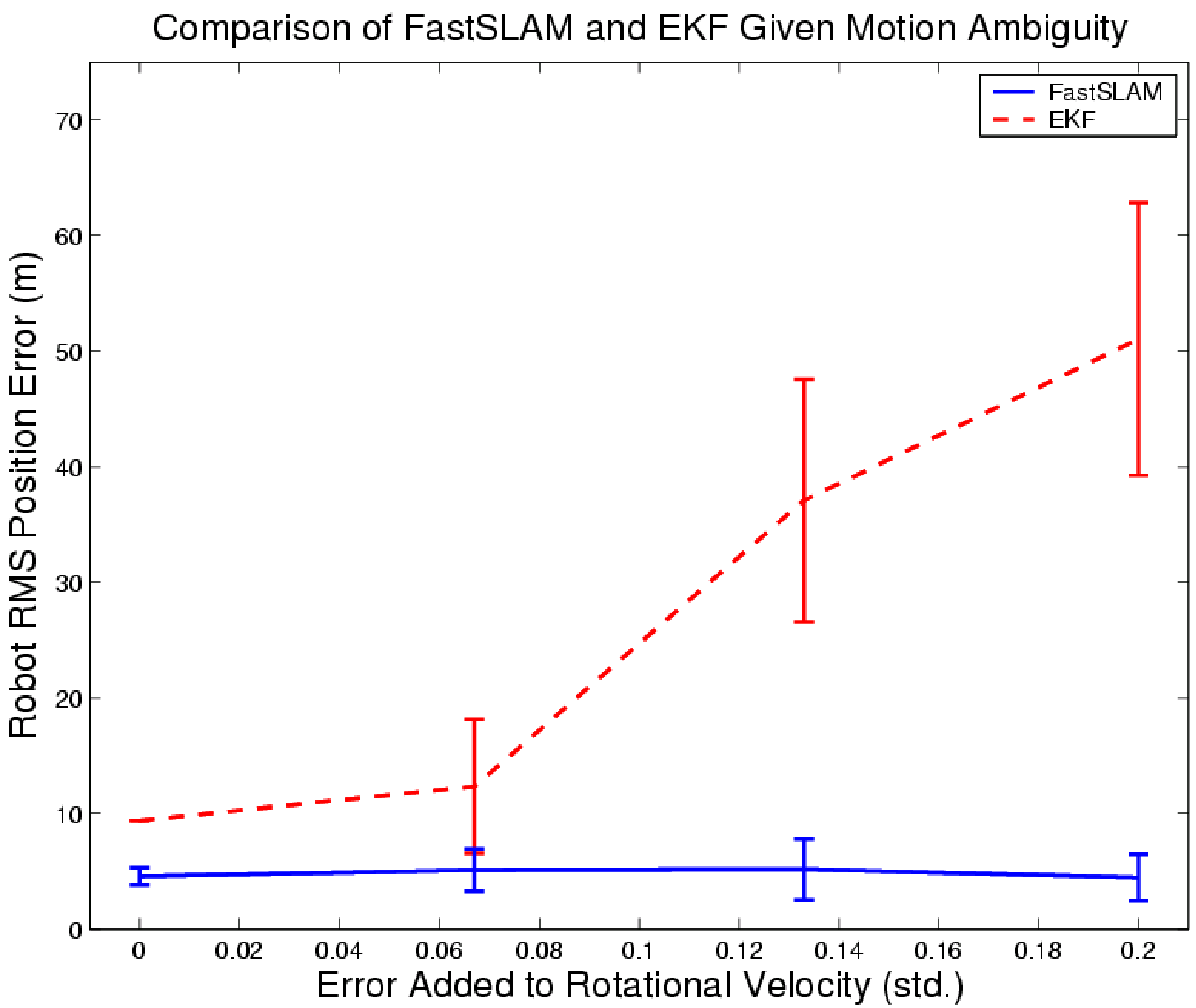
- 4 km traverse
- < 5 m RMS position error
- 100 particles



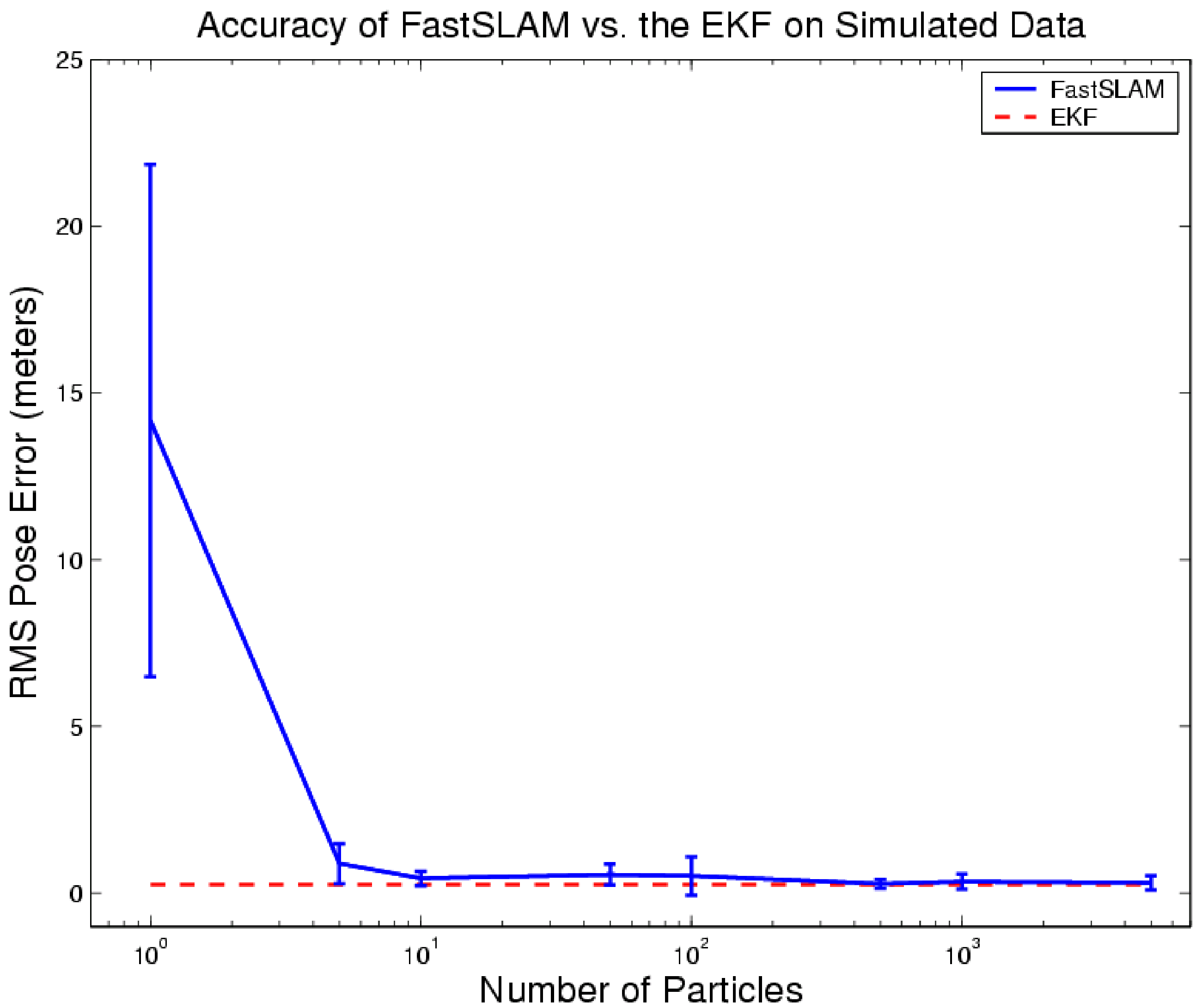
Blue = GPS
Yellow = FastSLAM

Dataset courtesy of University of Sydney

Results – Data Association



Results – Accuracy



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-Blackwellization

poses map observations & movements

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

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

Factorization first introduced by Murphy in 1999

Rao-Blackwellization

poses map observations & movements

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

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

SLAM posterior

Robot path posterior

Mapping with known poses

Factorization first introduced by Murphy in 1999

Rao-Blackwellization

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

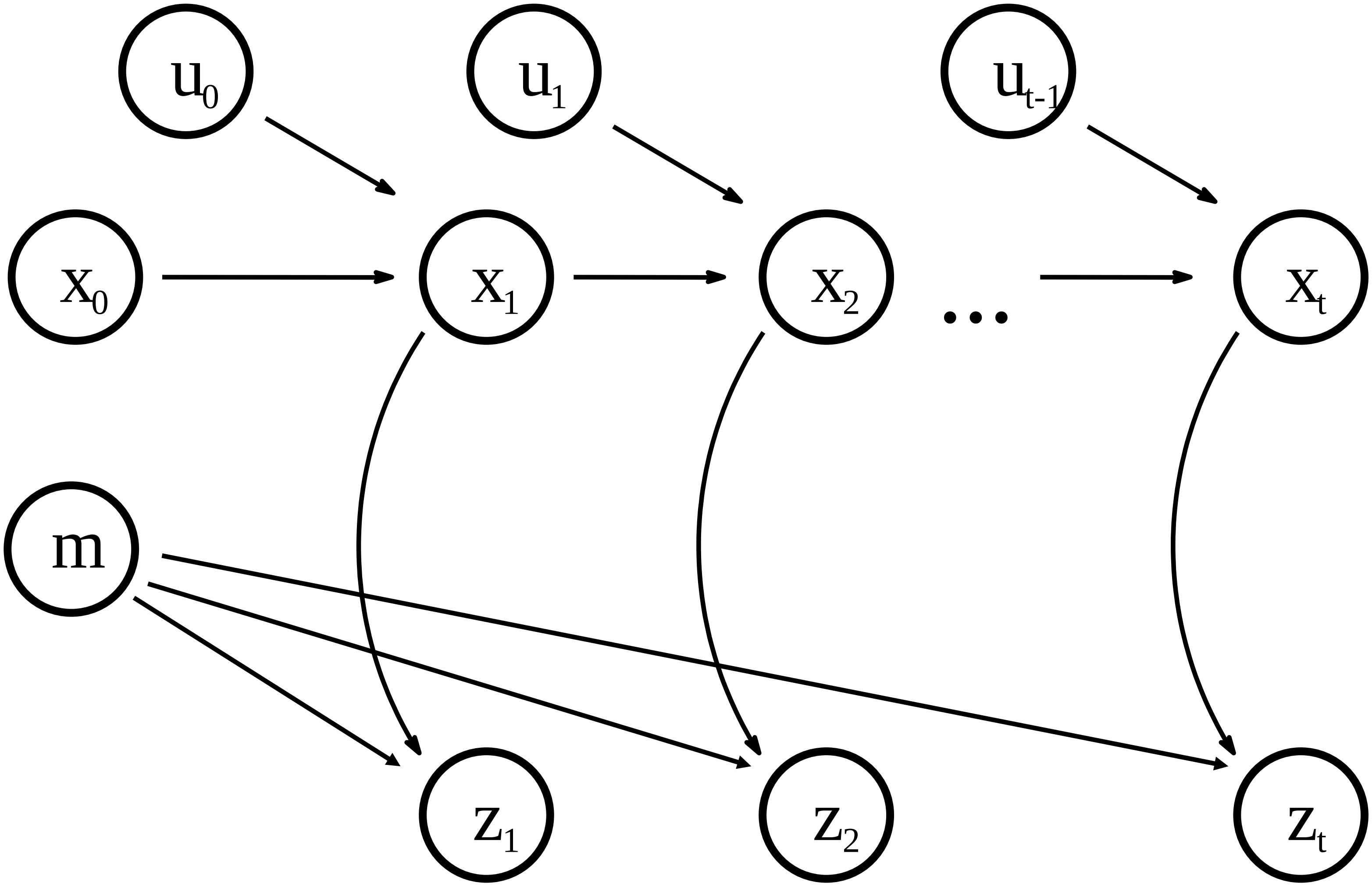
This is localization, use MCL

Use the pose estimate from the MCL part and apply mapping with known poses

MCL Monte Carlo Localization

Algorithm MCL($\mathcal{X}_{t-1}, u_t, z_t, m$)**:**
  $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
  for $m = 1$ *to* M *do*
    $x_t^{[m]} =$ **sample_motion_model**$(u_t, x_{t-1}^{[m]})$
    $w_t^{[m]} =$ **measurement_model**$(z_t, x_t^{[m]}, m)$
    $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
  endfor
  for $m = 1$ *to* M *do*
    draw i *with probability* $\propto w_t^{[i]}$
    add $x_t^{[i]}$ *to* $\mathcal{X}_t$
  endfor
  return $\mathcal{X}_t$

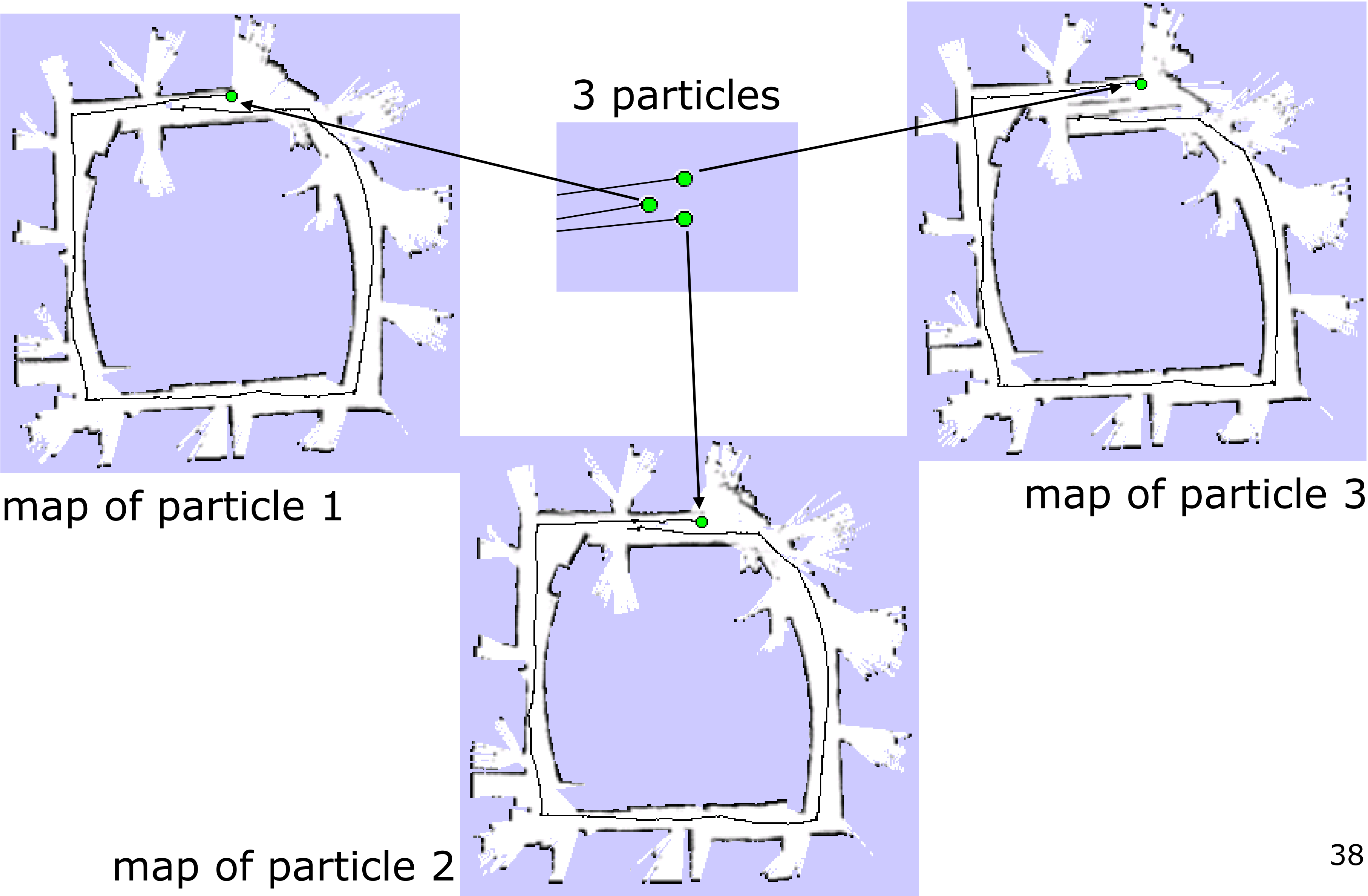
A Graphical Model of Rao-Blackwellized Mapping



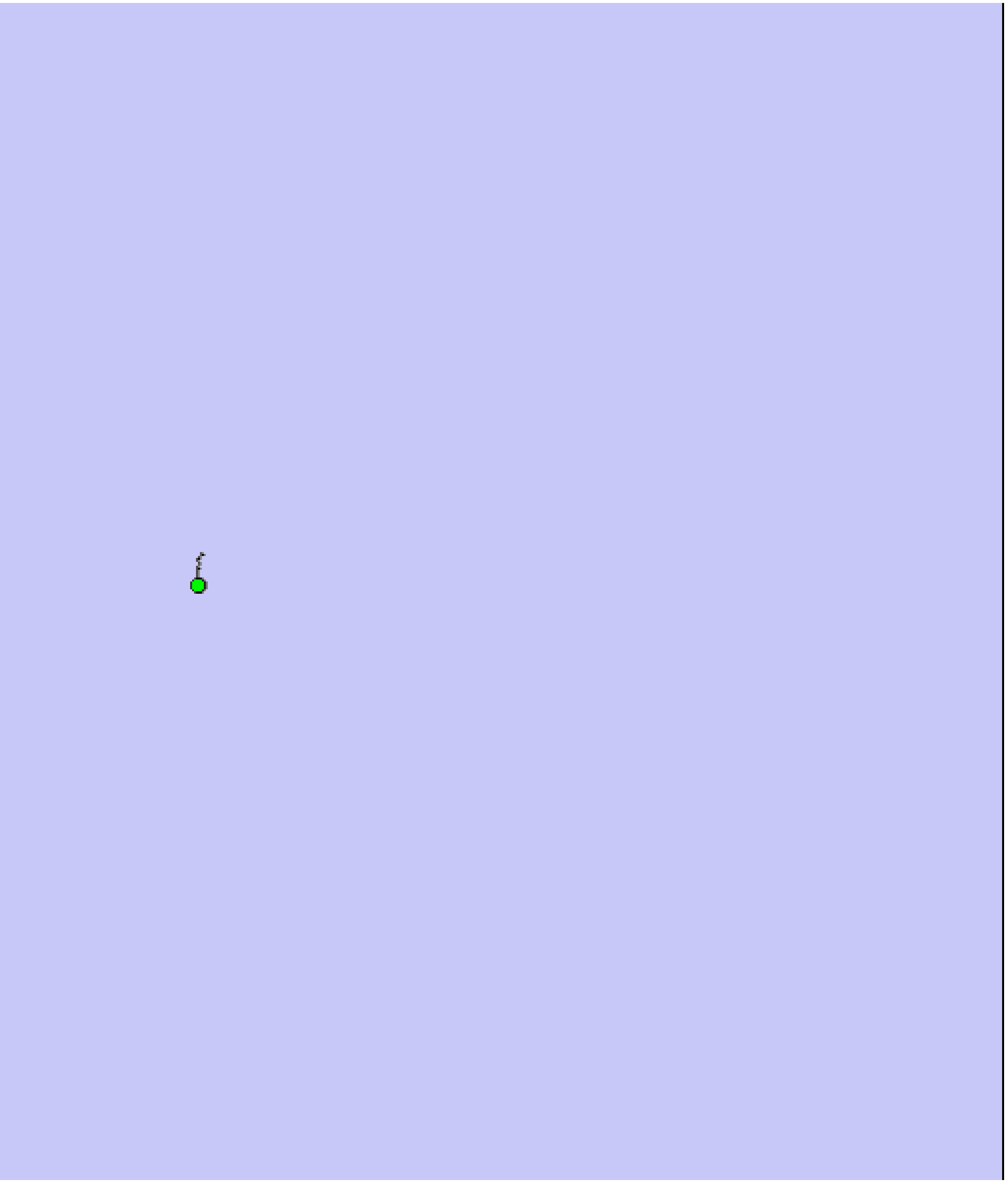
Rao-Blackwellized Mapping

- Each particle represents a possible trajectory of the robot
- Each particle
  - maintains its own map and
  - updates it upon “mapping with known poses”
- Each particle survives with a probability proportional to the likelihood of the observations relative to its own map

Particle Filter Example



Occupancy grid Fast SLAM



Problem

- Each map is quite big in case of grid maps
- Since each particle maintains its own map
- Therefore, one needs 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

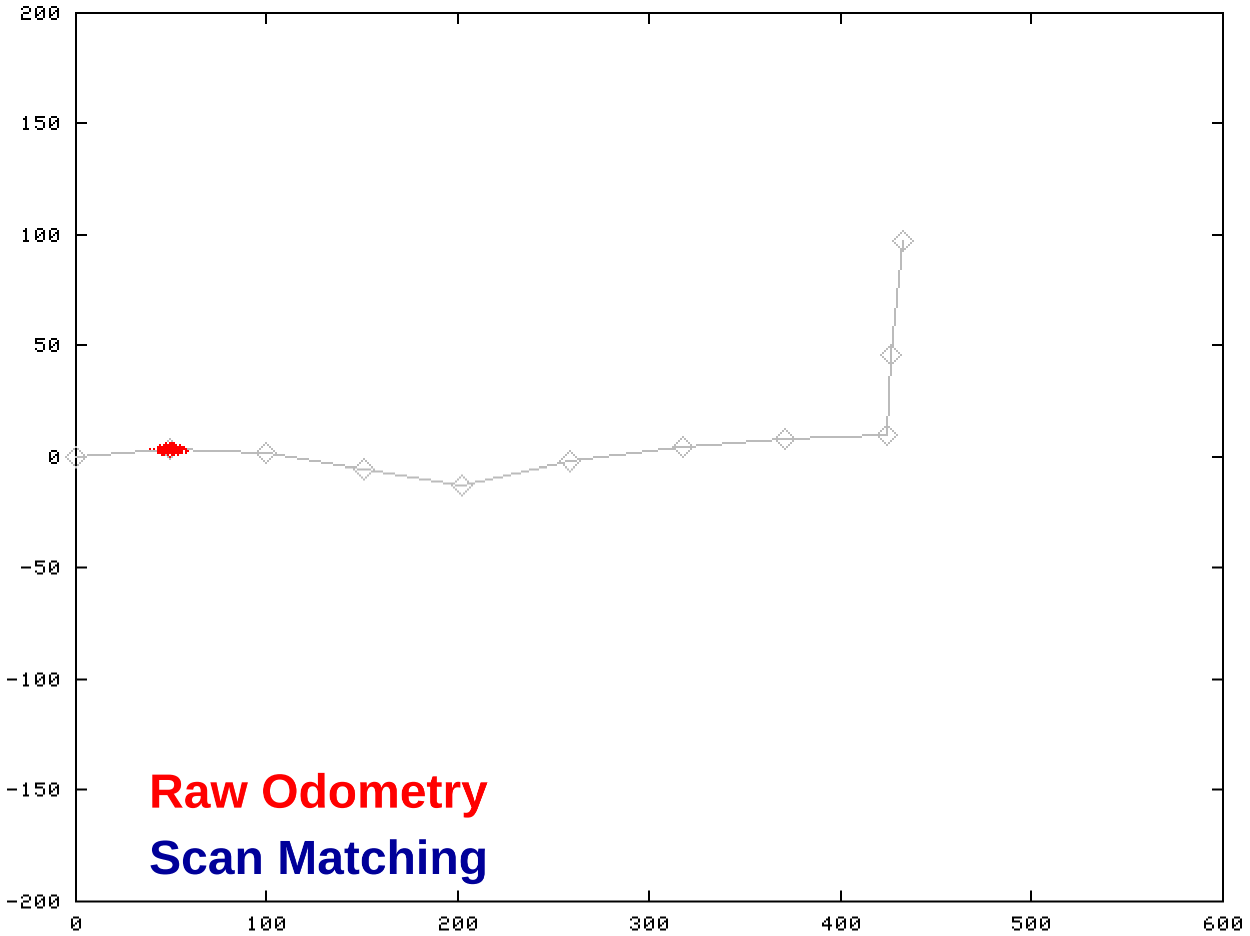
$$\hat{x}_t = \underset{x_t}{\arg\max} \left\{ p(z_t \mid x_t, \hat{m}_{t-1}) \cdot p(x_t \mid u_{t-1}, \hat{x}_{t-1}) \right\}$$

current measurement

robot motion

map constructed so far

Motion Model for Scan Matching

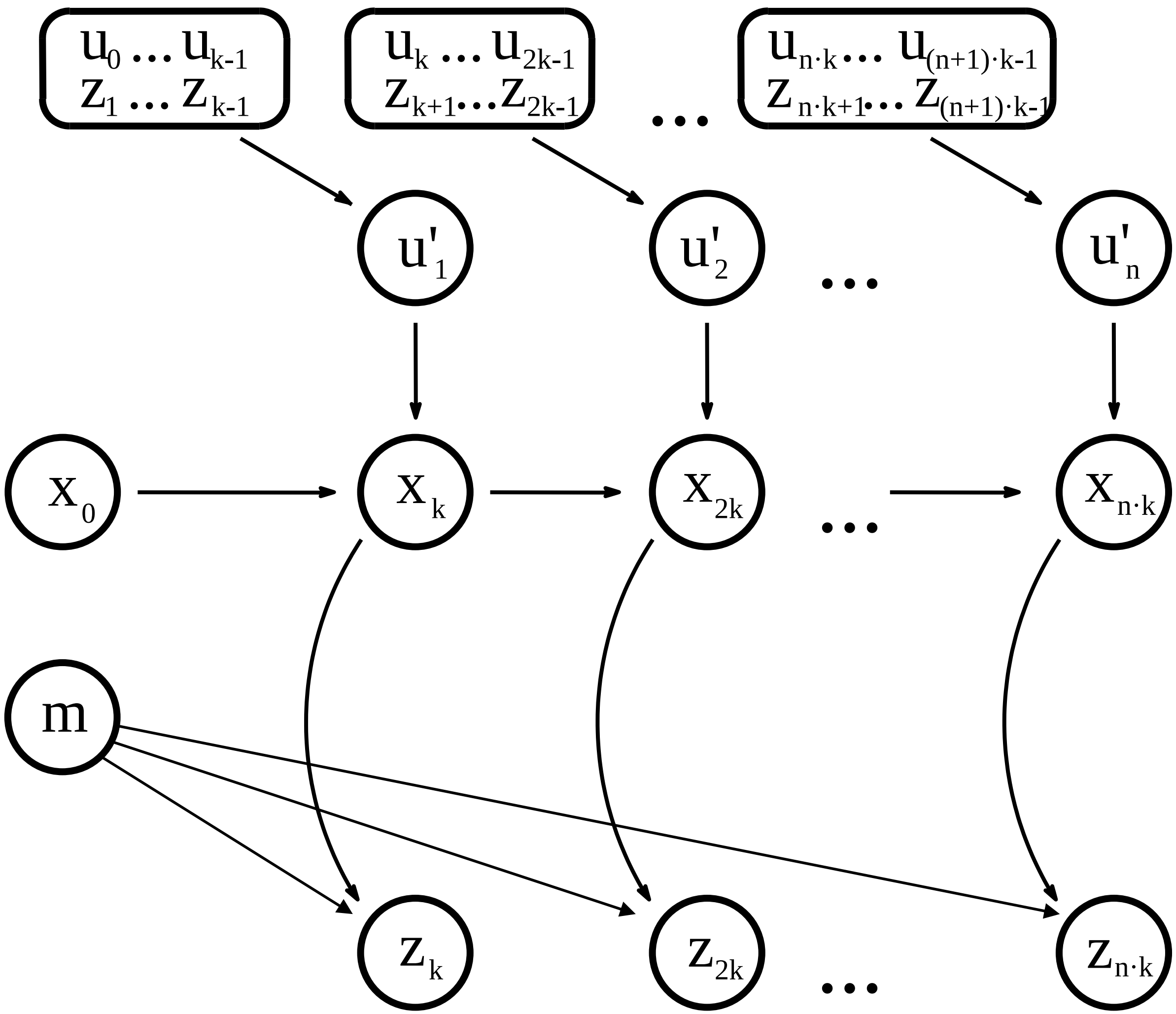


FastSLAM with Improved Odometry

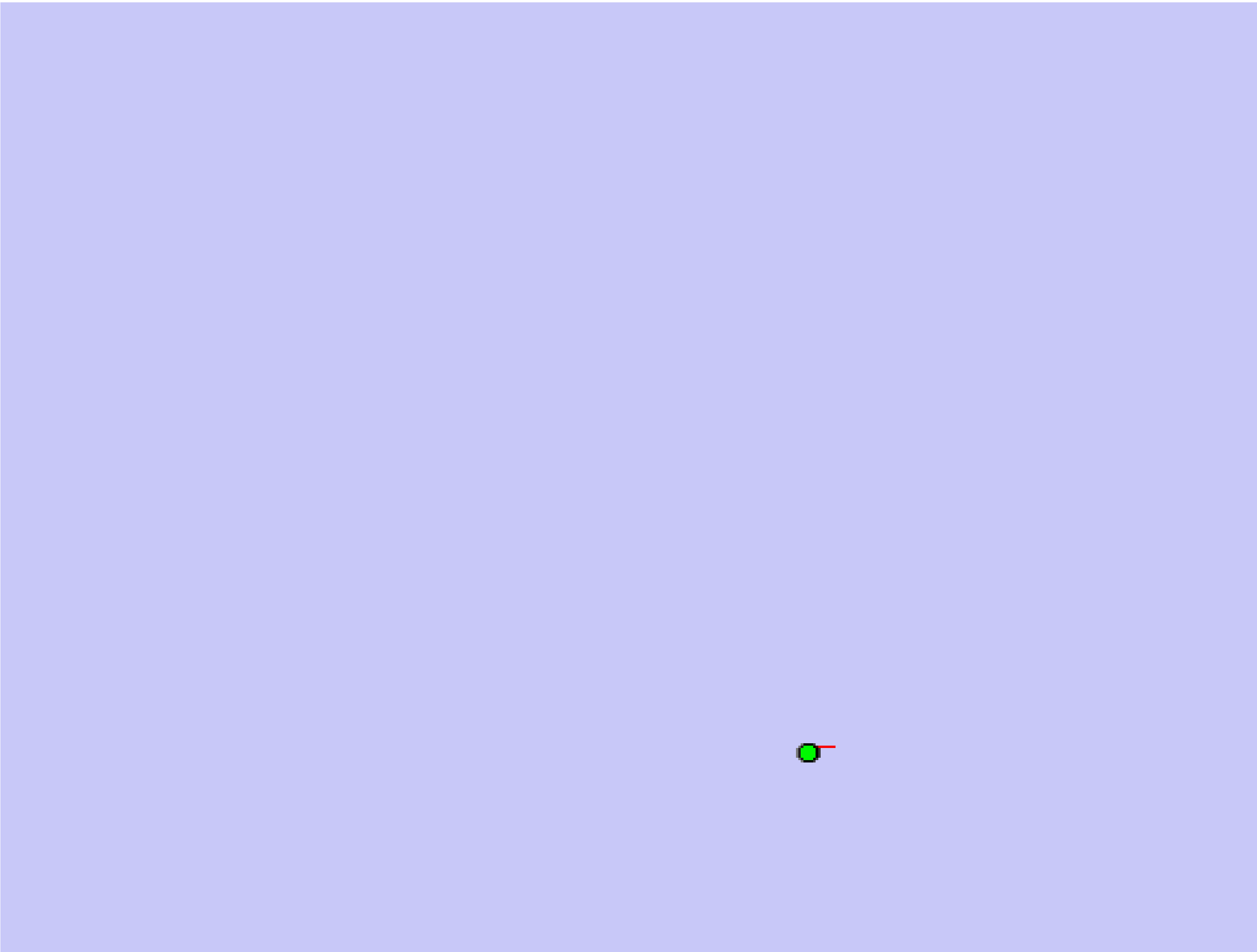
- Scan-matching provides a **locally consistent** pose correction
- Pre-correct short odometry sequences using scan-matching and use them as input to FastSLAM
- Fewer particles are needed, since the error in the input in smaller

[Haehnel et al., 2003]

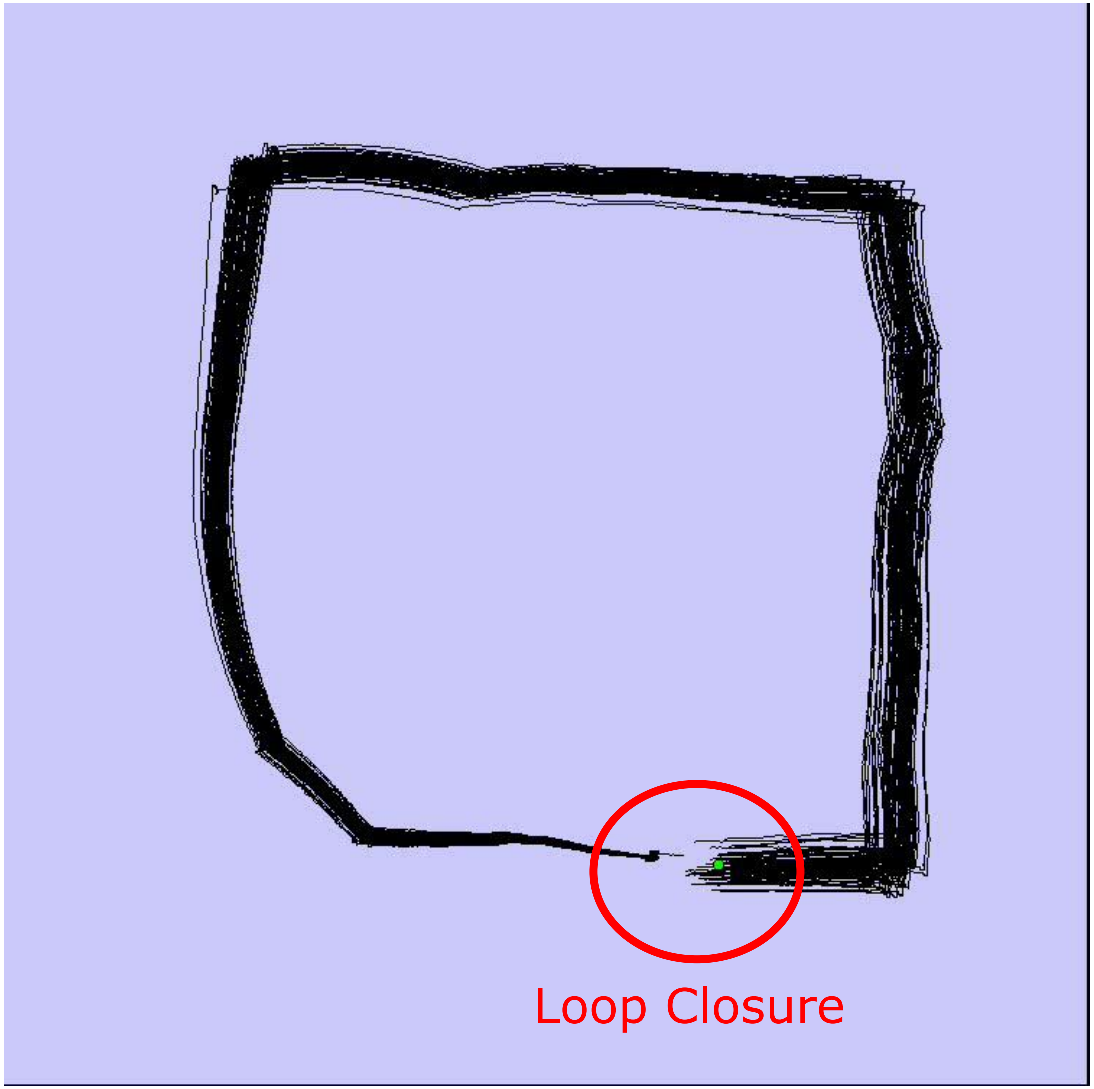
Graphical Model for Mapping with Improved Odometry



FastSLAM with Scan-Matching



FastSLAM with Scan-Matching

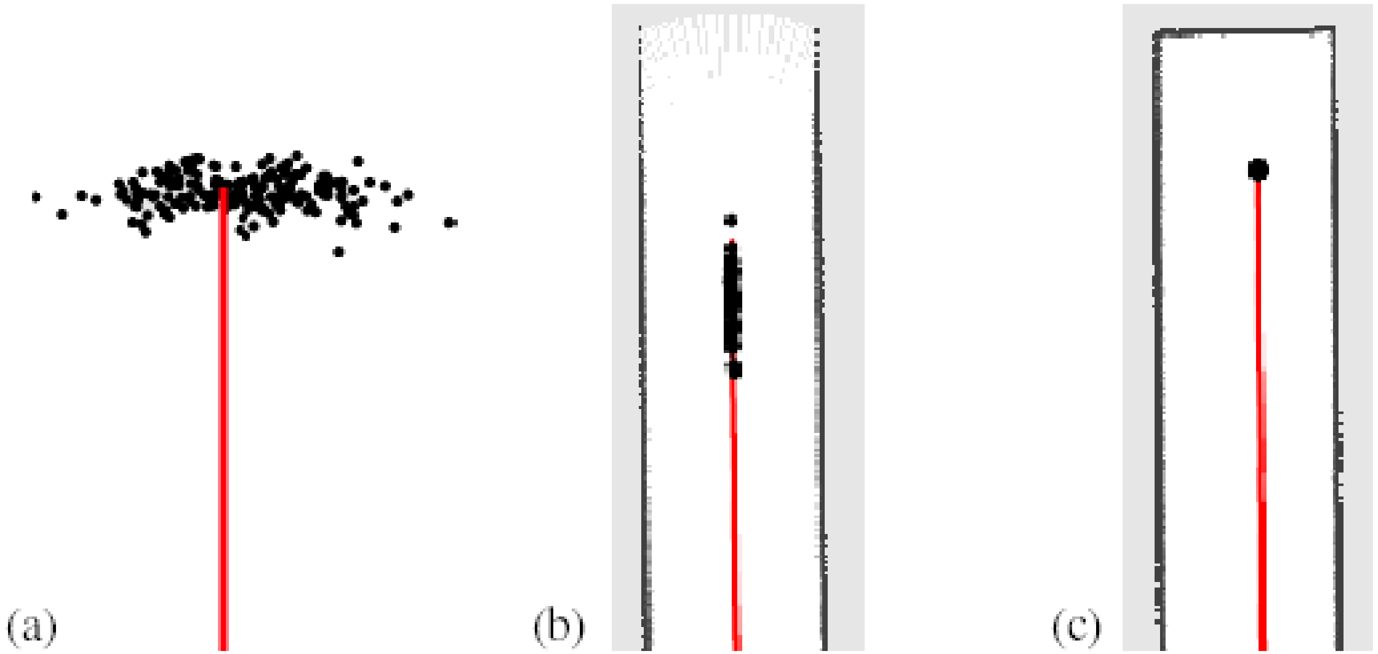


Further Improvements

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

Improved Proposal

- The proposal adapts to the structure of the environment



Selective Re-sampling

- Re-sampling is dangerous, since 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?

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 base
- 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 one order of magnitude fewer samples

More Details on FastSLAM

- M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to simultaneous localization and mapping, *AAAI02*
- D. Haehnel, W. Burgard, D. Fox, and S. Thrun. An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements, IROS03
- M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit. FastSLAM 2.0: An Improved particle filtering algorithm for simultaneous localization and mapping that provably converges. IJCAI-2003
- G. Grisetti, C. Stachniss, and W. Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling, ICRA05
- A. Eliazar and R. Parr. DP-SLAM: Fast, robust simultanous localization and mapping without predetermined landmarks, IJCAI03