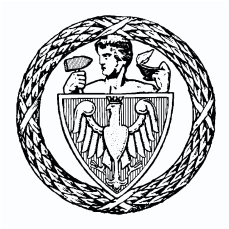
Mobile Robots

Simultaneous Localization and Mapping (SLAM)



Lecture 8

Simultaneous Localization and Mapping (SLAM) Problem

SLAM

Simultaneous Localization and Mapping (SLAM) addresses the problem of acquiring a map (a model) of a mobile robot environment while simultaneously localizing the robot relative to this map.

- A solution to the SLAM problem is regarded as one of the most important problems in the pursuit of building truly autonomous mobile robots.
- Typically, a mobile robot is placed at an unknown location in an unknown environment and the task is to incrementally build a consistent map of this environment while simultaneously determining robot's location within this map.
- SLAM is a chicken-or-egg problem
  - ✓ a map is needed for localizing a robot
  - ✓ a good pose estimate is needed to build a map
- Both the path of the vehicle and the map of the environment are estimated online without the need for any a priori knowledge of location.

SLAM Problem

- There is no single best solution to the SLAM problem. The chosen method can depend on a number of factors, such as: the desired map resolution, the update time, the nature of the features in the map.
- At present, there are exist robust methods for mapping environments that are static, structured, and of limited size.
- Mapping unstructured, dynamic, or large-scale environments remains largely an open research problem.
- Probabilistic methods outperform most other techniques.
- Perception and motion play an important role in SLAM.
  The following factors have to be taken into account:
  - ✓ sensor noise,
  - ✓ sensor aliasing,
  - ✓ effector noise.

SLAM Problem

Sensor noise is mainly influenced by:

- the environment (e.g. surface, illumination),
- the measurement principle itself.

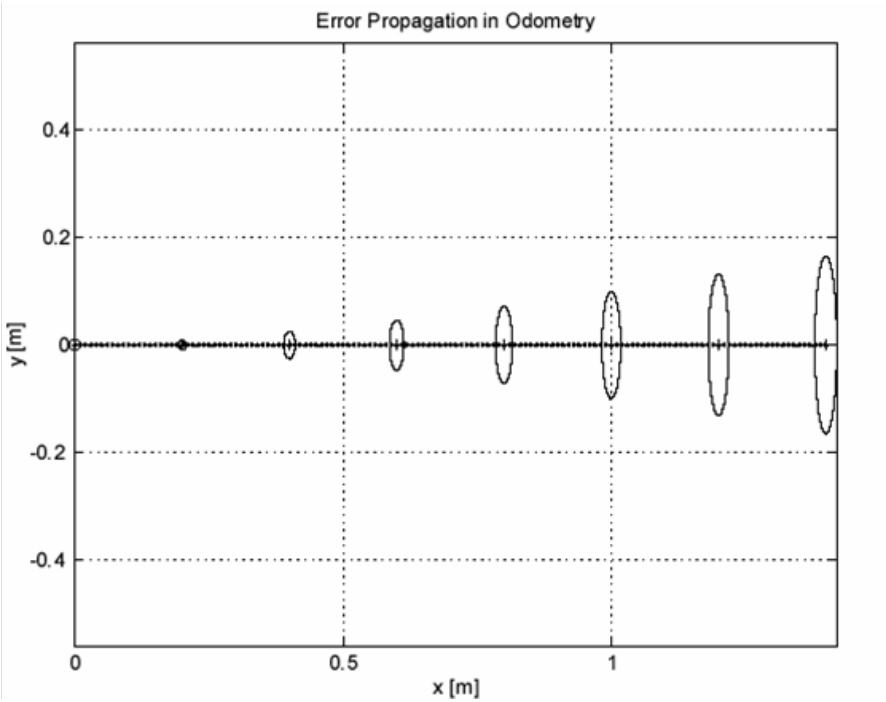
Sensor aliasing:

- In mobile robots, non-uniqueness of sensors readings is a typical situation, i.e. the mapping between observations and objects is unknown.
- Taking wrong data associations can have catastrophic consequences.
- Pose error correlates data associations.
- Even with multiple sensors, there is a many-to-one mapping from environmental states to robot perceptual inputs.
- Therefore the amount of information perceived by the sensors is generally insufficient to identify the robot's pose from a single reading.

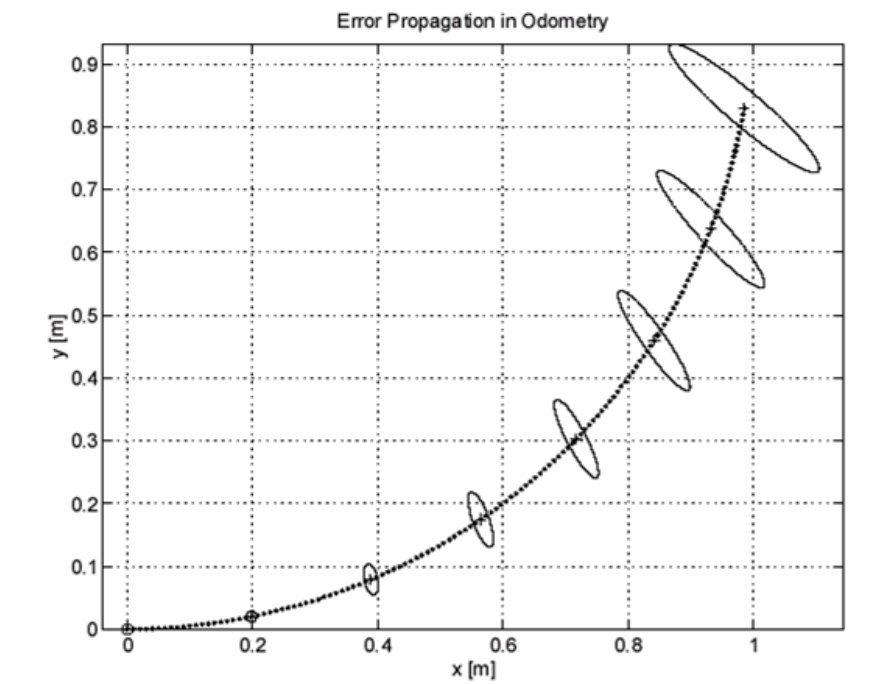
Effector Noise: Odometry, Deduced (or Dead) Reckoning:

- Pose update is based on proprioceptive sensors (odometry: wheel sensors only; dead reckoning: also heading sensors).
- The movement of the robot, sensed with wheel encoders and/or heading sensors is integrated to the pose.
  - \+ Pros: Straightforward, easy,
  - – Cons: Errors are integrated and can be unbounded.

Pose Uncertainty in Odometry



Pose uncertainty for straight line movement



Pose uncertainty for movement on a circle

Odometry errors:

1. deterministic (systematic) errors – can be eliminated by proper calibration of the system,
2. non-deterministic (random) errors – have to be described by probabilistic models and will always lead to uncertain pose estimate.

Pose Uncertainty in Odometry

Major Error Sources:

- Limited resolution during integration (time increments, measurement resolution)
- Misalignment of the wheels (deterministic)
- Unequal wheel diameter (deterministic)
- Variation in the contact point of the wheel
- Unequal floor contact (slipping, not planar . . .)

Classification of Integration Errors:

- *Range error* : integrated path length (distance) of the robots movement (sum of the wheel movements).
- *Turn error* : similar to range error, but for turns (difference of the wheel motions).
- *Drift error* : difference in the error of the wheels leads to an error in the robot's angular orientation.

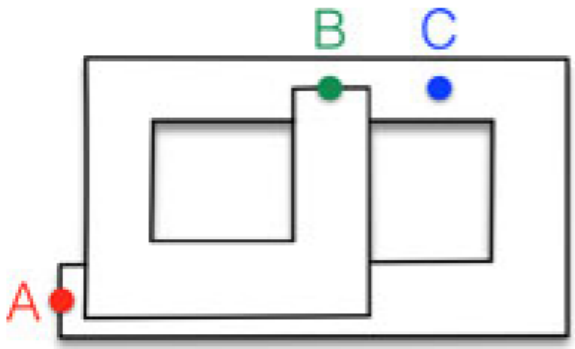
✓ Over long periods of time, turn and drift errors far outweigh range errors.

SLAM vs odometry

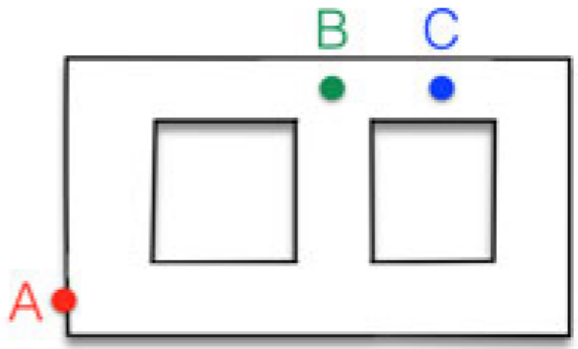
SLAM Problem

Simultaneous state estimation (e.g., pose) of a robot equipped with sensors and building a globally consistent model (map) of the environment these sensors measure.

- A robot performing odometry and neglecting loop closures interprets the world as a "long corridor".
- Points that are close in reality (B and C) may be arbitrarily far in the odometric map.
- By leveraging loop closures, SLAM estimates the actual topology of the environment, and "discovers" shortcuts in the map.

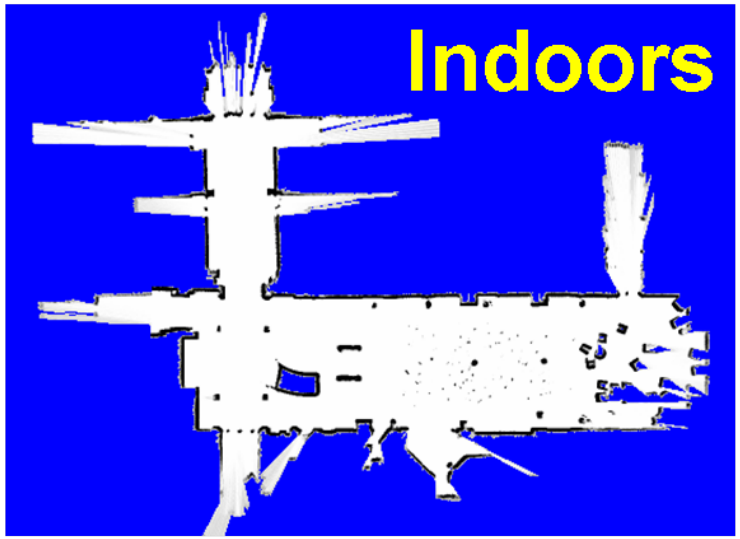


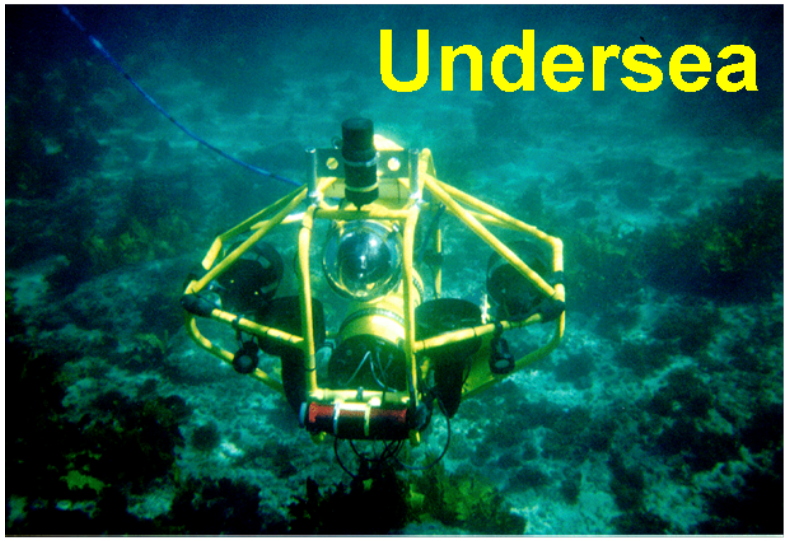
Map built from odometry

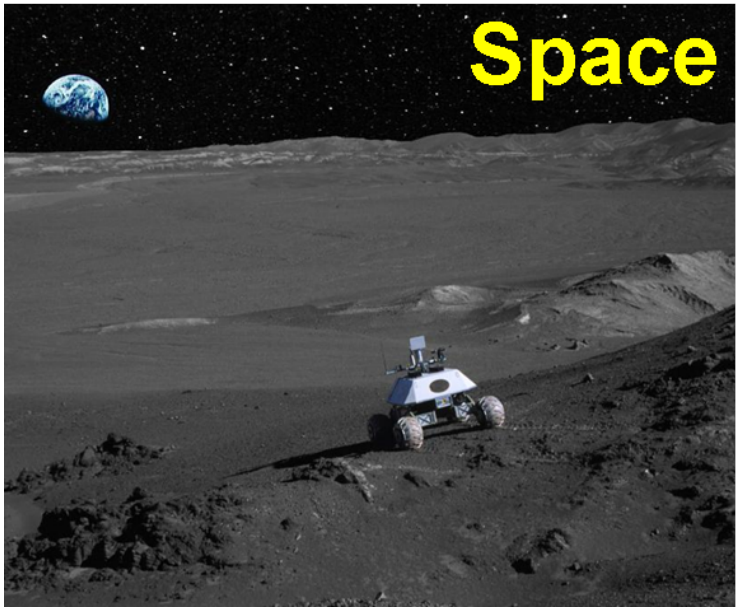


Map build from SLAM

SLAM Applications







SLAM Problem Definition

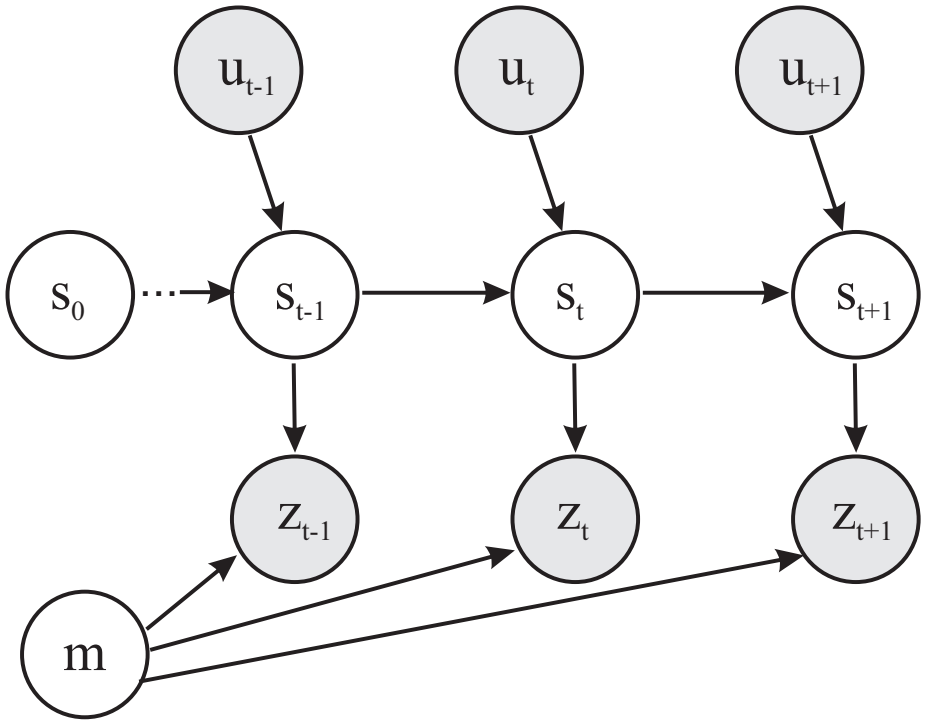
At a time instant t, the following quantities are defined:

- s_t – a state vector describing the location and orientation of the vehicle at time t (where $s_t = [x, y, \theta]^\mathsf{T}$).
- u_t – a control vector applied at $t-1$ to drive the vehicle from s_{t-1} to s_t. Odometry provides relative information between two consecutive locations, then u_t can denote the odometry that characterized the motion between time $t-1$ and t; such data might be obtained from the robot's wheel encoders or from the controls given to those motors.
- m – a true map of the environment, which may be comprised of landmarks, objects, surfaces, etc., and m describes their locations. The environment map m is typically assumed to be time invariant (i.e., static).
- m_i – a vector describing the location of the i-th landmark whose true location is assumed time invariant.
- z_t – observation of a landmark taken at time t (usually we assume that the robot takes exactly one measurement at each point in time).

SLAM Problem Definition

- $S_t = \{s_0, s_1, \ldots, s_t\} = \{S_{t-1}, s_t\}$: a history of vehicle locations, or path. Usually the initial location s_0 is known.
- $U_t = \{u_1, u_2, \ldots, u_t\} = \{U_{t-1}, u_t\}$: a history of control inputs.
- $Z_t = \{z_1, z_2, \ldots, z_t\} = \{Z_{t-1}, z_t\}$: a set of all landmark observations.
- For noise-free motion, U_t would be sufficient to recover the past S_t from the initial location s_0.
- However, odometry measurements are noisy, and path-integration techniques inevitably diverge from the truth.

SLAM Problem Definition



Graphical model of the SLAM problem (Dynamic Bayes Network): arcs indicate causal relationships, and shaded nodes are directly observable to the robot.

Two Forms of SLAM Problem

There are two main forms of the SLAM problem, which are both of equal practical importance:

1. The **full SLAM problem** – it involves estimating the posterior over the entire robot path together with the map:

$$p(S_t, m|Z_t, U_t)$$

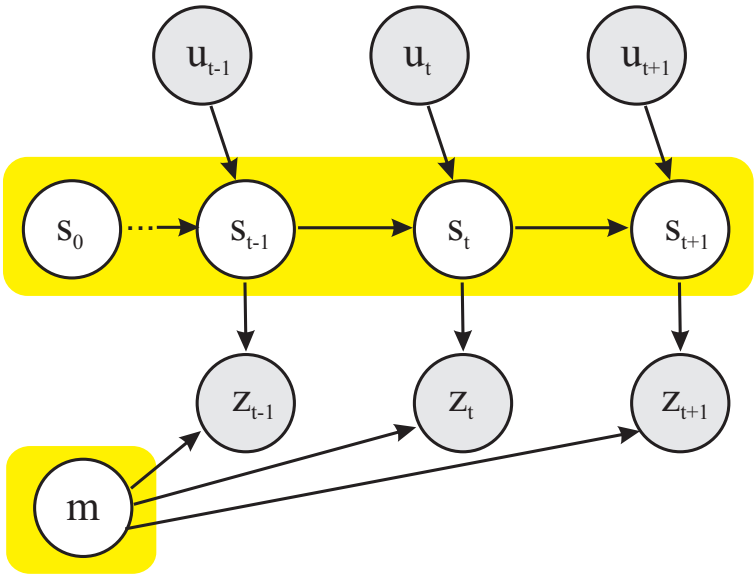
   The full SLAM problem is the problem of calculating the joint posterior probability over S_t and m from the available data.

2. The **online SLAM problem** – estimates most recent pose and map, instead of the entire path:

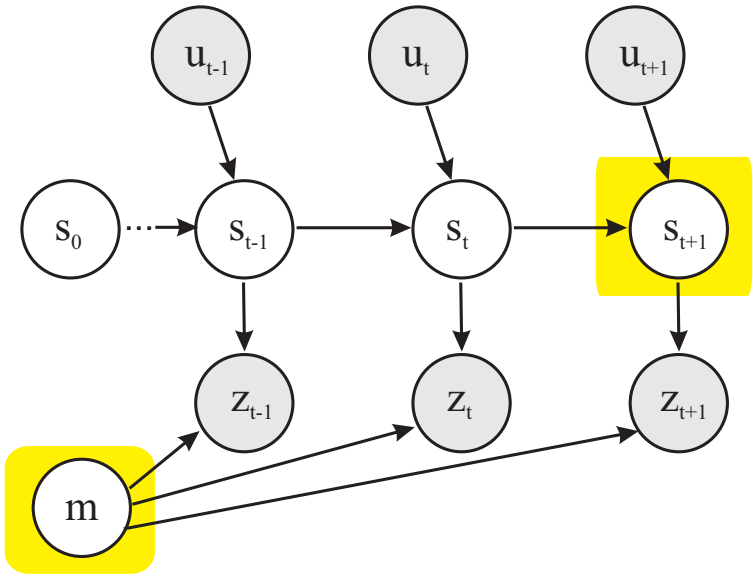
$$p(s_t, m|Z_t, U_t)$$

   Algorithms that solve the online SLAM problem are usually incremental and can process one data item at a time.

Two Forms of SLAM Problem



Full SLAM



On-line SLAM

SLAM Problem: Models

To solve SLAM problem, we need two more models:

1. **State transition model** (or *motion model*) for the vehicle which can be described in terms of a probability distribution on state transitions

$$p(s_t|s_{t-1}, u_t)$$

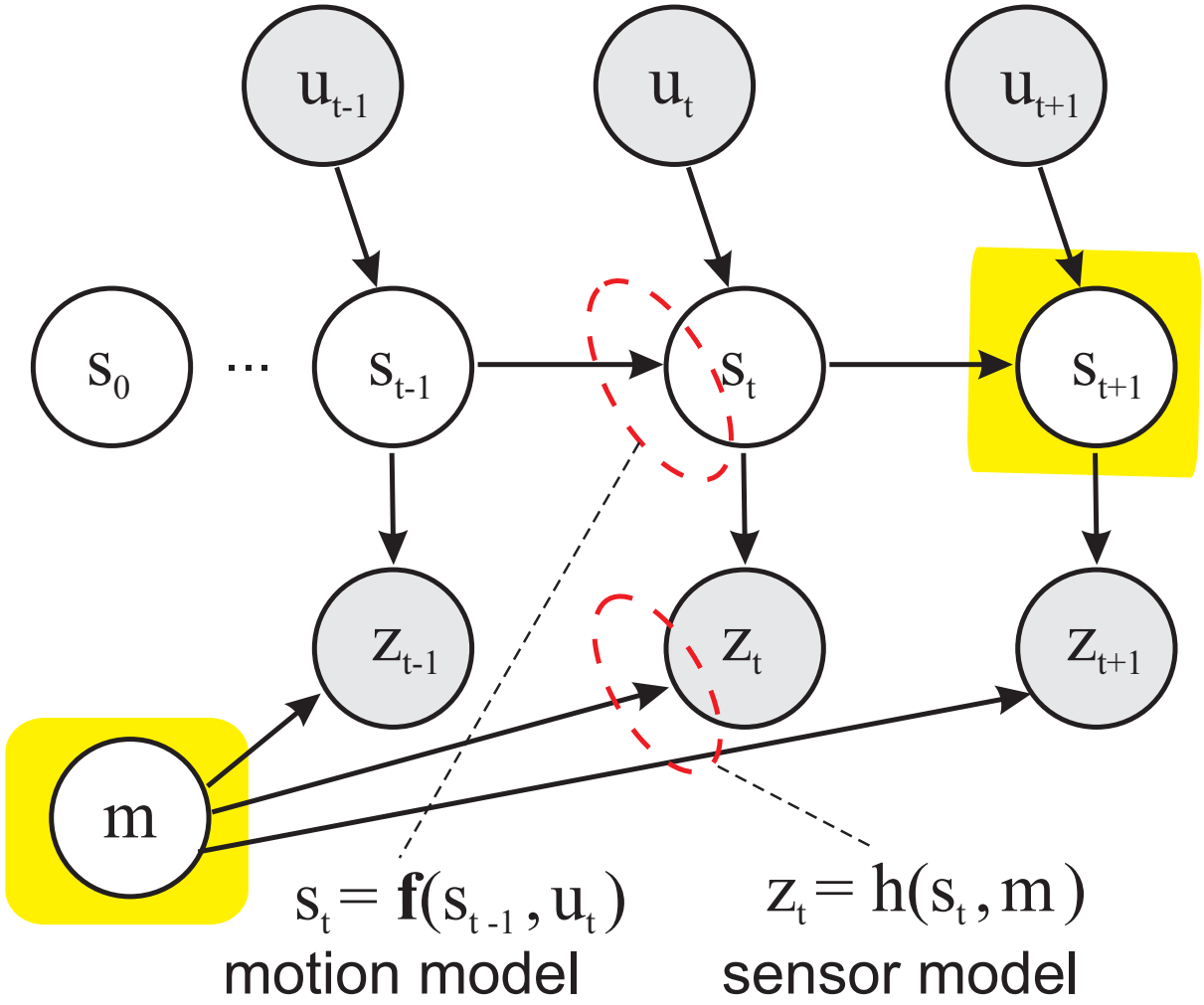
   the state transition is assumed to be a Markov process in which the next state s_t depends only on the immediately preceding state s_{t-1} and the applied control u_t and is independent of both the observations and the map.

2. **Observation model** (or *sensor model*) describes the probability of making an observation z_t when the vehicle location s_t and the map m are known

$$p(z_t|s_t, m)$$

   Of course, in the SLAM problem we do not know the robot location, and neither do we know the environment.

Motion and observation models in the SLAM



Ellipses indicate uncertainty of pose and measurement (observation)

SLAM Problem Formulation

We use Bayes rule to transform these mathematical relationships into a form where we can recover probability distributions over those latent variables from the measured data (recursive formulation):

$$bel(s_t, m) = \eta\, p(z_t \mid s_t, m) \iint p(s_t, m \mid s_{t-1}, m, u_{t-1}) bel(s_{t-1}, m) ds_{t-1} dm$$

where $bel(\cdot)$ – "belief" distribution, η – normalization factor.

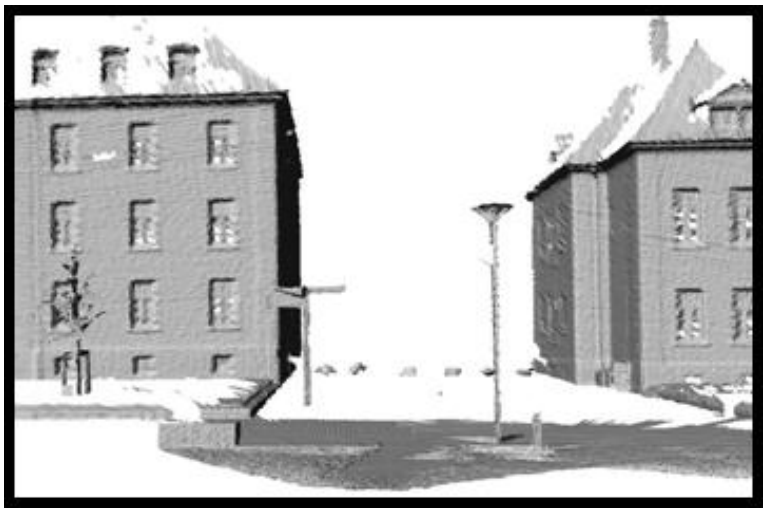
Usually, a static environment is assumed:

$$\underbrace{bel(s_t, m)}_{\text{Estimate}} = \eta\ \underbrace{p(z_t \mid s_t, m)}_{\text{Sensor Model}} \int \underbrace{p(s_t \mid s_{t-1}, u_{t-1})}_{\text{Motion Model}} bel(s_{t-1}, m) ds_{t-1}$$

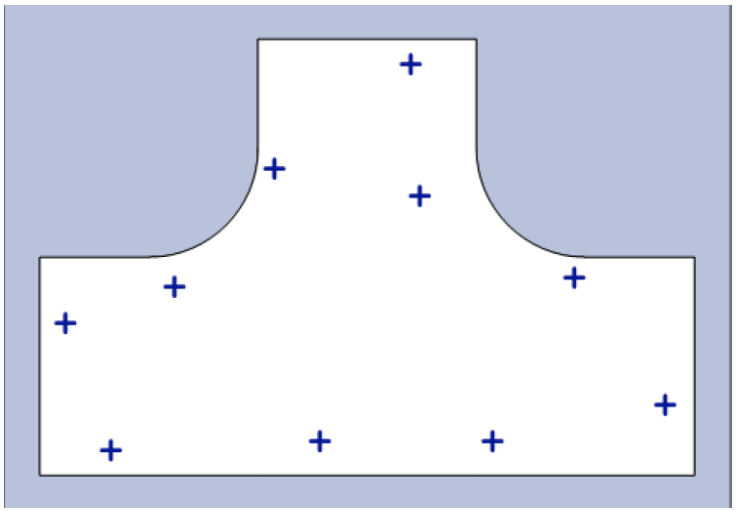
Taxonomy of the SLAM Problem I

SLAM problems are distinguished along a number of different dimensions:

1. Volumetric vs. Feature-Based
   - In volumetric SLAM, the map is sampled at a resolution high enough to allow for photorealistic reconstruction of the environment. The map m is usually high dimensional.
   - Feature-based SLAM extracts sparse features from the sensor measurements. The map is only comprised of features (point-landmark is an instance of feature-based SLAM).



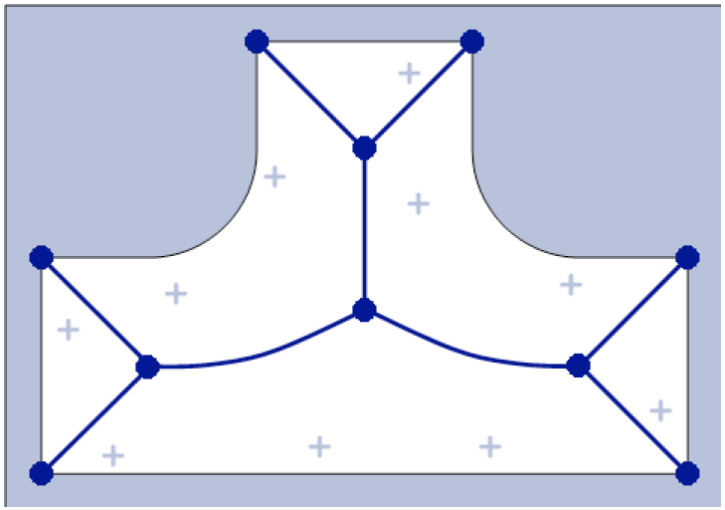
Volumetric Map



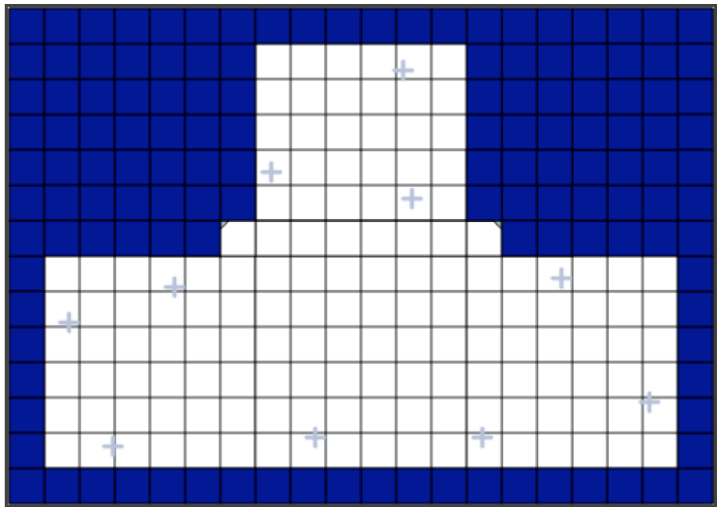
Feature-based Map

Taxonomy of the SLAM Problem II

2. Topological vs. Metric
   - A topological map can be defined over a set of distinct places and a set of qualitative relations between these places (e.g., place A is adjacent to place B).
   - Metric SLAM methods provide metric information between the relation of such places.



Topological Map



Metric Map

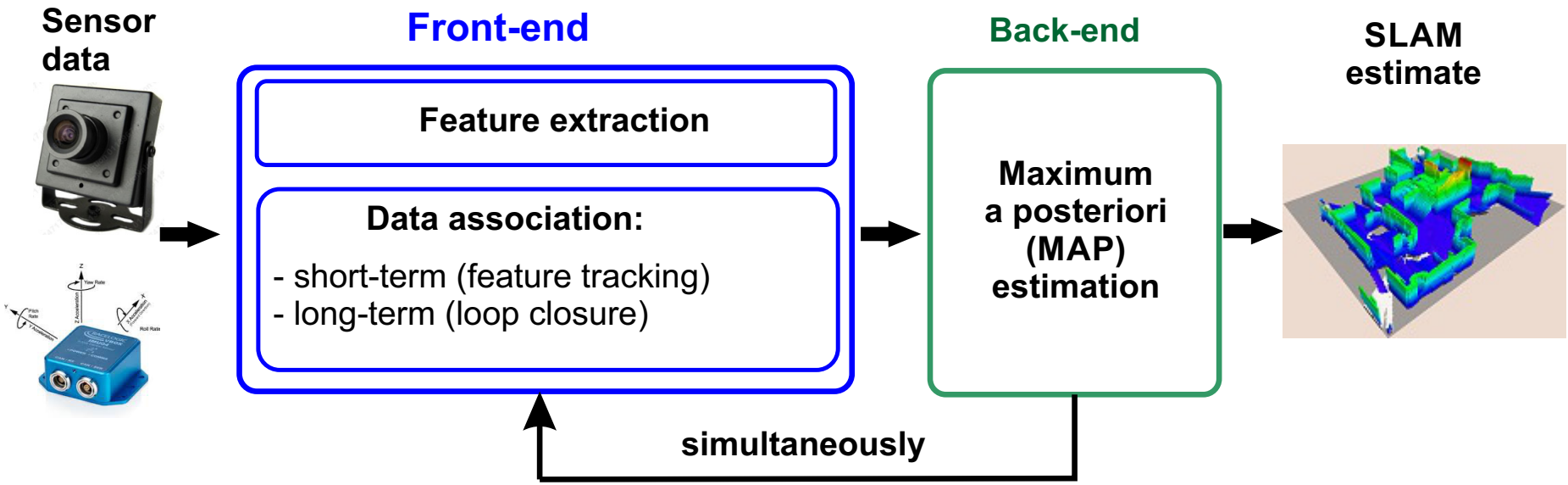
Taxonomy of the SLAM Problem III

3. Known vs. Unknown Correspondence
   - The correspondence problem is the problem of relating the identity of sensed things to other sensed things.
   - The problem of estimating the correspondence is known as the data association problem, and is one of the most difficult problems in SLAM.
4. Static vs. Dynamic
   - Static SLAM algorithms assume that the environment does not change over time.
   - Dynamic methods allow for changes in the environment.
   - The vast majority of SLAM algorithms assume static environments; dynamic effects are often treated just as measurement outliers.

Taxonomy of the SLAM Problem IV

5. Small vs. Large Uncertainty
   - The most simple SLAM algorithms allow only for small errors in the location estimate.
   - They are good for situations in which a robot goes down a path that does not intersect itself, and then returns along the same path.
   - In many environments it is possible to reach the same location from multiple directions.
   - This problem is known as the *loop-closing problem*, when closing a loop, the uncertainty may be large.
   - The uncertainty can be reduced if the robot can sense information about its position in some absolute coordinate frame (e.g. through the use of GPS).
6. Active vs. Passive
   - In passive SLAM algorithms, some other entity controls the robot, and the SLAM algorithm is purely observing.
   - The vast majority of algorithms are of this type.
   - In active SLAM, the robot actively explores its environment in the pursuit of an accurate map.
   - Active SLAM methods tend to yield more accurate maps in less time, but they constrain the robot motion.

SLAM system architecture



The back end can provide feedback to the front end for loop closure detection and verification.

MAP (*Maximum a Posteriori*) estimation:

$$x^* = \arg\max_x \; p(x \mid z) = \arg\max_x \; p(z \mid x)\, p(x), \quad \text{where } x = \{s, m\}$$

$p(z \mid x)$ is the likelihood of the measurements z given the assignment x, and $p(x)$ is a prior probability over x.

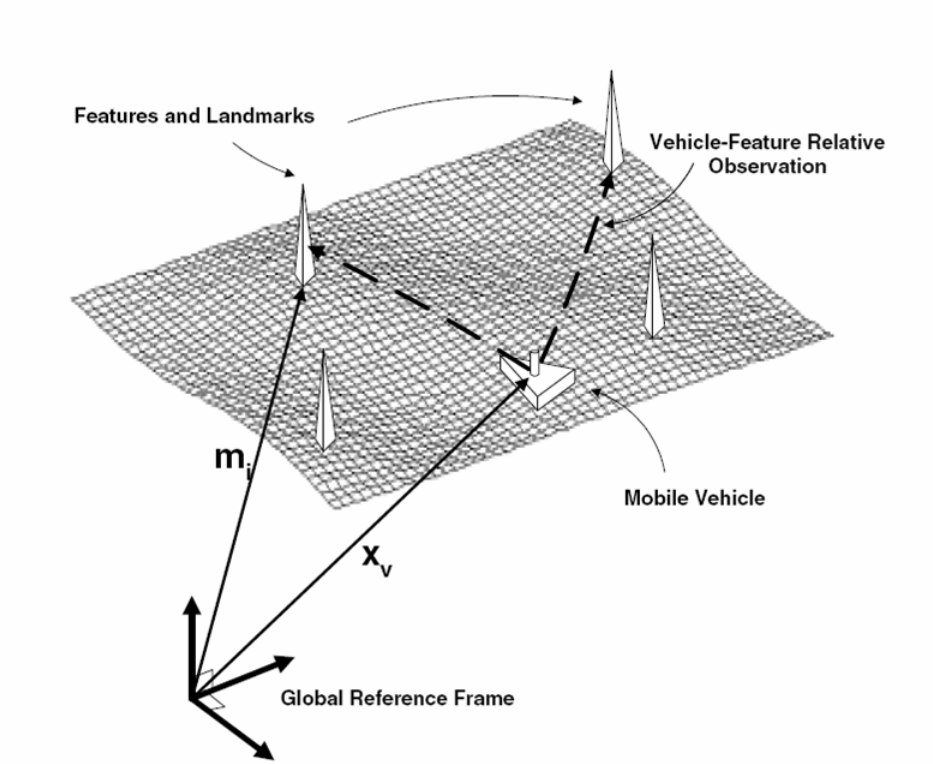
The Main Approaches to SLAM Problem

Three basic SLAM paradigms:

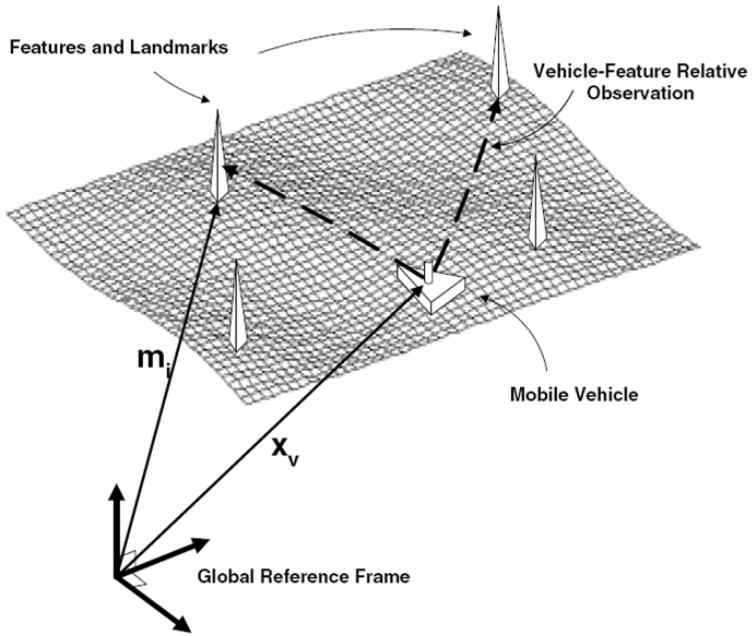
1. **Extended Kalman Filter (EKF)**:
   - Historically the earliest approach.
   - Assumes a metrical, feature-based environmental representation.
   - The position of the robot and the locations of features form a network of uncertain spatial relationships.
2. **Graph-Based Optimization** techniques:
   - Solve the SLAM problem through nonlinear sparse optimization.
   - Landmarks and robot locations are nodes in a graph.
   - Arcs in this graph are soft constraints. Relaxing these constraints yields the robot's best estimate for the map and the full path.
   - The main paradigm for solving the full SLAM problem.
3. **Particle Filters**:
   - They are non-parametric statistical filtering techniques.
   - Particle filters represent a posterior through a set of particles.
   - It is a popular method for online SLAM.

Structure of the Landmark-based SLAM-Problem

- A vehicle with a known kinematic model moving through an environment containing a population of landmarks – *motion model*.
- The vehicle is equipped with a sensor that can take measurements of the relative location between any individual landmark and the vehicle itself – *observation model*.



Structure of the Landmark-based SLAM-Problem



- The environment is populated by point landmarks.
- When building 2D maps, point landmarks may correspond to door posts and corners of rooms, etc., which, when projected into a 2D map are characterized by point coordinates.
- The robot can sense three quantities:
  - the relative range to nearby landmarks,
  - their relative bearing,
  - the identity of these landmarks.

Structure of the Landmark-based SLAM-Problem

- The exact, noise-free measurement function $z_t = h(s_t, m)$ describes how sensor works.
- The probabilistic measurement model is derived from this measurement function by adding a noise term:

$$p(z_t|s_t, m) \sim \mathcal{N}(h(s_t, m), Q_t),$$

where $\mathcal{N}$ denotes the two-dimensional normal distribution, and the (2×2) matrix Q_t is the measurement noise covariance.

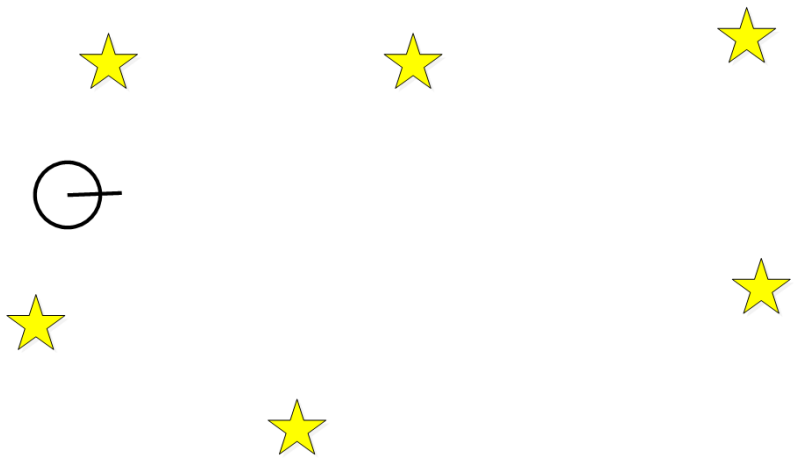
- The motion model is derived from a kinematic model of robot motion $s_t = g(s_{t-1}, u_t)$:

$$p(s_t|s_{t-1}, u_t) = \mathcal{N}(g(s_{t-1}, u_t), R_t),$$

where R_t is a covariance matrix of size (3×3), since the location is a 3D vector.

Why SLAM problem is hard to solve?

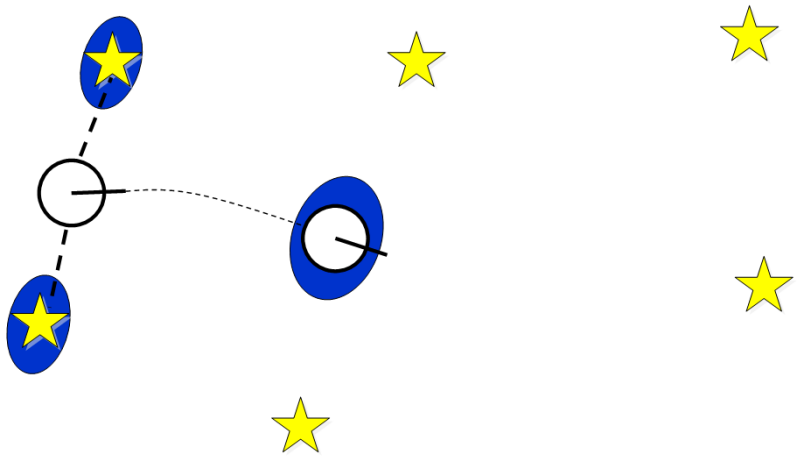
Robot path and map are both **unknown**:



Errors in map and pose estimates are correlated.

Why SLAM problem is hard to solve?

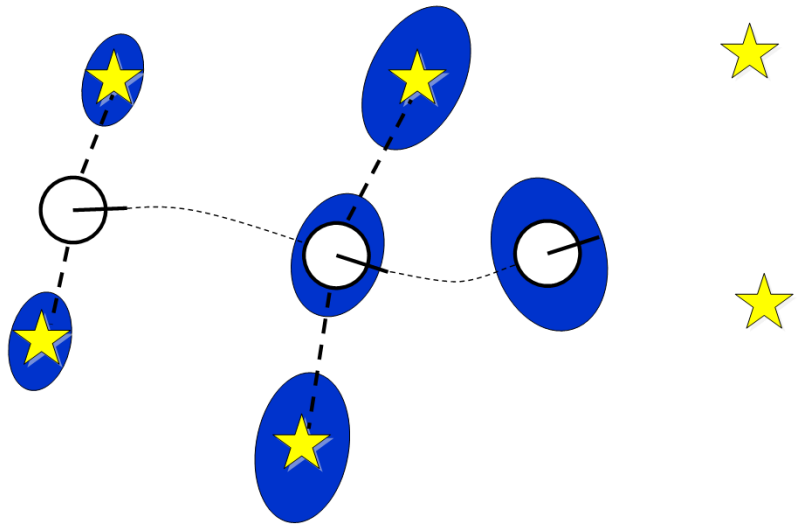
Robot path and map are both **unknown**:



Errors in map and pose estimates are correlated.

Why SLAM problem is hard to solve?

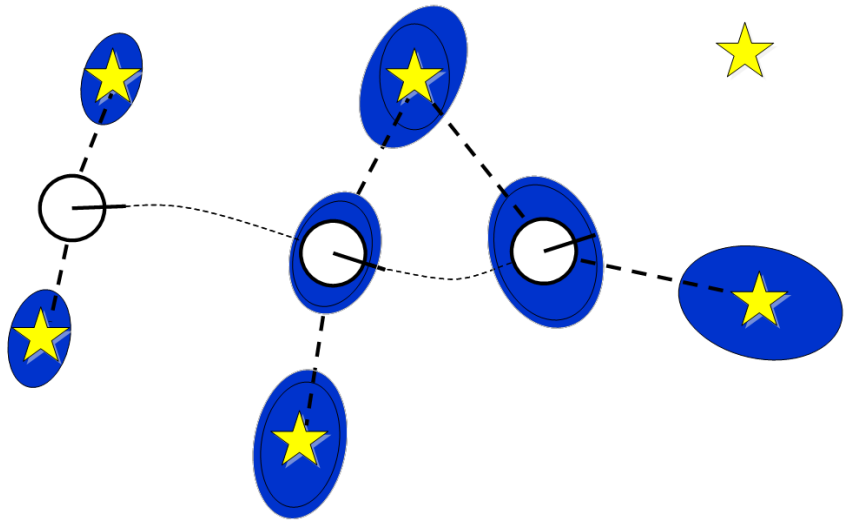
Robot path and map are both **unknown**:



Errors in map and pose estimates are correlated.

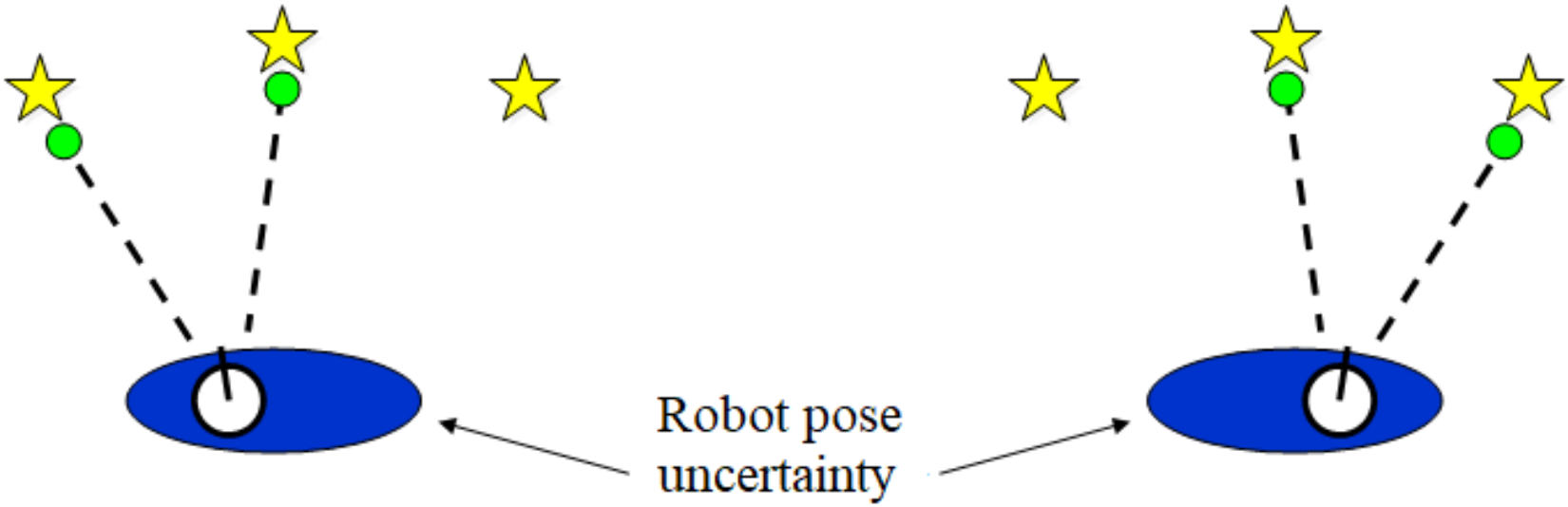
Why SLAM problem is hard to solve?

Robot path and map are both **unknown**:



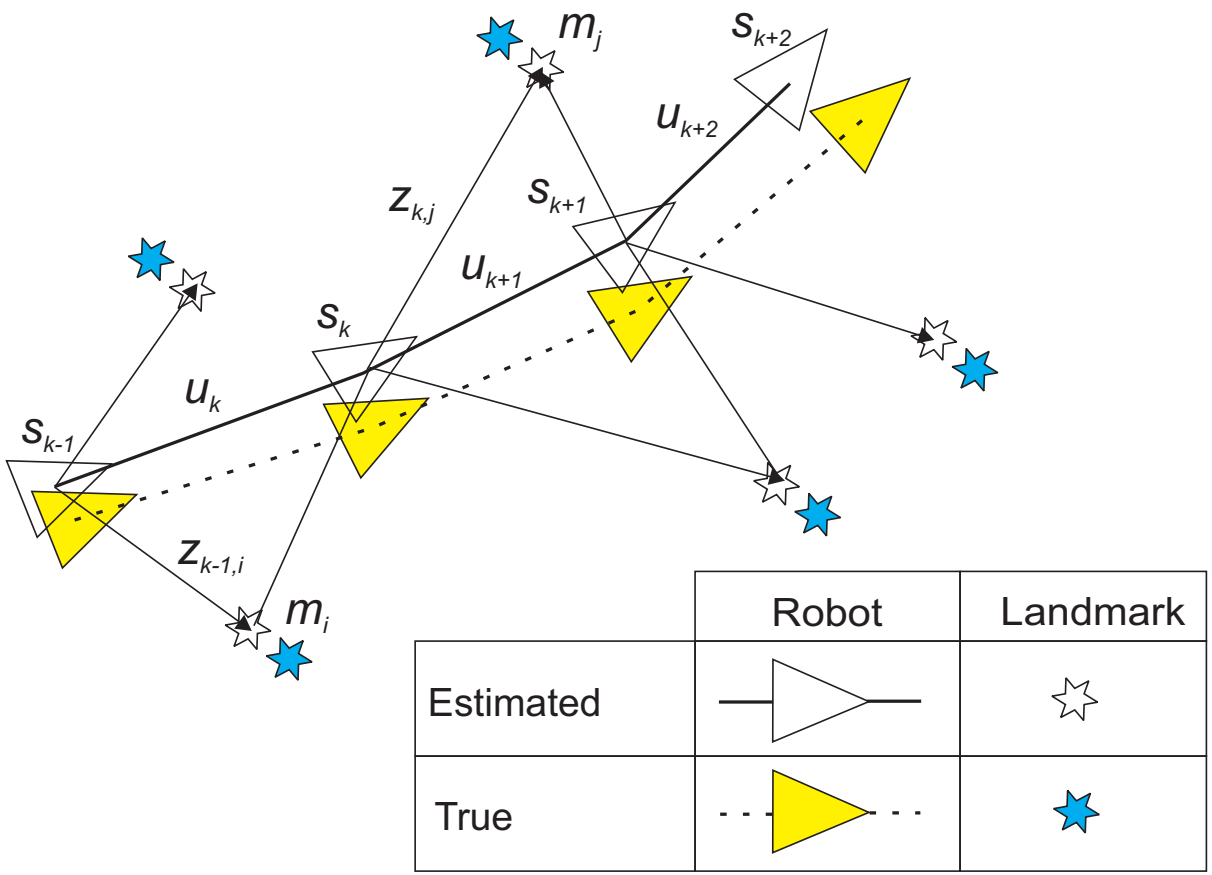
Errors in map and pose estimates are correlated.

Why SLAM problem is hard to solve?



- In a real world the mapping between observations and landmarks is unknown.
- Robot pose estimate becomes correlated with the feature locations.
- Picking wrong data associations can have catastrophic consequences (divergence).

The EKF SLAM I



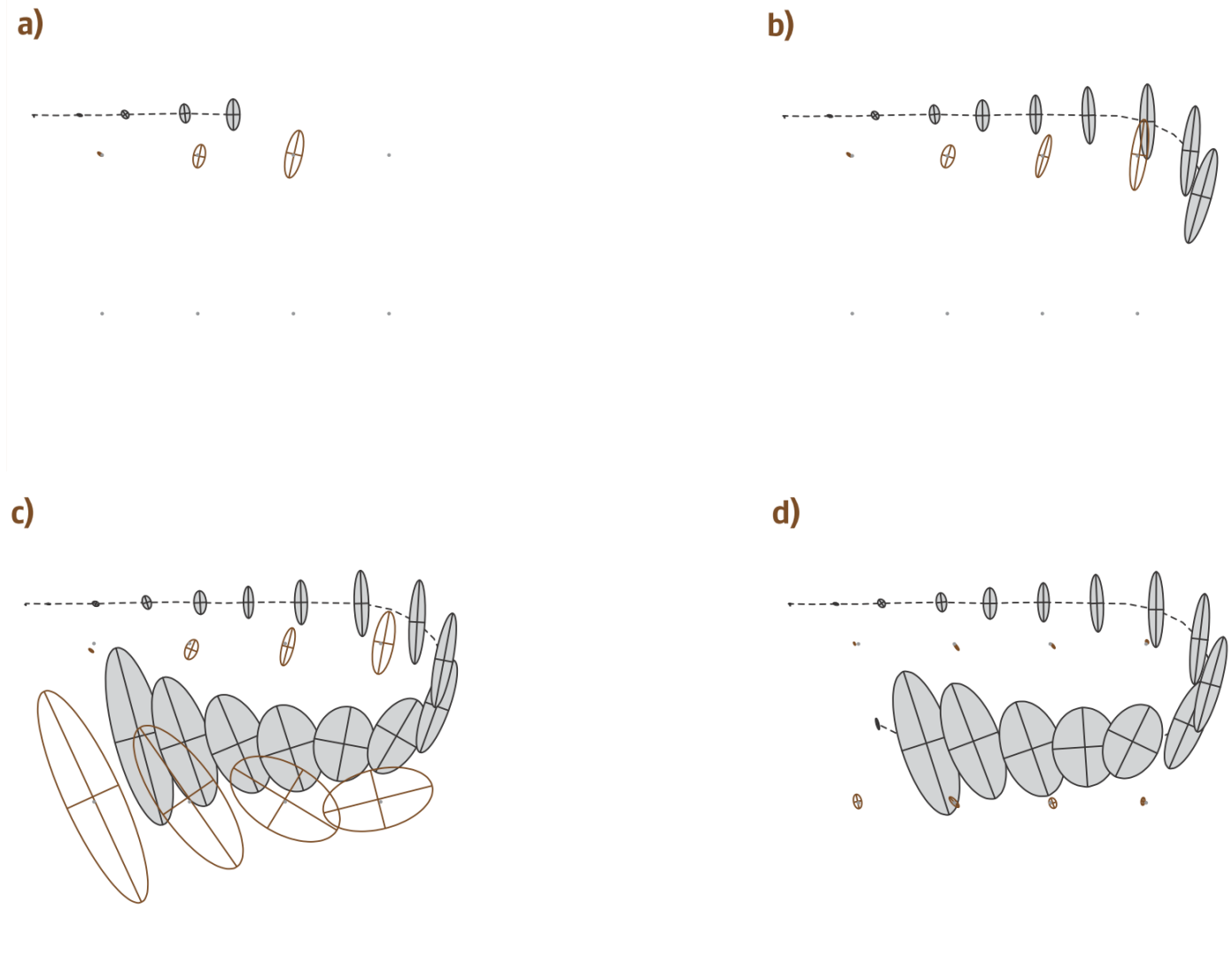
The EKF algorithm represents the robot estimate by a multivariate Gaussian:

$$p(s_t, m|Z_t, U_t) \sim \mathcal{N}(\mu_t, \Sigma_t)$$

The EKF SLAM II

- The $(3 + 2N)$ vector μ_t contains the robot's best estimate of its own location and the location of the features in the environment.
- The quadratic $(3 + 2N) \times (3 + 2N)$ matrix Σ_t is the covariance of the robot's assessment of its expected error in the guess μ_t.
- The off-diagonal elements capture the correlations in the estimates of different variables.
- There are nonzero correlations because the robot's location is uncertain, and as a result the locations of the landmarks in the maps are uncertain.
- The importance of maintaining those off-diagonal elements is one of the key properties of EKF SLAM.
- EKF SLAM linearizes the functions g and h using Taylor-series expansion.

EKF SLAM: The Example I



EKF applied to the on-line SLAM problem

EKF SLAM: The Example II

- The robot's path is *a dotted line*, and its estimates of its own position are *shaded ellipses*.
- Eight distinguishable landmarks of unknown location are shown as *small dots*, and their location estimates are shown as *white ellipses*.
- In (a-c) the robot's positional uncertainty is increasing, as is its uncertainty about the landmarks.
- In (d) the robot observes the first landmark again, and the uncertainty of all landmarks decreases, as does the uncertainty of its current pose.
- This phenomenon arises from a correlation that is expressed in the covariance matrix of the Gaussian posterior.
- Since most of the uncertainty in earlier landmark estimates is caused by the robot pose, and since this very uncertainty persists over time, the location estimates of those landmarks are correlated.
- Information that helps localize the robot is propagated through the map, and as a result improves the localization of other landmarks in the map.

EKF SLAM: Summary

- The basic formulation of the EKF-SLAM has a quadratic complexity in the number of landmarks: $O(N^2)$.
- Convergence and consistency can only be guaranteed in the linear case. EKF-SLAM employs linearized models of nonlinear motion and observation models.
- Can diverge if nonlinearities are large!
- The standard formulation of the EKF-SLAM solution is especially fragile to incorrect association of observations to landmarks.
- The loop-closure problem, when a robot returns to re-observe landmarks after a large traverse, is especially difficult.
- Have been applied successfully in many environments.
- It is easy to implement.

Particle Filters

- Particle filters are mathematical models that represent a posterior through a set of particles:

$$bel(s_k) \approx S_k = \left\{ (s_k^{[i]}, w_k^{[i]}) \ : \ i = 1, \ldots, M \right\},$$

  where
  - $s_k^{[i]}$ is a sample (state hypothesis), and
  - $w_k^{[i]}$ is an importance weight (all weights should sum up to 1).
- Each particle is best thought as a concrete guess as to what the true value of the state may be.
- It is a nonparametric representation that represents multi-modal distributions.
- The samples represent the posterior

$$P(x) = \sum_{i=1}^{M} w^{[i]} \cdot \delta_{s^{[i]}}(x), \text{ where } \quad \delta_{s^{[i]}} \quad \text{– Dirac delta function}$$

Particle Filter Algorithm

Algorithm: *particle_filter*(S_{k-1}, u_{k-1}, z_k)

1: $S_k = \emptyset, \quad \eta = 0$

2: **for** $i = 1$ **to** M $\leftarrow$ *Generate new samples*

3: Sample index $j(i)$ from the discrete distribution given by w_k

4: Sample $s_k^{[i]}$ z $P(s_k|s_{k-1}, u_{k-1})$ using $s_{k-1}^{j[i]}$ and u_{k-1}

5: $w_k^{[i]} = P(z_k|s_k^{[i]})$ $\leftarrow$ *Compute importance weight*

6: $\eta = \eta + w_k^{[i]}$ $\leftarrow$ *Update normalization factor*

7: $S_k = S_k \cup \{(s_k^{[i]}, w_k^{[i]})\}$ $\leftarrow$ *Insert sample*

8: **for** $i = 1$ **to** M

9: $w_k^{[i]} = w_k^{[i]}/\eta$ $\leftarrow$ *Normalize weights*

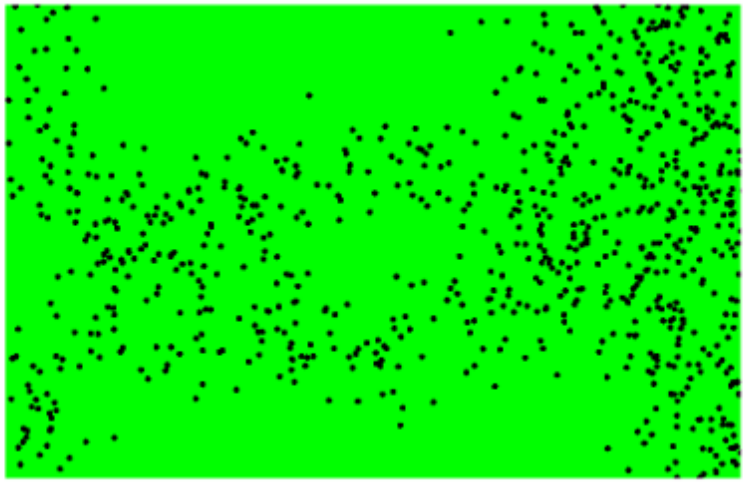
Importance Sampling with Resampling (SIR)

Importance Sampling with Resampling (SIR)

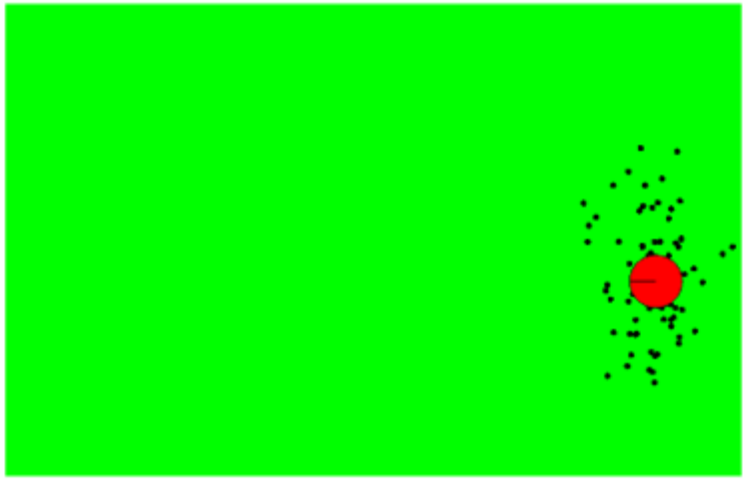
1. Draw the new generation of particles: In the update step a new particle distribution given motion model and controls applied is generated.
2. Assign an importance weight to each particle:
   - Compare particle's prediction of measurements with actual measurements
   - Particles whose predictions match the measurements are given a high weight
3. Resampling:
   - The probability of drawing a particle is its normalized importance weight.
   - Each particle survives with a probability that is proportional to the likelihood of the observation given that particle and its map.

Problem: The number of particles needed to represent a posterior grows exponentially with the dimension of the state space!

Importance Sampling with Resampling (SIR)



Weighted samples



After resampling (red circle = robot real pose)

Particle Filters: Factorization

- The high dimensional state-space of the SLAM problem makes direct application of particle filters computationally infeasible.
- However, it is possible to reduce the samples pace by applying so-called Rao-Blackwellization (R-B), whereby a joint state is partitioned according to the product rule $p(s_1, s_2) = p(s_2 \mid s_1)p(s_1)$, and $p(s_2 \mid s_1)$ can be represented analytically, only $p(s_1)$ need be sampled $s_1^{(i)} \sim p(s_1)$.
- The joint distribution, therefore, is represented by the set $\{s_1^{(i)}, p(s_2 \mid s_1^{(i)})\}_i^n$, and the marginal

$$p(s_2) \approx \frac{1}{n} \sum_i^n p(s_2 \mid s_1^{(i)})$$

  can be computed with greater accuracy than is possible by sampling over the joint space.
- The joint SLAM state may be factored into a vehicle component and a conditional map component:

$$p(s_{0:t}, m | z_{1:t}, u_{1:t}, s_0) = \underbrace{p(m | z_{1:t}, s_{0:t})}_{\text{map}} \cdot \underbrace{p(s_{0:t} | z_{1:t}, u_{1:t}, s_0)}_{\text{motion model}}$$

FastSLAM Algorithm I

- The probability distribution is on the path $s_{0:t}$ rather than the single pose s_t because, when conditioned on the path, the map landmarks become independent.
- A key property of FastSLAM: the map is represented as a set of independent Gaussians, with linear complexity, rather than a joint map covariance with quadratic complexity (like in the EKF-SLAM).
- The path is represented by weighted samples and the map is computed analytically.
- The joint distribution, at time t, is represented by the set $\{s_{0:t}^{(i)}, w_t^{(i)} p(m|z_{1:t}, s_{0:t}^{(i)})\}_i^n$, where the map accompanying each particle is composed of independent Gaussian distributions:

$$p(m|z_{1:t}, s_{0:t}) = \prod_j^M p(m_j|z_{1:t}, s_{0:t})$$

FastSLAM Algorithm II

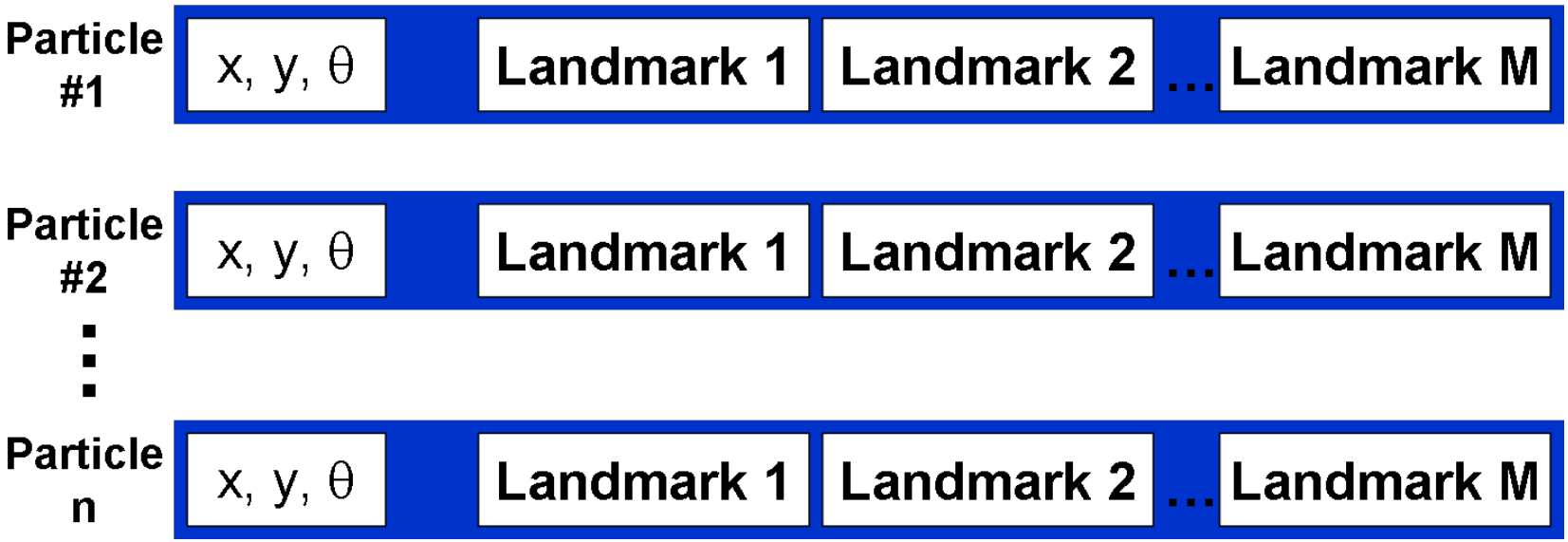
- Recursive estimation is performed by particle filtering for the pose states and the EKF for the map states.

$$\underbrace{p(s_{0:t}, m_{1:M}|z_{1:t}, u_{0:t-1})}_{\text{SLAM posterior}} = \underbrace{p(s_{0:t}|z_{1:t}, u_{0:t-1})}_{\text{Robot path posterior}} \cdot \underbrace{\prod_{j}^{M} p(m_j|z_{1:t}, s_{0:t})}_{\substack{\text{Conditionally independent} \\ \text{landmark positions}}}$$

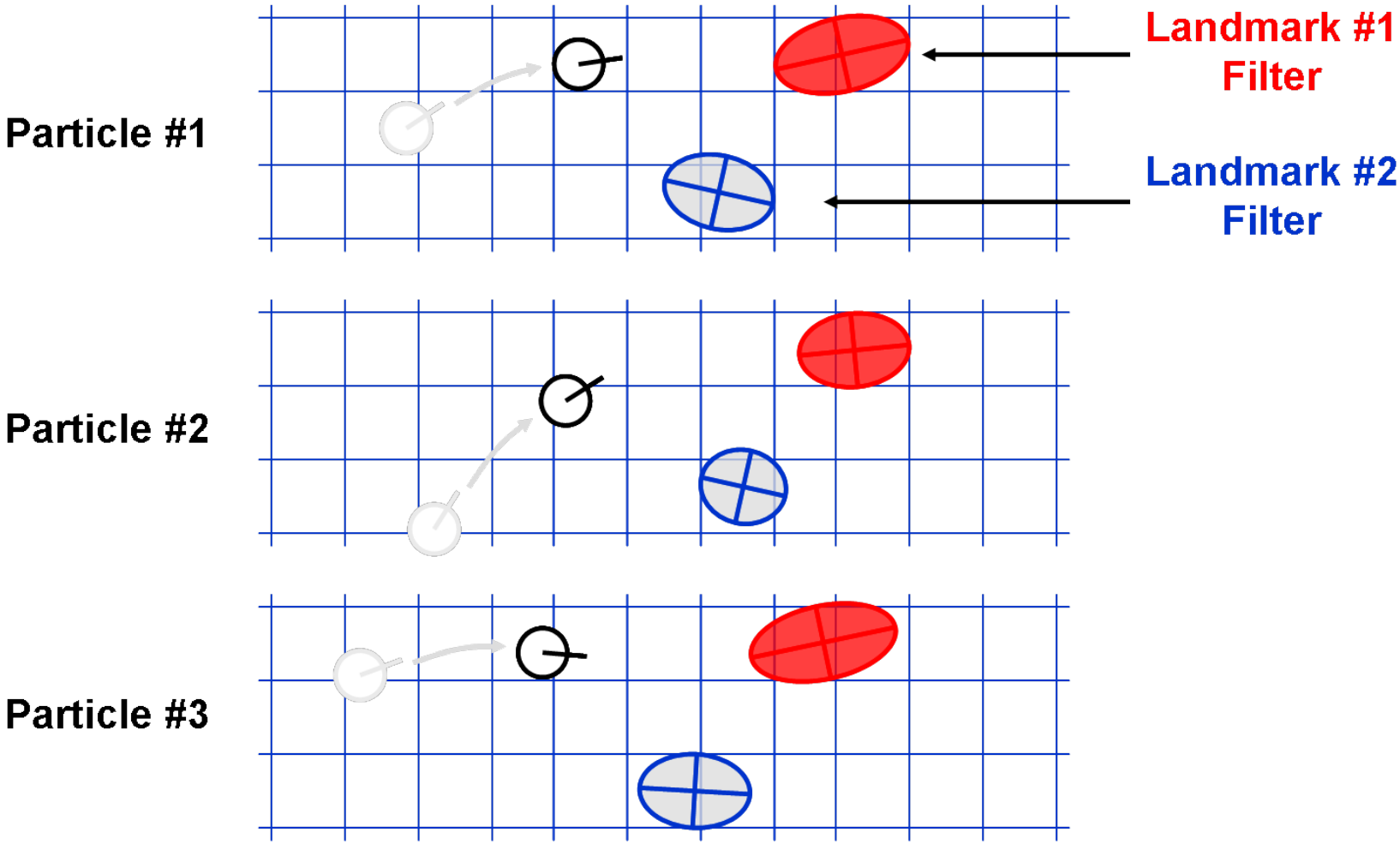
FastSLAM idea:

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

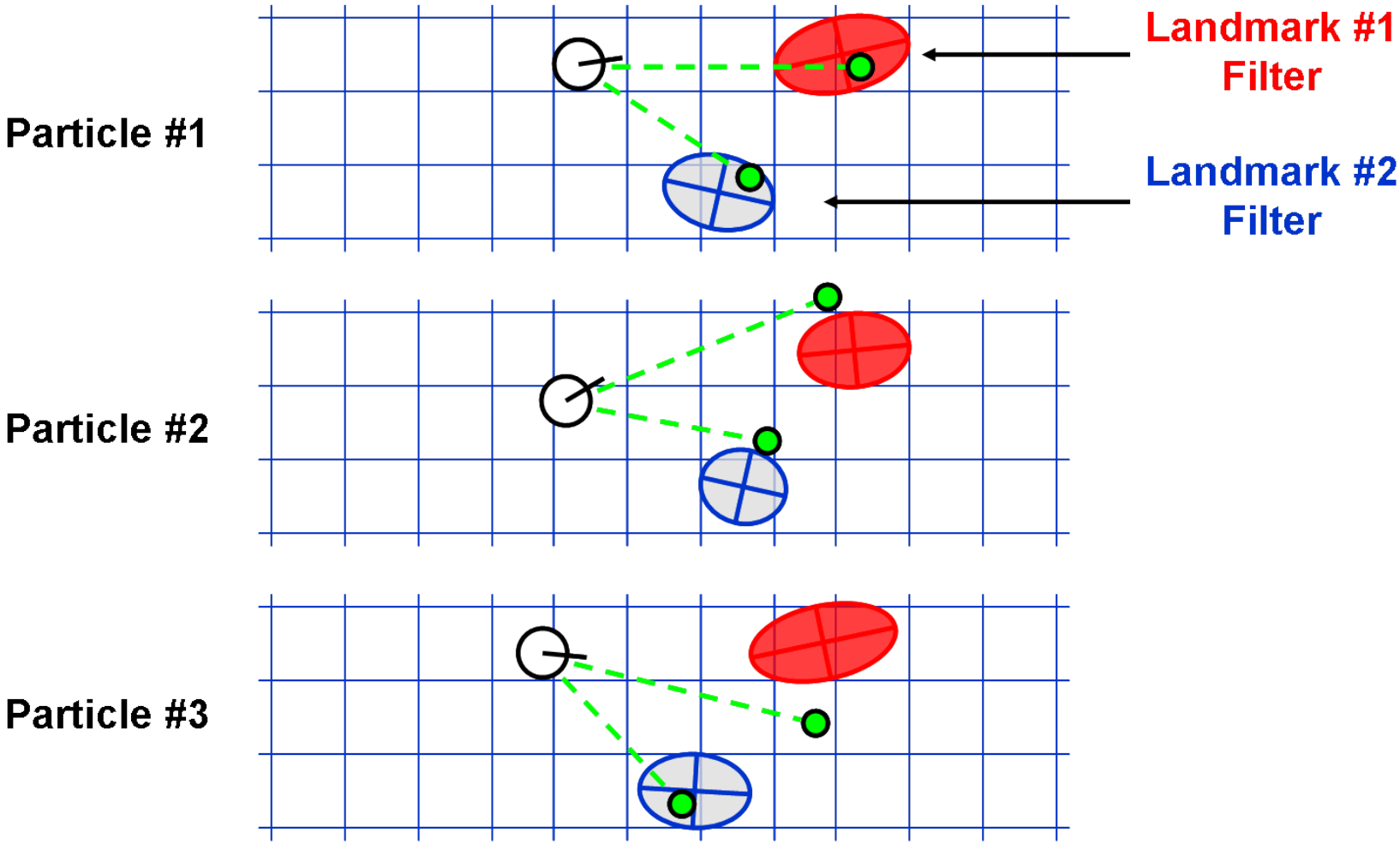
FastSLAM Algorithm III



FastSLAM: Action (Motion) Update

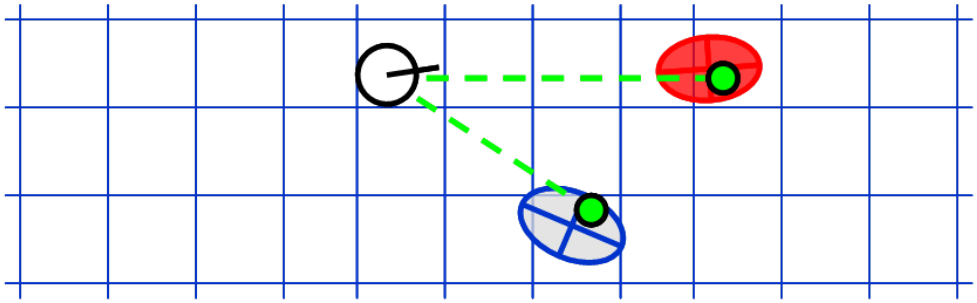


FastSLAM: Sensor Update



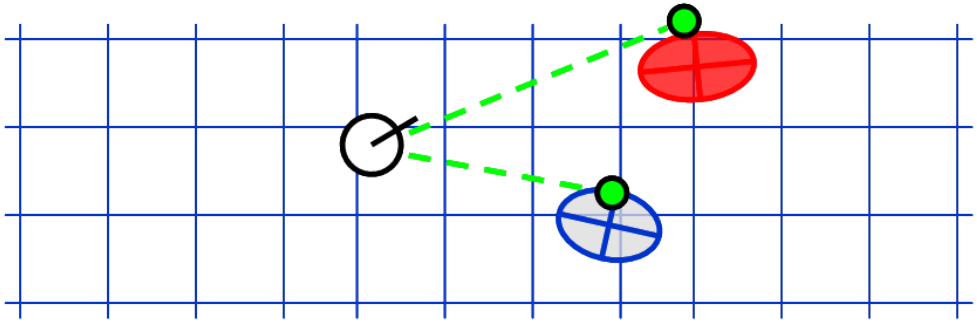
FastSLAM: Sensor Update

Particle #1



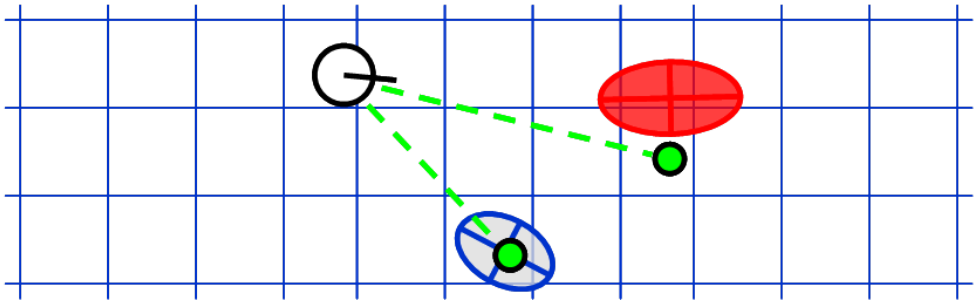
Weight = 0.8

Particle #2



Weight = 0.4

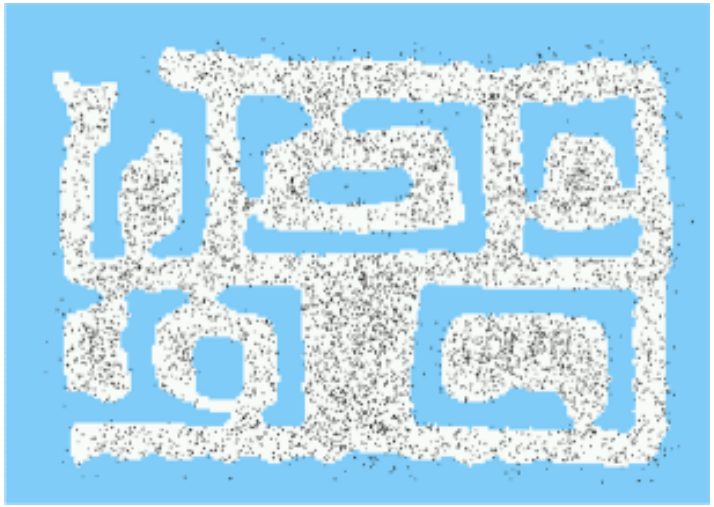
Particle #3



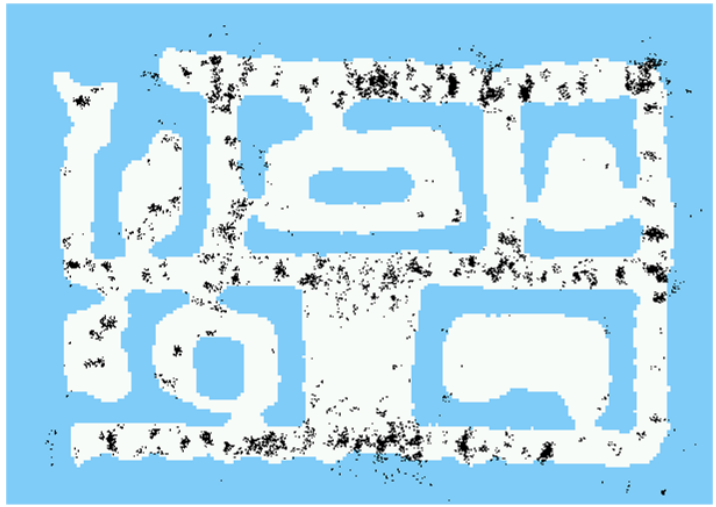
Weight = 0.1

FastSLAM in Action

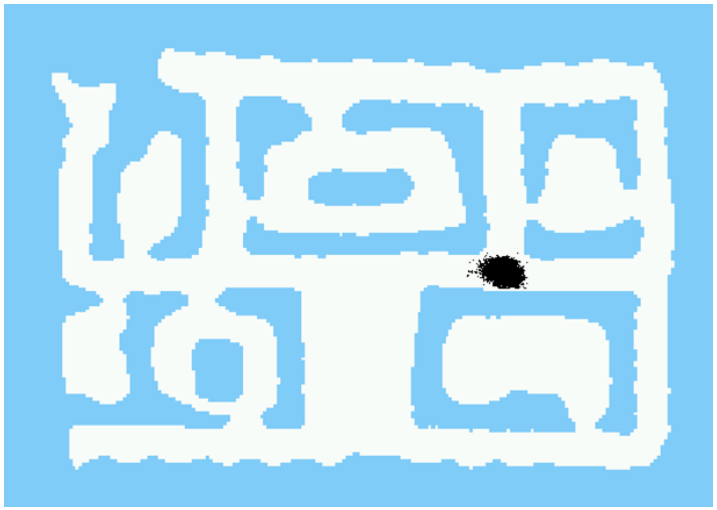
a)



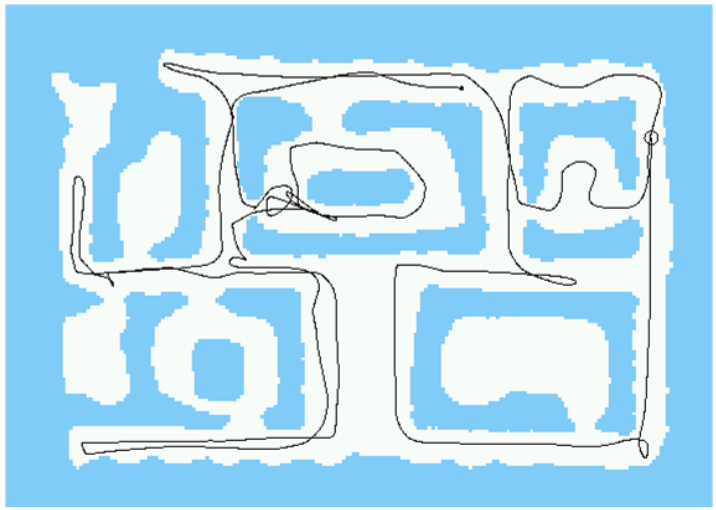
b)



c)

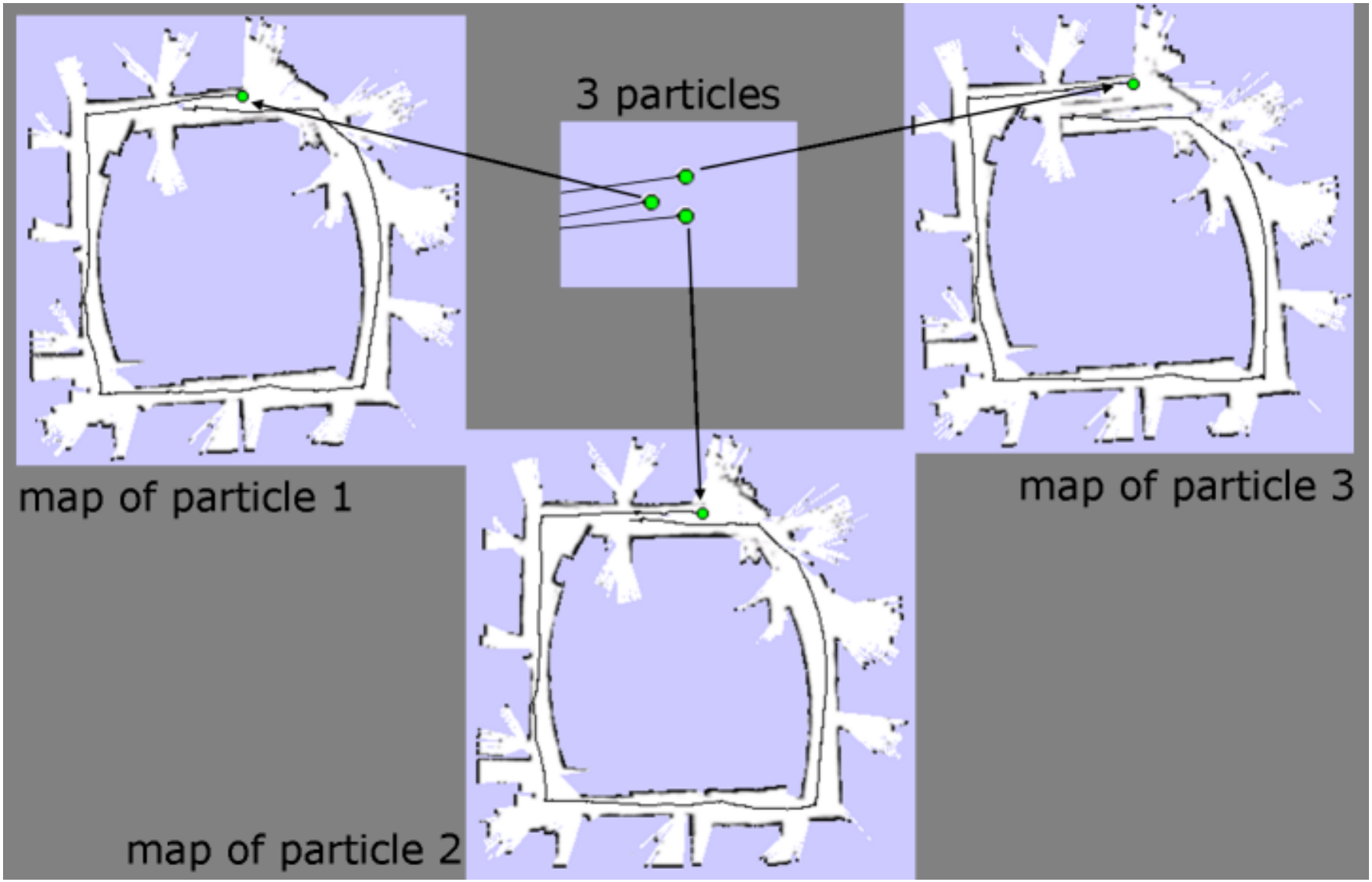


d)



FastSLAM: a) initial distribution, b) after 15 steps c) after 65 step, d) robot path

Particle Filter Example



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).
- Total complexity: O(n log(M))

FastSLAM: Summary

- It solves both the full and the on-line SLAM problems.
- It factors the SLAM posterior into low-dimensional estimation problems.
- FastSLAM can represent the posterior by a sampled robot pose, and many local, independent Gaussians for its landmarks.
- Each particle has a sample of an entire path, but the actual update equation only uses the most recent pose.
- FastSLAM can be implemented very efficiently.
- Its update requires linear-logarithmic time (logarithmic in the size of the map M, and linear in the number of particles n) where EKF needed quadratic time.