Introduction to

Simultaneous Localization and Mapping

Spring Semester Course 2019

CS284

Course Convener:

Prof Laurent Kneip

lkneip@ shanghaitech.edu.cn

Disclaimer

- Some of the lecturing material is naturally taken from publically available online material from other groups. Sources include:
  - Autonomous Systems Lab ETH Zurich
  - Research School of Engineering, ANU
  - Informatik Department, Uni Freiburg
  - MIT
  - …
- By using the present material you confirm that
  - You use it only internally and for the purpose of education
  - You are aware that the lecture material may originate from other sources, even if explicit reference is occasionally missing

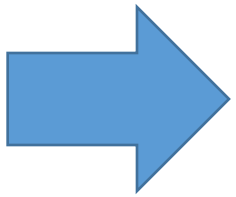
Where are we at?

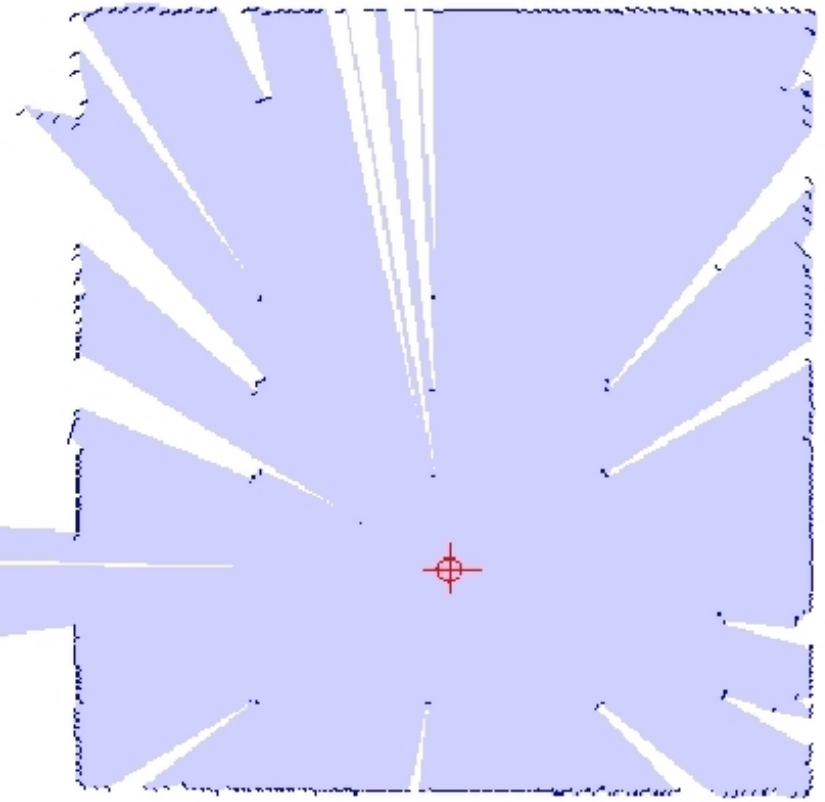
- Today: Front-end choices, SLAM as a graphical optimization problem

Week	Tuesday course	Thursday course
18th February	Introduction (what is SLAM, curriculum, course structure (homeworks, projects), course webpage	Homogeneous coordinates, Transformations, etc.
25th February	Filtering methods (GF, PF)	Filtering methods (GF,PF)
4th March	Pose-only SLAM, Graph SLAM	Introduction to the projects
11th March	Camera as a sensor, camera calibration	Introduction into Visual SLAM, feature extraction, tracking, and matching
18th March	Feature extraction, tracking, and matching	Case study: MonoSLAM (why filter?)
25th March	Bootstrapping: The Relative Pose Problem	Full visual odometry pipeline, Non-linear optimization, Bundle adjustment
1st April	Non-linear optimization, Bundle adjustment	Place recognition, loop closure
8th April	Place recognition, loop closure, the absolute pose problem	Case study: ORB SLAM
15th April	Dense Tracking and Mapping (DTAM), Photometric vs geometric errors, semi-dense opt.	RGBD SLAM, Iterative closest points (Laser-point cloud registration), TSDF
22nd April	SLAM with Stereo and Multi-Perspective cameras	Midterm exam
29th April	Visual-Inertial SLAM	Project discussions/buffer
6th May	Dynamic and Multi-body SLAM	Project discussions/buffer
13th May	Semantic SLAM	Project Presentations

What we have so far

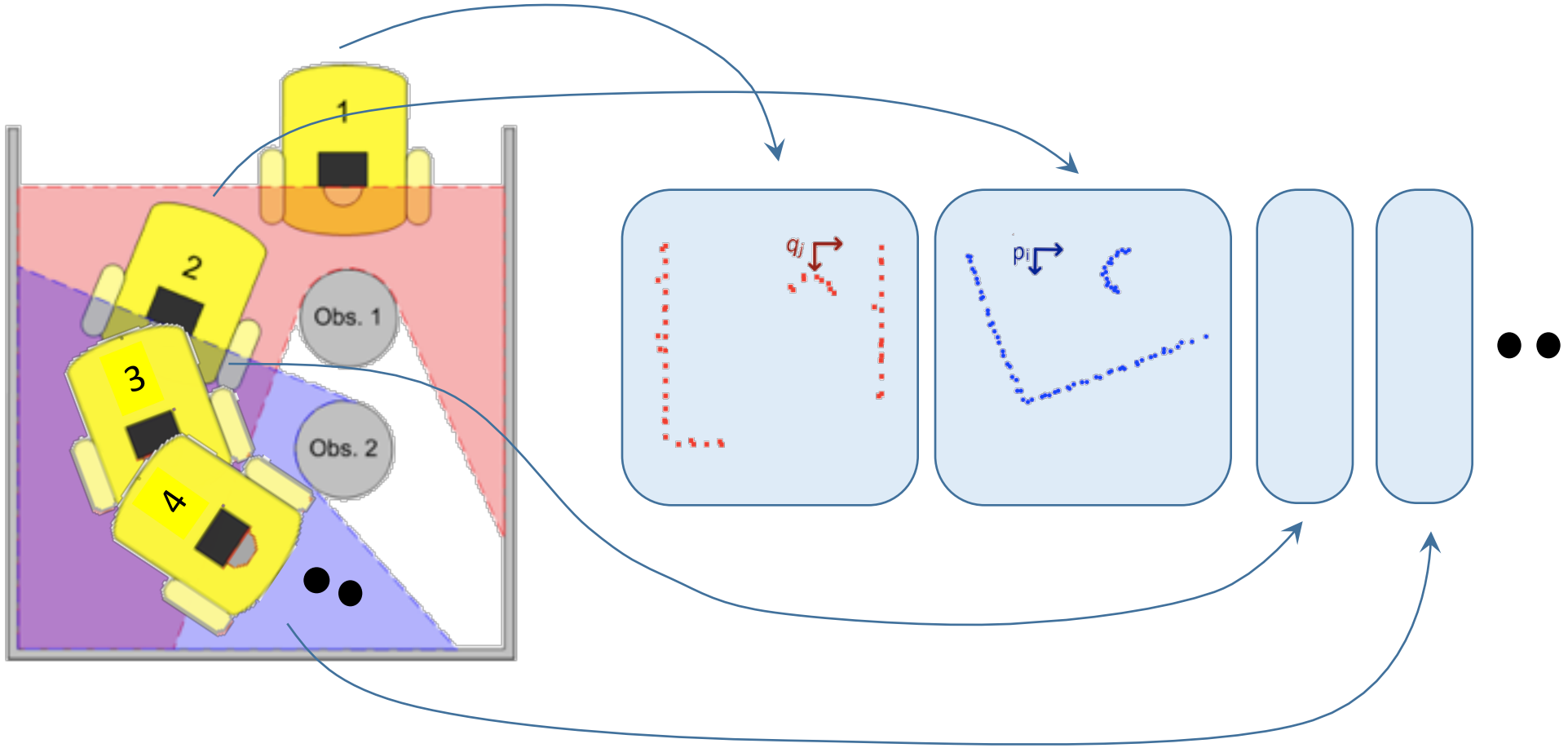
- Robot explores environment and takes scans





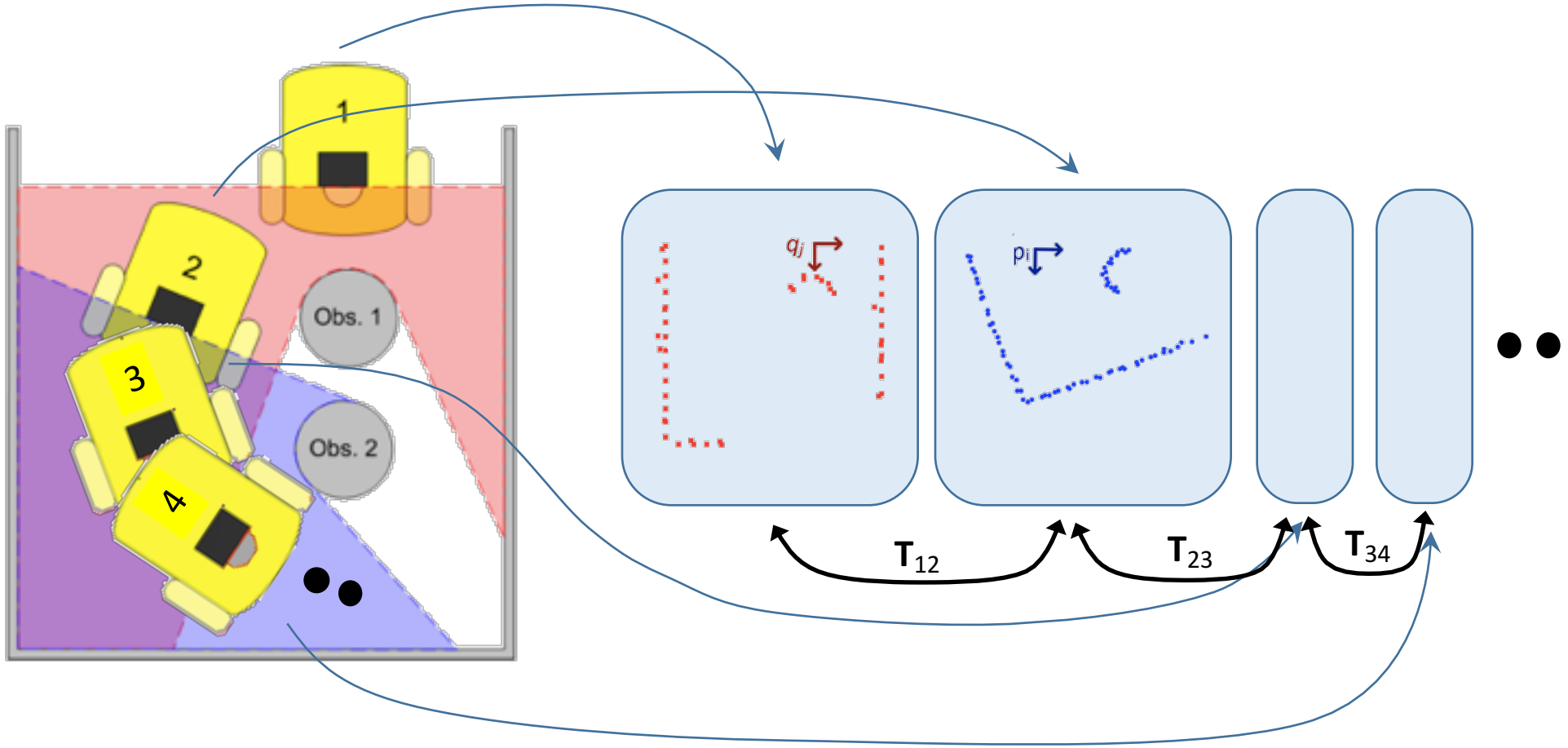
What we have so far

- Robot explores environment and takes scans



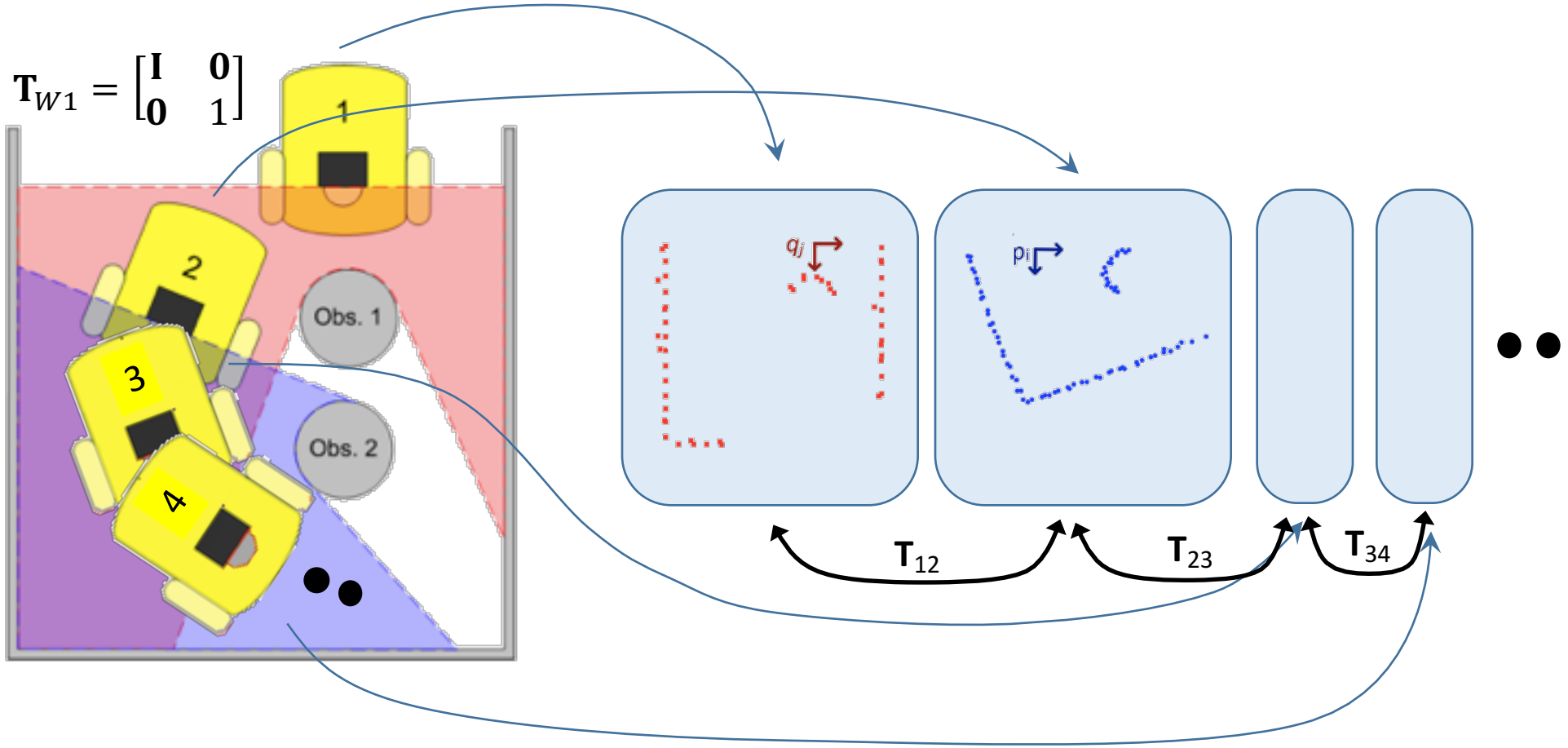
What we have so far

- Scans can be registered with respect to each other



What we have so far

- Global pose estimation: Initialize first pose to zero and identity

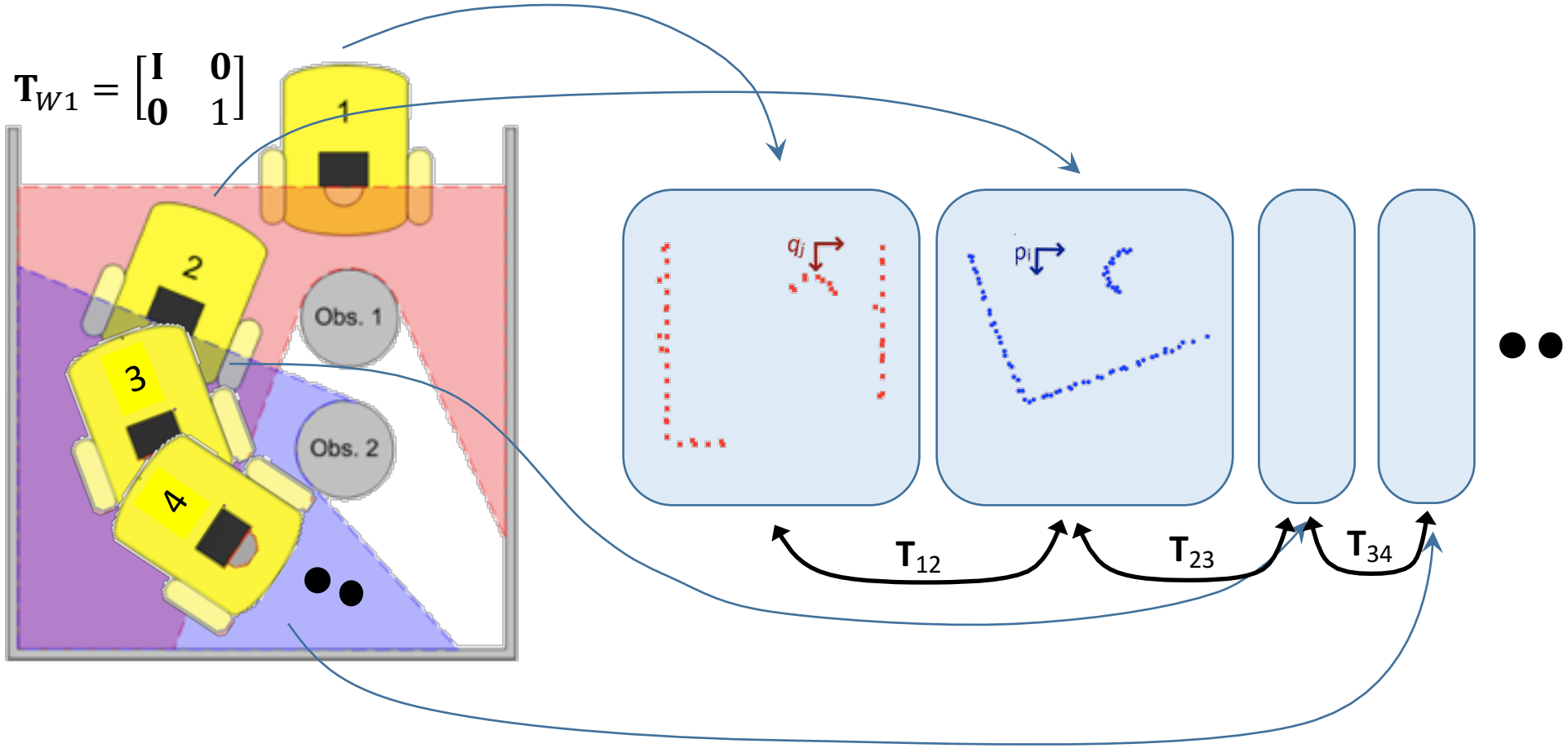


What we have so far

- Global pose estimation: for each new relative pose, update absolute pose

$$\mathbf{T}_{W1} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}$$

$$\mathbf{T}_{W2} = \mathbf{T}_{W1}\mathbf{T}_{12} = \mathbf{T}_{12}$$



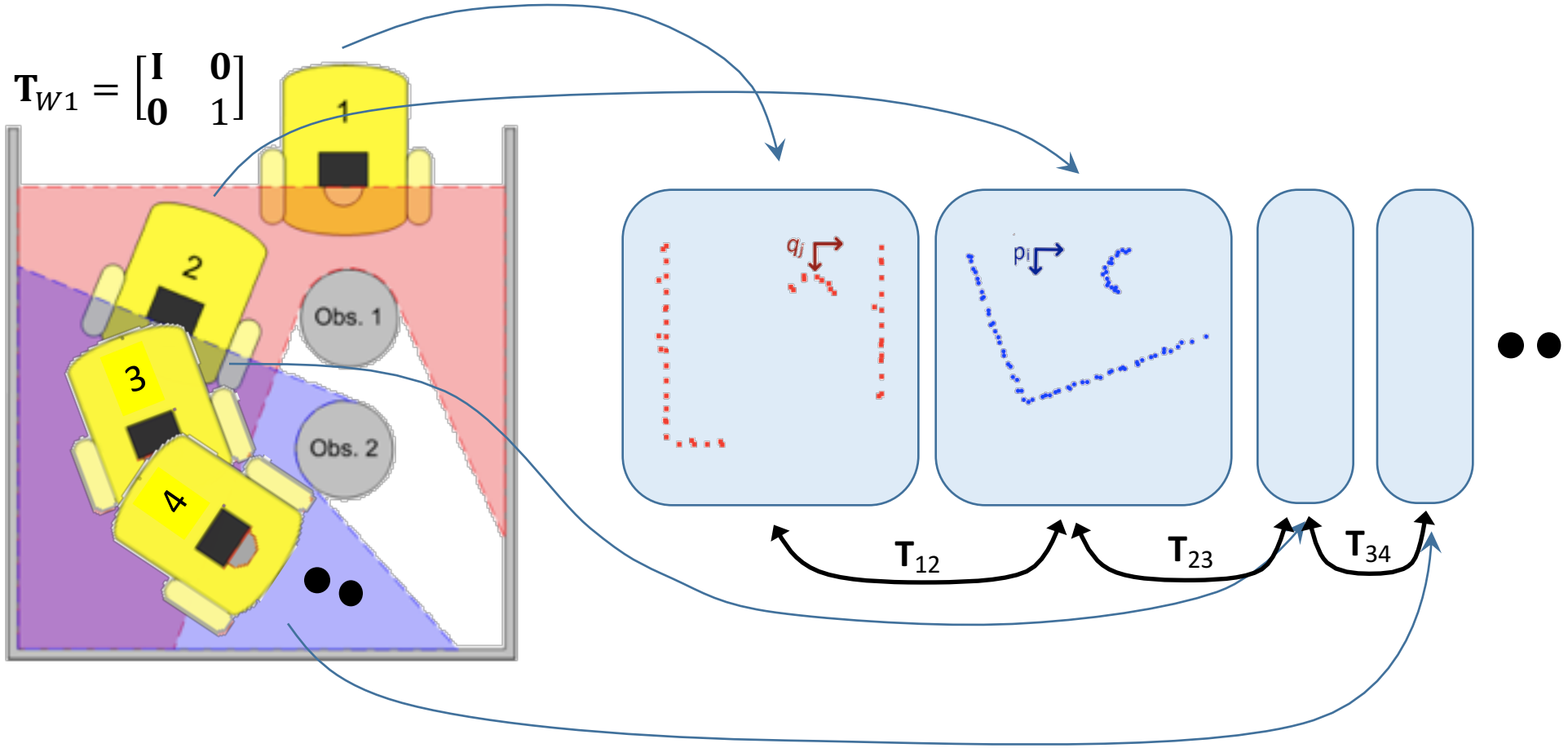
What we have so far

- Global pose estimation: for each new relative pose, update absolute pose

$$\mathbf{T}_{W1} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}$$

$$\mathbf{T}_{W2} = \mathbf{T}_{W1}\mathbf{T}_{12} = \mathbf{T}_{12}$$

$$\mathbf{T}_{W3} = \mathbf{T}_{W2}\mathbf{T}_{23}$$



What we have so far

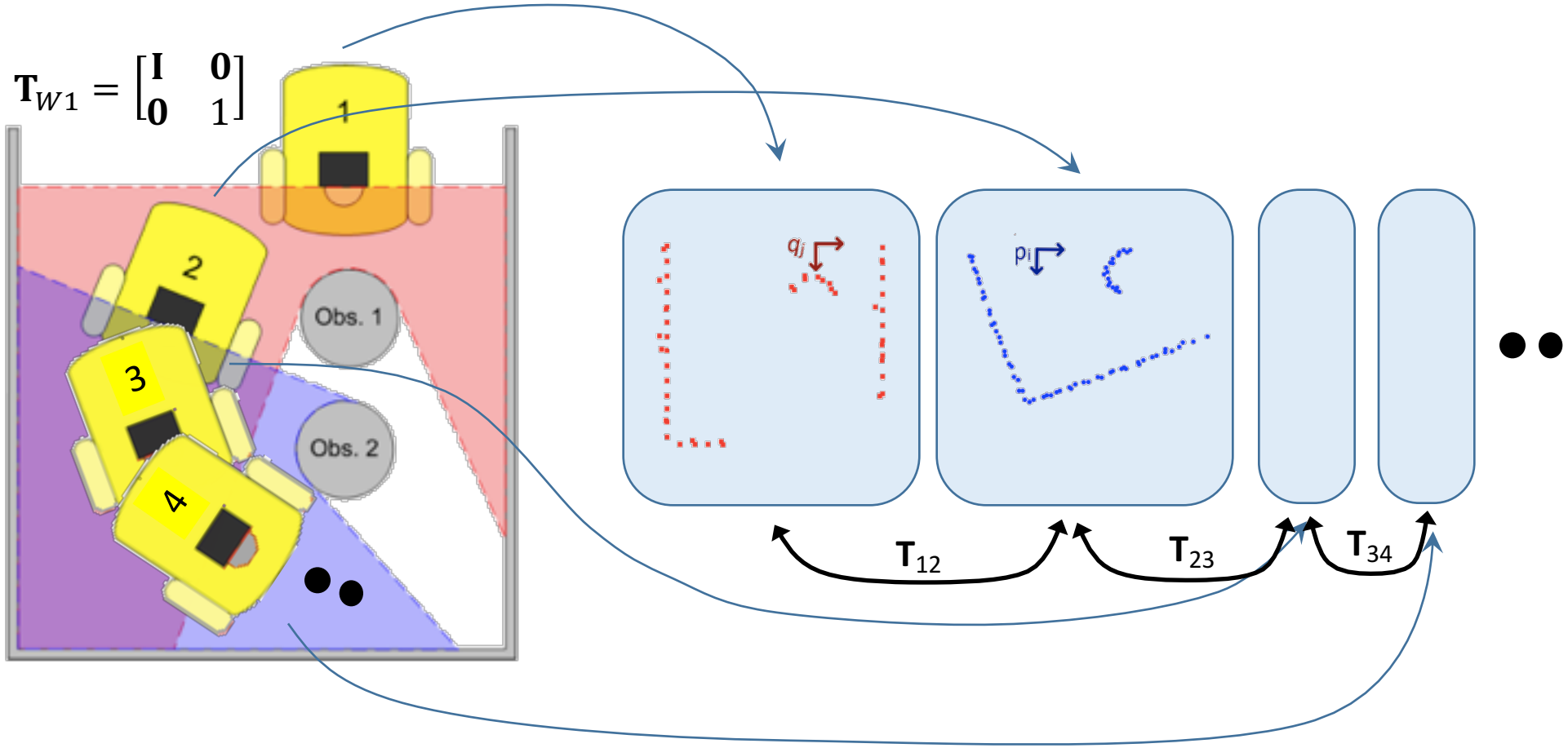
- Global pose estimation: for each new relative pose, update absolute pose

$$\mathbf{T}_{W1} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}$$

$$\mathbf{T}_{W2} = \mathbf{T}_{W1}\mathbf{T}_{12} = \mathbf{T}_{12}$$

$$\mathbf{T}_{W3} = \mathbf{T}_{W2}\mathbf{T}_{23}$$

$$\mathbf{T}_{W4} = \mathbf{T}_{W3}\mathbf{T}_{34}$$



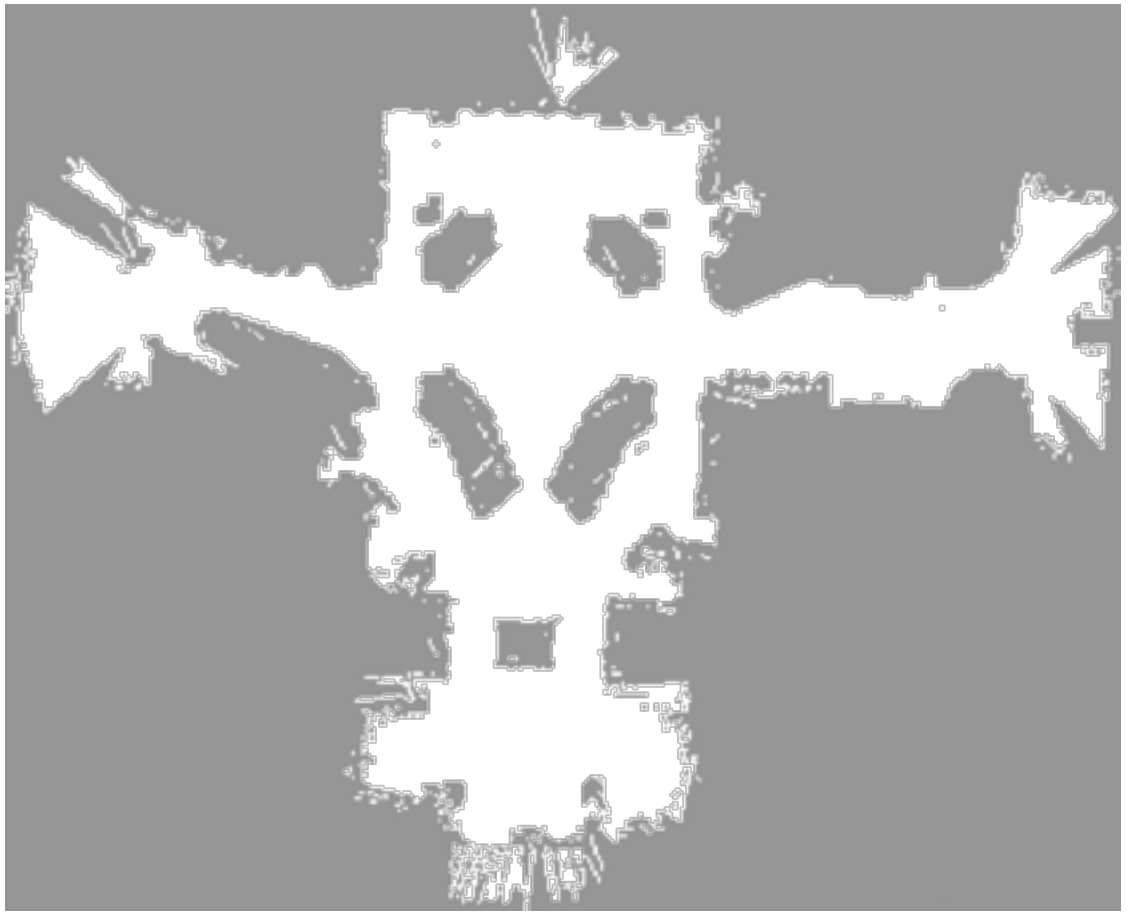
What we have so far

- Simple proposal for a SLAM algorithm:

```
while (robot.isMoving()) {
        S_i = getNewScan();
        T_{i-1,i} = ICP(S_{i-1},S_i);          //find relative pose
        T_{W,i} = T_{W,i-1} T_{i-1,i};         //update absolute pose
        M = [M   T_{W,i}S_i];                  //extend map by new points
}
```

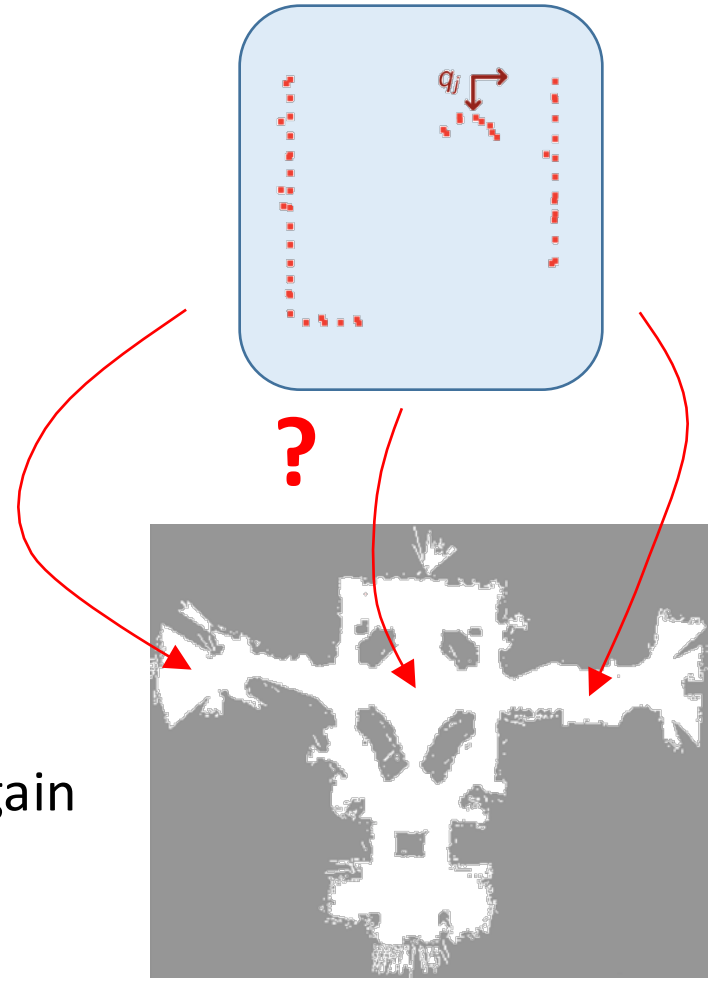
What we have so far

- Result: A map of the environment



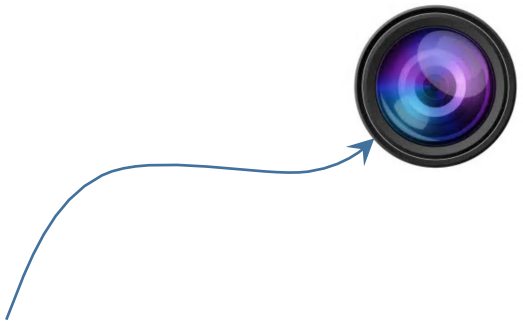
The global localization problem

- Problem formulation
  - Given: a new scan and an existing map of the environment
  - Find: robot position and orientation
- Synonyms:
  - Robot kidnapping problem
  - Absolute pose estimation
  - …
- Relevant in the following scenarios:
  - Robot is switched off, then moved, and then switched on again
  - Map captured by one robot is reused by another/new one
  - Tracking fails but robot still moves

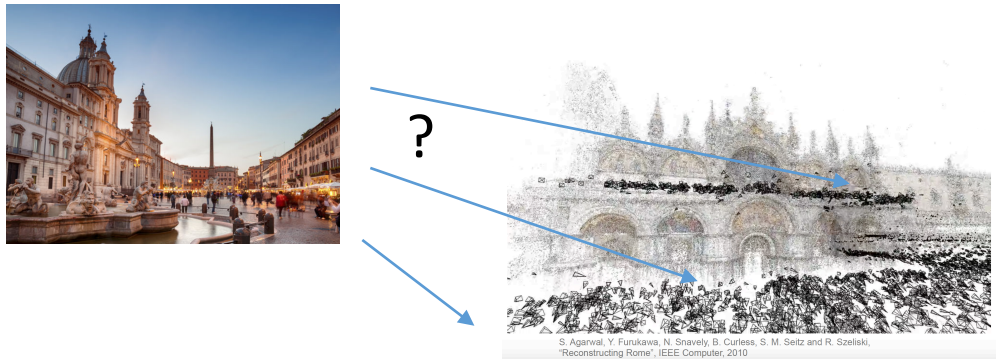


Local vs Global Localization

- Local localization
  - We only track the pose with respect to the previous pose
  - A relative problem
  - Requires only sensor tracking
  - Drifts over time



- Global localization
  - We want to know where we are w.r.t. all prior poses/the map
  - An absolute problem
  - Drift-free under the assumption of a sufficiently accurate map



Global localization: initial pose?

- Since we already have a map, the initial pose cannot be initialized to zero translation and identity rotation
- The initial position can be “anywhere”
- The robot is moving and capturing measurements over time

- → Need probability density distribution
- → Need filter for estimating this distribution
  - Filters perform incremental estimation of complete probability density distribution

Incremental estimation

- Problem definition:
  - Estimate the state x of a system given observations z and controls u (Note: x contains the pose of the robot)
  - Find

$$p(x \mid z, u)$$

  - Robot pose at time t: x_t
    → Robot path up to this time: $\{x_1, x_2, \ldots, x_t\}$
  - Robot motion between time t-1 and t: u_t (control inputs / sensor readings)
    → Sequence of robot relative motions: $\{u_1, u_2, \ldots, u_t\}$
  - At each time t the robot makes measurements z_t
    → Set of all measurements (observations): $\{z_1, z_2, \ldots, z_t\}$
  - The "true" map of the environment: m

Incremental estimation

- Problem definition:
  - Estimate the state x of a system given observations z and controls u (Note: x contains the pose of the robot)
  - Find

$$p(x \mid z, u)$$

- The full SLAM problem: → estimate the posterior $p(x_{1:t}, m_{1:N} \mid z_{1:t}, u_{1:t})$
- **Online localization**: → estimate the posterior $p(x_t \mid z_{1:t}, u_{1:t})$

Filter origins: Recursive Bayes filter

$$bel(x_t) = p(x_t \mid z_{1:t}, u_{1:t})$$

Definition of the belief

Rule 1: Bayes Rule (Review)

- $p(A\ and\ B) = p(A|B)\ p(B)$
- $p(B\ and\ A) = p(B|A)\ p(A) = p(A\ and\ B)$

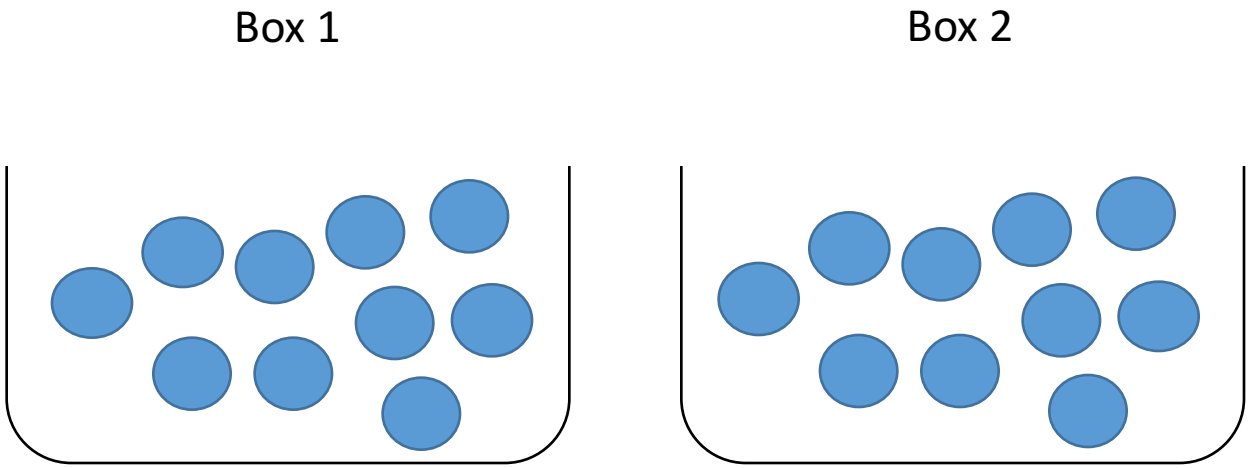
- Therefore:

$$p(A|B) = \frac{p(B|A)p\ (A)}{p(B)}$$

- The "working" version: $p(A|B) = \frac{p(B|A)p(A)}{p(B|A)p(A)+p(B|not-A)p(not-A)}$

Rule 1: Bayes rule

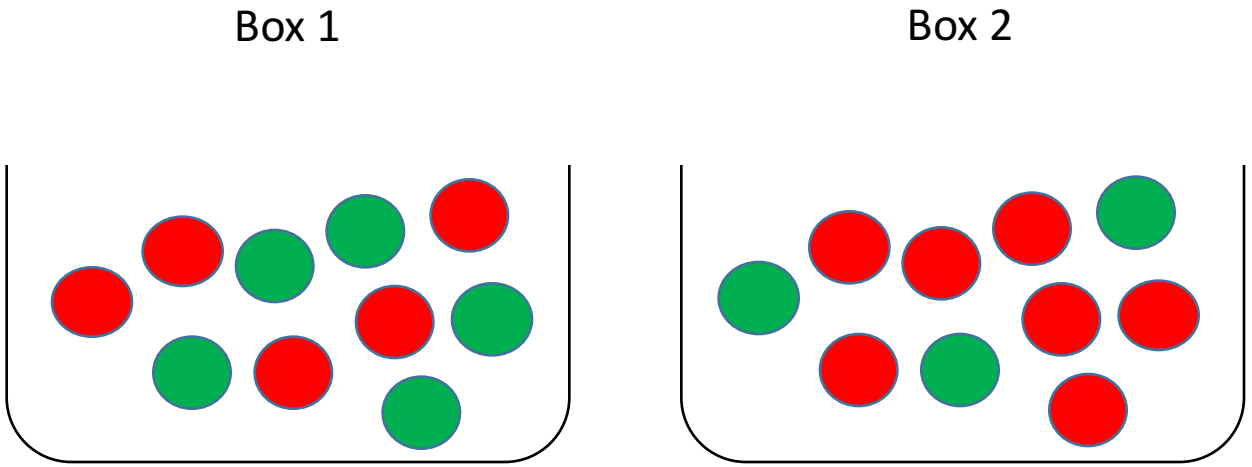
- Review



- Two containers contain a mix of red and green balls.
- A random ball is taken. The ball is green.
- What is the probability of it coming from container 1?

Rule 1: Bayes rule

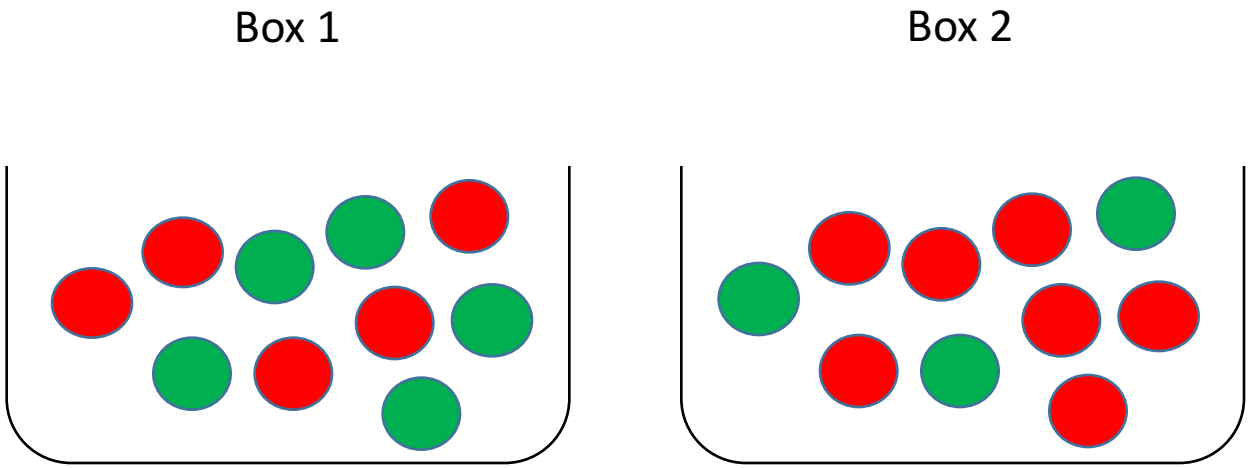
- Review



- Two containers contain a mix of red and green balls.
- A random ball is taken. The ball is green.
- What is the probability of it coming from container 1?

Rule 1: Bayes rule

- Review



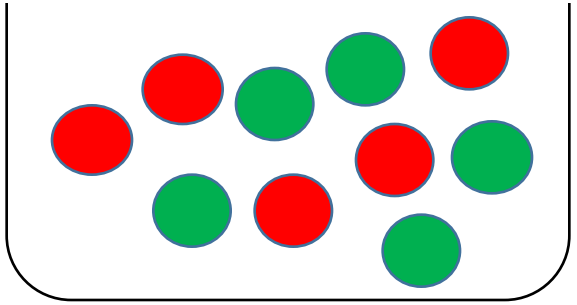
g = the ball is green
b_1 = box 1 is selected
b_2 = box 2 is selected

$p(b_1|g) = ?$

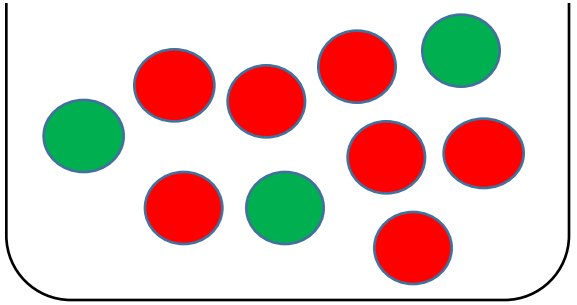
Rule 1: Bayes rule

- Review

Box 1



Box 2



$p(b_1|g) = ?$

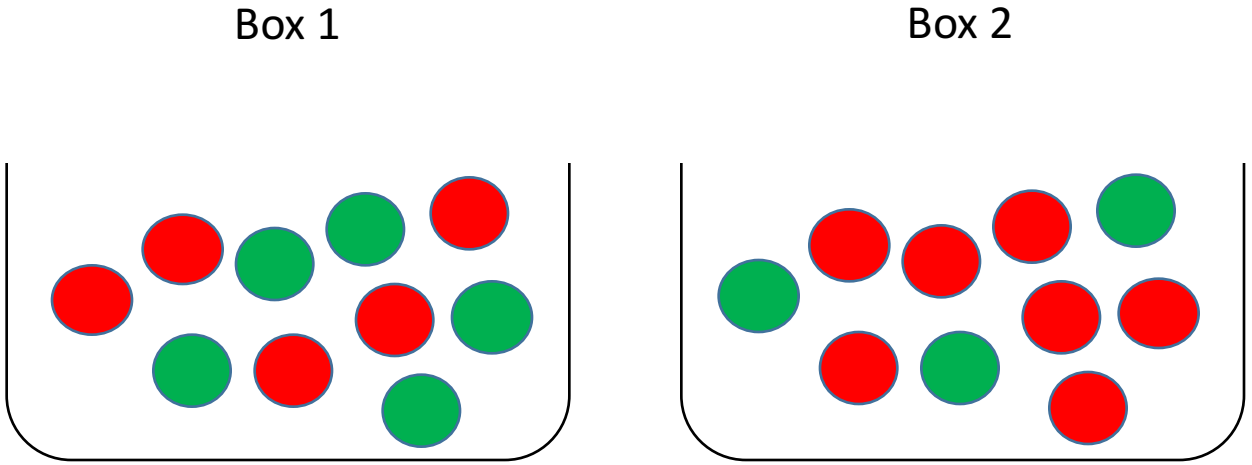
g = the ball is green
b_1 = box 1 is selected
b_2 = box 2 is selected

$p(b_1) = 0.5$
$p(b_2) = 0.5$

$p(g|b_1) = 0.5$
$p(g|b_2) = 0.3$

Rule 1: Bayes rule

- Review



$$p(b_1|g) = \frac{p(g|b_1)\,p(b_1)}{p(g|b_1)\,p(b_1) + p(g|b_2)\,p(b_2)}$$

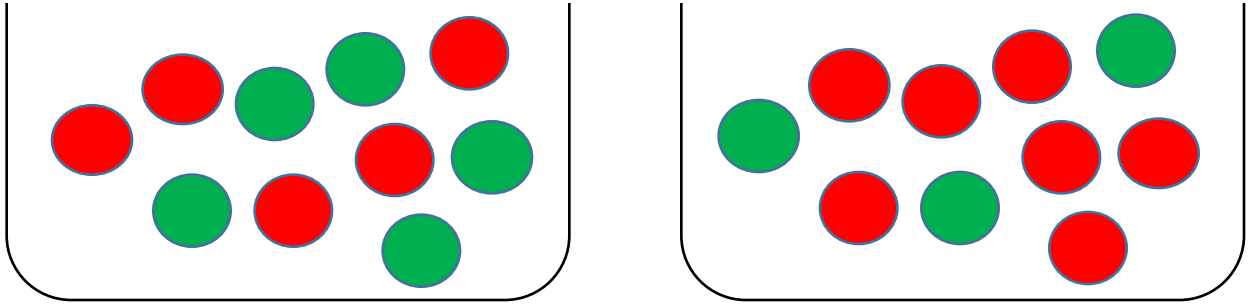
g = the ball is green
b_1 = box 1 is selected
b_2 = box 2 is selected

$p(b_1) = 0.5$
$p(b_2) = 0.5$

$p(g|b_1) = 0.5$
$p(g|b_2) = 0.3$

Rule 1: Bayes rule

- Review



$$p(b_1|g) = \frac{0.5\ 0.5}{0.5\ 0.5 + 0.5\ 0.3} \longrightarrow 62.5\%$$

g = the ball is green
b_1 = box 1 is selected
b_2 = box 2 is selected

$p(b_1) = 0.5$
$p(b_2) = 0.5$

$p(g|b_1) = 0.5$
$p(g|b_2) = 0.3$

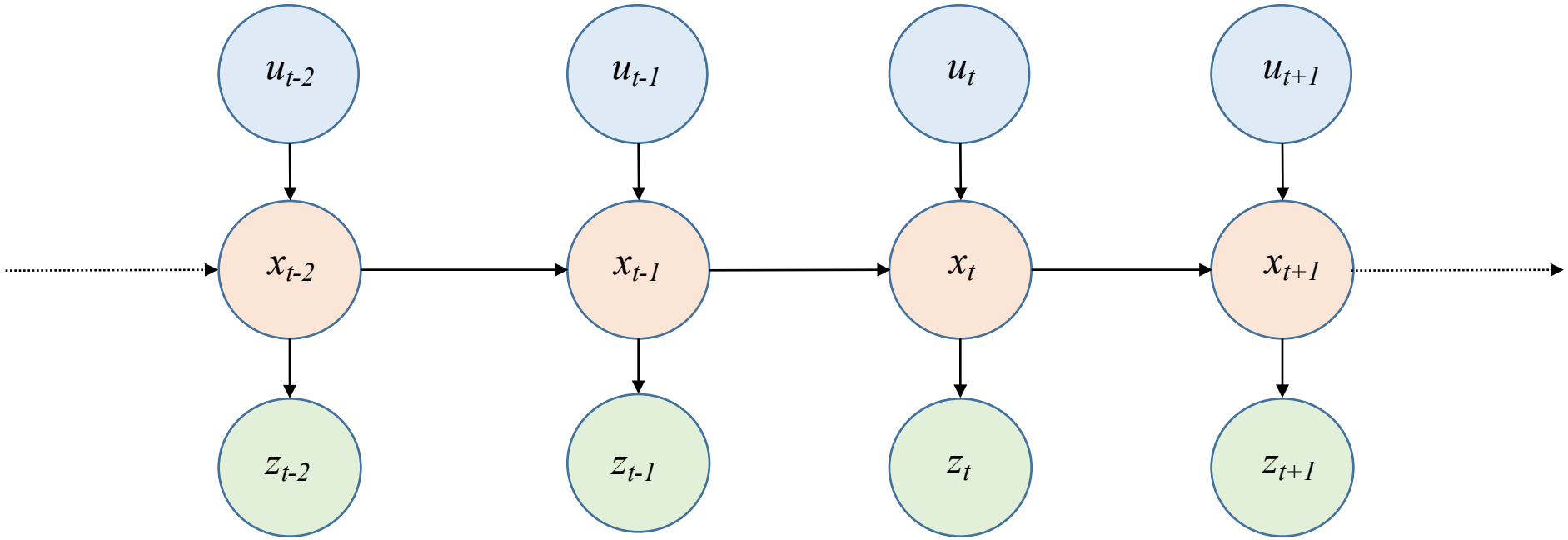
Recursive Bayes filter

$$bel(x_t) = p(x_t \mid z_{1:t}, u_{1:t})$$
$$= \eta\, p(z_t \mid x_t, z_{1:t-1}, u_{1:t})\, p(x_t \mid z_{1:t-1}, u_{1:t})$$

Bayes' rule

Rule 2: Markov assumption (Review)

- The “memoryless property” of a stochastic process
- Hidden Markov Model (HMM):



Recursive Bayes filter

$$
\begin{aligned}
bel(x_t) &= p(x_t \mid z_{1:t}, u_{1:t}) \\
&= \eta\, p(z_t \mid x_t, z_{1:t-1}, u_{1:t})\, p(x_t \mid z_{1:t-1}, u_{1:t}) \\
&= \eta\, p(z_t \mid x_t)\, p(x_t \mid z_{1:t-1}, u_{1:t})
\end{aligned}
$$

Markov assumption

Recursive Bayes filter

$$
\begin{aligned}
bel(x_t) &= p(x_t \mid z_{1:t}, u_{1:t}) \\
&= \eta\, p(z_t \mid x_t, z_{1:t-1}, u_{1:t})\; p(x_t \mid z_{1:t-1}, u_{1:t}) \\
&= \eta\, p(z_t \mid x_t)\; p(x_t \mid z_{1:t-1}, u_{1:t}) \\
&= \eta\, p(z_t \mid x_t) \int_{x_{t-1}} p(x_t \mid x_{t-1}, z_{1:t-1}, u_{1:t}) \\
&\qquad\qquad p(x_{t-1} \mid z_{1:t-1}, u_{1:t})\; dx_{t-1}
\end{aligned}
$$

Law of total probability

Recursive Bayes filter

$$
\begin{aligned}
bel(x_t) &= p(x_t \mid z_{1:t}, u_{1:t}) \\
&= \eta\, p(z_t \mid x_t, z_{1:t-1}, u_{1:t})\; p(x_t \mid z_{1:t-1}, u_{1:t}) \\
&= \eta\, p(z_t \mid x_t)\; p(x_t \mid z_{1:t-1}, u_{1:t}) \\
&= \eta\, p(z_t \mid x_t) \int_{x_{t-1}} p(x_t \mid x_{t-1}, z_{1:t-1}, u_{1:t}) \\
&\qquad\qquad p(x_{t-1} \mid z_{1:t-1}, u_{1:t})\; dx_{t-1} \\
&= \eta\, p(z_t \mid x_t) \int_{x_{t-1}} p(x_t \mid x_{t-1}, u_t)\; p(x_{t-1} \mid z_{1:t-1}, u_{1:t})\; dx_{t-1}
\end{aligned}
$$

Markov assumption

Recursive Bayes filter

$$
\begin{aligned}
bel(x_t) &= p(x_t \mid z_{1:t}, u_{1:t}) \\
&= \eta\, p(z_t \mid x_t, z_{1:t-1}, u_{1:t})\; p(x_t \mid z_{1:t-1}, u_{1:t}) \\
&= \eta\, p(z_t \mid x_t)\; p(x_t \mid z_{1:t-1}, u_{1:t}) \\
&= \eta\, p(z_t \mid x_t) \int_{x_{t-1}} p(x_t \mid x_{t-1}, z_{1:t-1}, u_{1:t}) \\
&\qquad\qquad p(x_{t-1} \mid z_{1:t-1}, u_{1:t})\; dx_{t-1} \\
&= \eta\, p(z_t \mid x_t) \int_{x_{t-1}} p(x_t \mid x_{t-1}, u_t)\; p(x_{t-1} \mid z_{1:t-1}, u_{1:t})\; dx_{t-1} \\
&= \eta\, p(z_t \mid x_t) \int_{x_{t-1}} p(x_t \mid x_{t-1}, u_t)\; p(x_{t-1} \mid z_{1:t-1}, u_{1:t-1})\; dx_{t-1}
\end{aligned}
$$

Markov assumption

Recursive Bayes filter

$$
\begin{aligned}
bel(x_t) &= p(x_t \mid z_{1:t}, u_{1:t}) \\
&= \eta\, p(z_t \mid x_t, z_{1:t-1}, u_{1:t})\; p(x_t \mid z_{1:t-1}, u_{1:t}) \\
&= \eta\, p(z_t \mid x_t)\; p(x_t \mid z_{1:t-1}, u_{1:t}) \\
&= \eta\, p(z_t \mid x_t) \int_{x_{t-1}} p(x_t \mid x_{t-1}, z_{1:t-1}, u_{1:t}) \\
&\qquad\qquad p(x_{t-1} \mid z_{1:t-1}, u_{1:t})\; dx_{t-1} \\
&= \eta\, p(z_t \mid x_t) \int_{x_{t-1}} p(x_t \mid x_{t-1}, u_t)\; p(x_{t-1} \mid z_{1:t-1}, u_{1:t})\; dx_{t-1} \\
&= \eta\, p(z_t \mid x_t) \int_{x_{t-1}} p(x_t \mid x_{t-1}, u_t)\; p(x_{t-1} \mid z_{1:t-1}, u_{1:t-1})\; dx_{t-1} \\
&= \eta\, p(z_t \mid x_t) \int_{x_{t-1}} p(x_t \mid x_{t-1}, u_t)\; \underline{bel(x_{t-1})}\; dx_{t-1}
\end{aligned}
$$

Recursive term

Prediction and Correction steps

- Bayes filter can be written as two-step process
- Prediction step

$$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\ bel(x_{t-1})\, dx_{t-1}$$

- Correction step

$$bel(x_t) = \eta\ p(z_t \mid x_t)\ \overline{bel}(x_t)$$

Different realizations

- The Bayes filter is a framework for recursive state estimation
- There are different realizations
- Different properties
  - Linear vs. non-linear models for motion and observation models
  - Gaussian distributions only?
  - Parametric vs. non-parametric filters
  - …
- Examples:
  - Gaussian filters (KF, EKF, for linear and linearized models)
  - **Particle filter** (non-parametric, for arbitrary models, requires sampling)

Particle filter is needed for solving the robot kidnapping problem!

Motion and observation model

- Prediction step

$$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\ bel(x_{t-1})\, dx_{t-1}$$

motion model

- Correction step

$$bel(x_t) = \eta\, p(z_t \mid x_t)\ \overline{bel}(x_t)$$

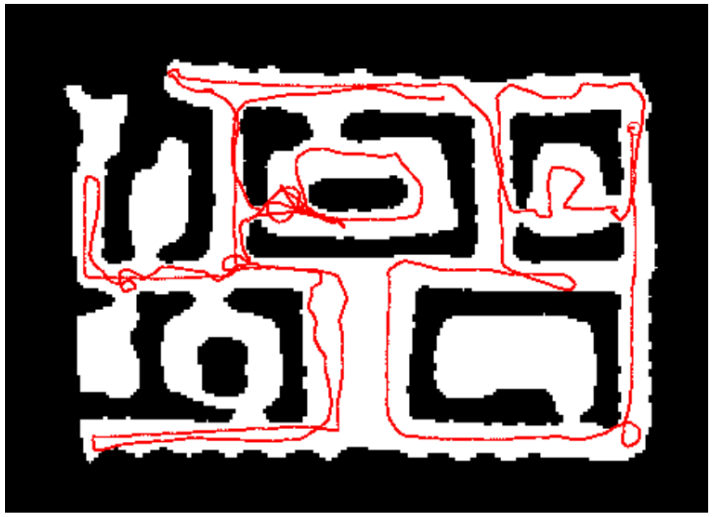
sensor or observation model

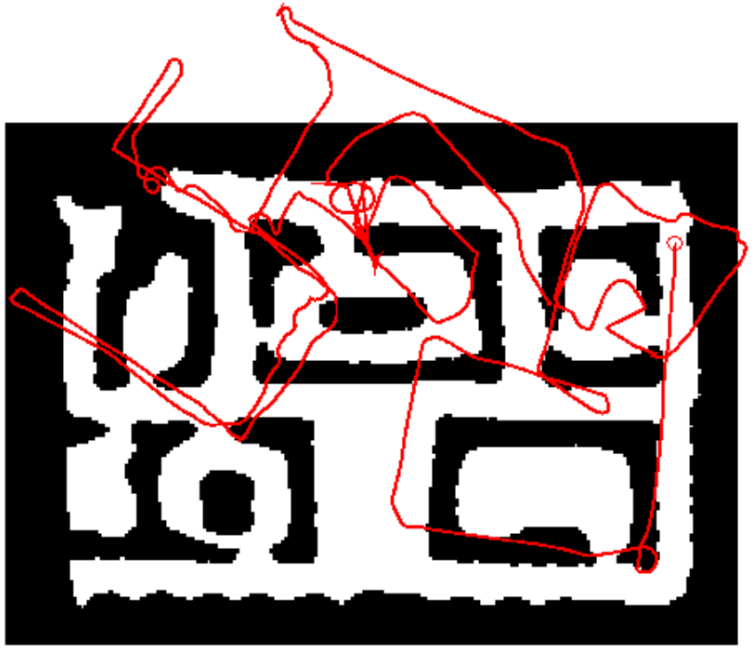
Motion model

$$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\ bel(x_{t-1})\, dx_{t-1}$$

Motion model

- A robot's motion is inherently uncertain
- How can we model the uncertainty?





Probabilistic motion model

- Specifies a posterior condition that action u carries the robot from x to x'

$$p(x_t \mid u_t, x_{t-1})$$

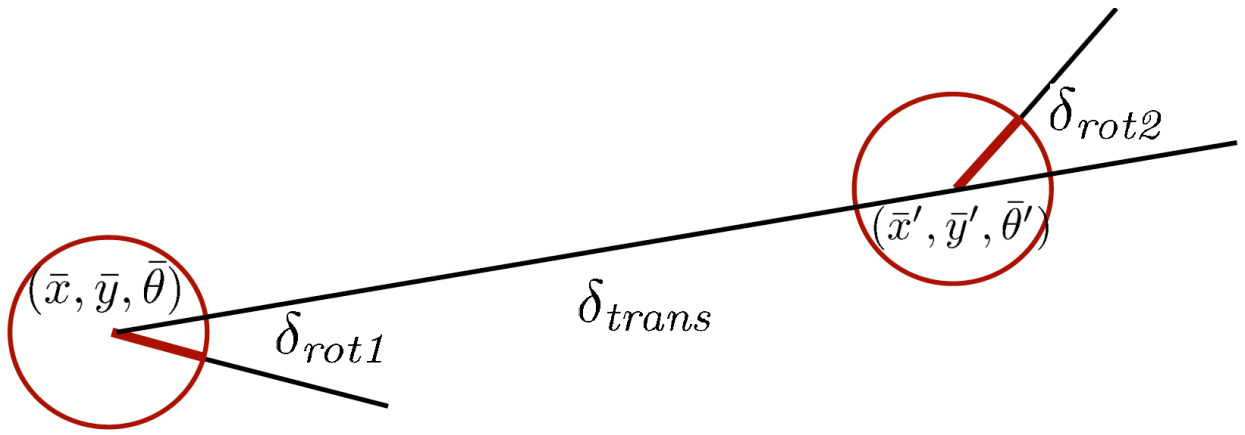
Typical motion models

- In practice, one often finds the following types of motion models:
  - Odometry-based: For systems that are equipped with wheel encoder
  - Sensor-based: For systems that have an exteroceptive sensor, such as a Laser scanner → using **ICP for scan alignment**
  - Control-based: use the control inputs
  - Velocity-based: For passive systems that have no odometry/interoceptive sensor and for which motion is hard to determine (e.g.: single camera)

Odometry model

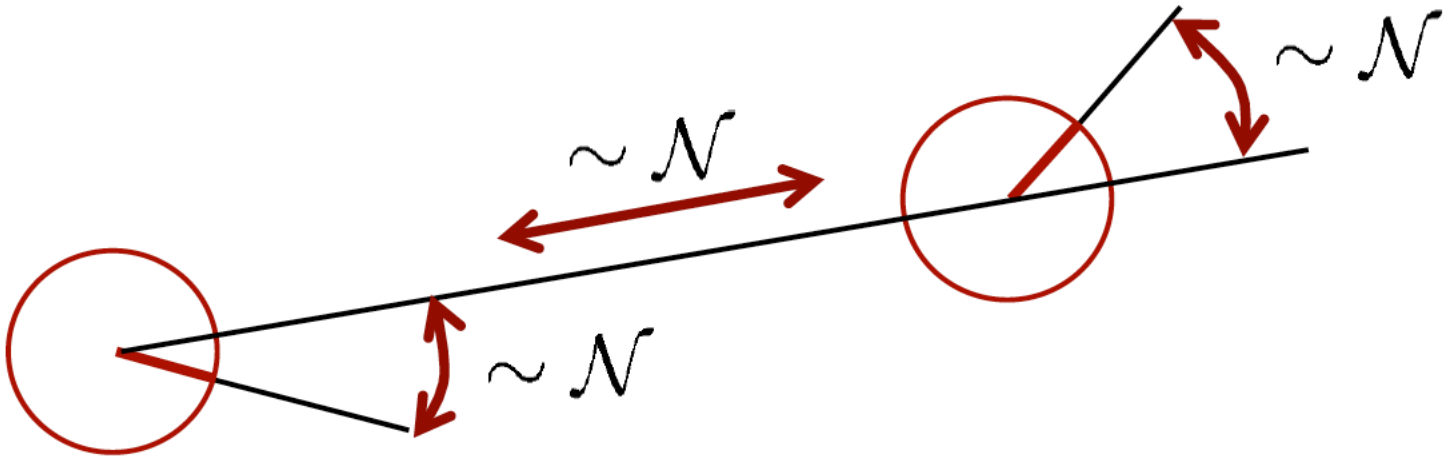
- Agent moves from $(\bar{x}, \bar{y}, \bar{\theta})$ to $(\bar{x}', \bar{y}', \bar{\theta}')$
- Odometry information $u = (\delta_{rot1}, \delta_{trans}, \delta_{rot2})$

$$
\begin{aligned}
\delta_{trans} &= \sqrt{(\bar{x}' - \bar{x})^2 + (\bar{y}' - \bar{y})^2} \\
\delta_{rot1} &= \operatorname{atan2}(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta} \\
\delta_{rot2} &= \bar{\theta}' - \bar{\theta} - \delta_{rot1}
\end{aligned}
$$



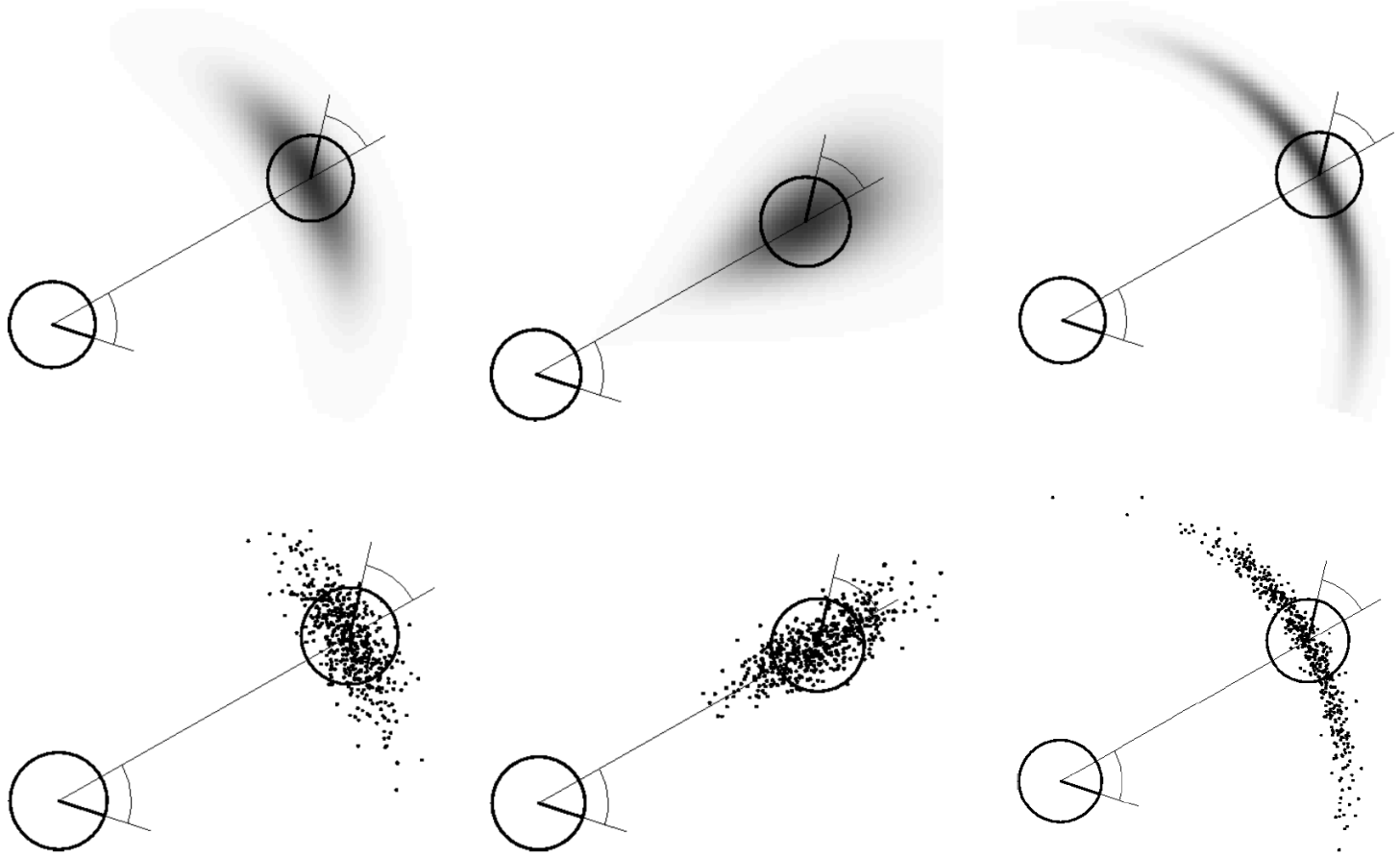
Odometry model

- Probability distribution
- Noise in odometry $u = (\delta_{rot1}, \delta_{trans}, \delta_{rot2})$
- Example: Gaussian noise $u \sim \mathcal{N}(0, \Sigma)$



Odometry model

- Examples (odometry-based)



Sensor model

$$bel(x_t) = \eta \; p(z_t \mid x_t) \; \overline{bel}(x_t)$$

Model for Laser Scanners

- Scan z consists of K measurements

$$z_t = \{z_t^1, \ldots, z_t^k\}$$

- Individual measurements are independent given the robot position

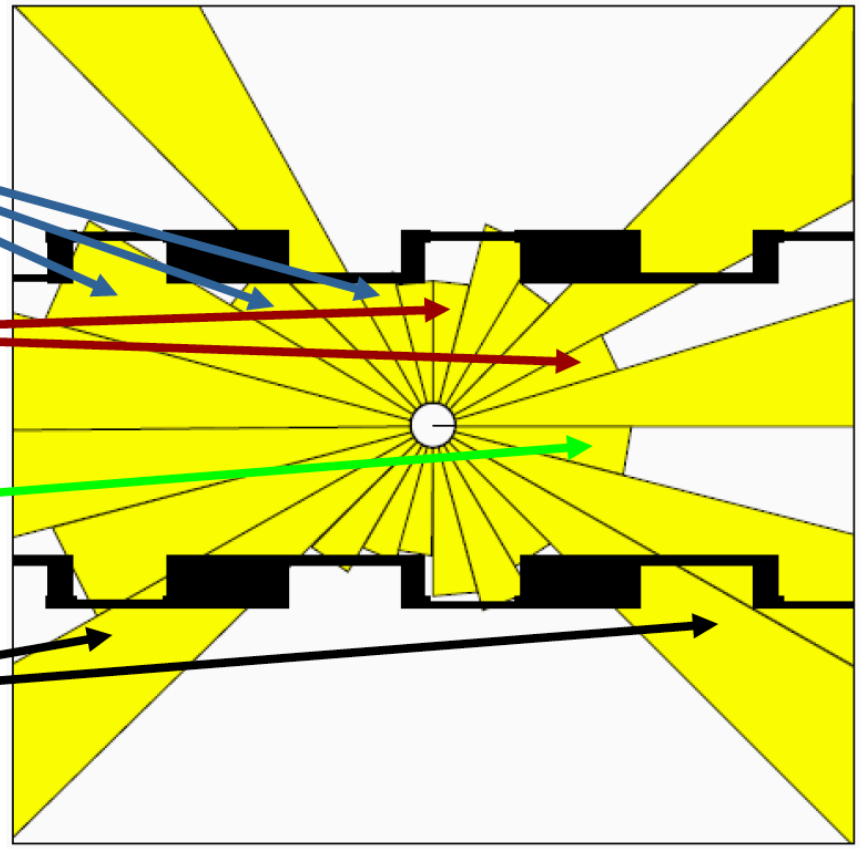
$$p(z_t \mid x_t, m) = \prod_{i=1}^{k} p(z_t^i \mid x_t, m)$$

- Measurement z_t is the distance of the environment from the robot

Measurement errors

- Typical measurement errors of range measurements

1. Beams reflected by obstacles
2. Beams reflected by persons / caused by crosstalk
3. Random measurements
4. Maximum range measurements

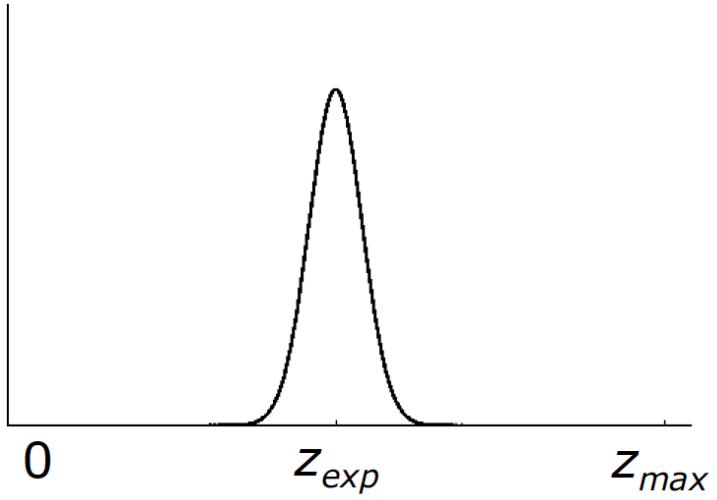


Proximity measurement

- Measurement can be caused by …
  - a known obstacle.
  - cross-talk.
  - an unexpected obstacle (people, furniture, …).
  - missing all obstacles (total reflection, glass, …).
- Noise is due to uncertainty …
  - in measuring distance to known obstacle.
  - in position of known obstacles.
  - in position of additional obstacles.
  - whether obstacle is missed.

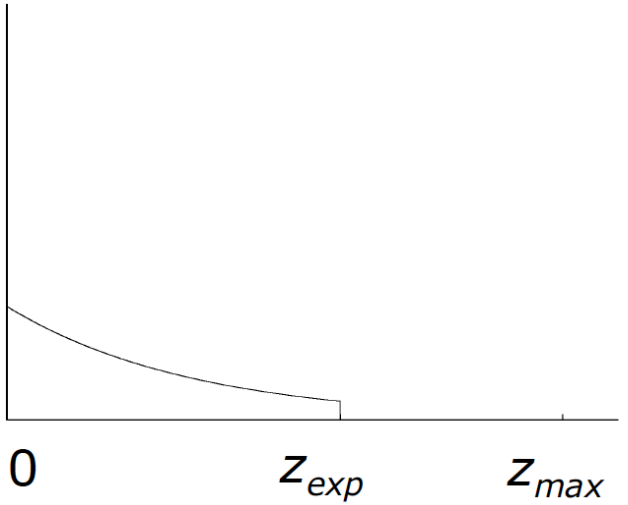
Beam-based proximity model

Measurement noise



$$P_{hit}(z \mid x,m) = \eta \frac{1}{\sqrt{2\pi b}} e^{-\frac{1}{2}\frac{(z-z_{\exp})^2}{b}}$$

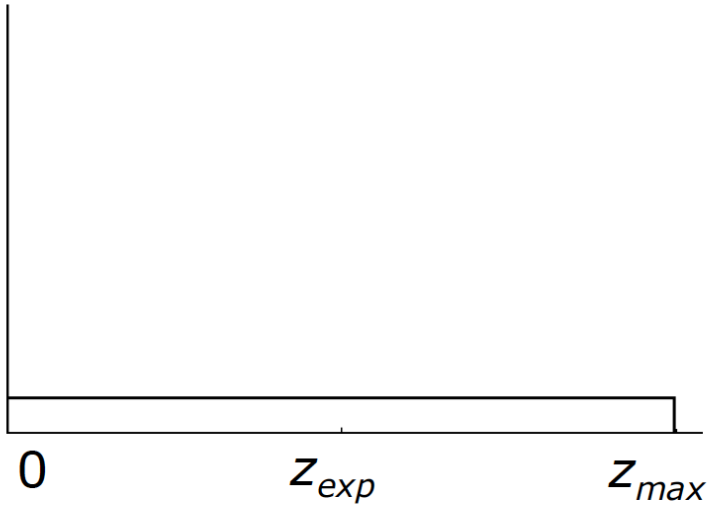
Unexpected obstacles



$$P_{\text{unexp}}(z \mid x,m) = \begin{cases} \eta\, \lambda\, \mathrm{e}^{-\lambda z} & z < z_{\exp} \\ 0 & otherwise \end{cases}$$

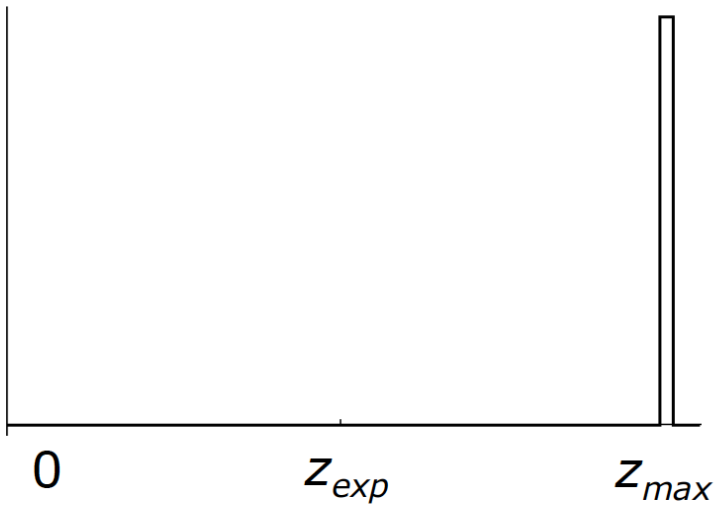
Beam-based proximity model

Random measurement



$$P_{rand}(z \mid x, m) = \eta \frac{1}{z_{\max}}$$

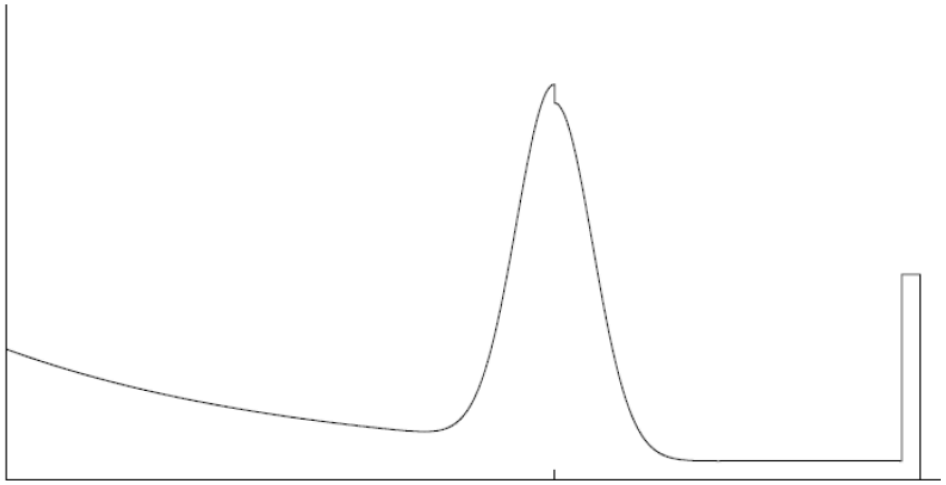
Max range



$$P_{\max}(z \mid x, m) = \eta \frac{1}{z_{small}}$$

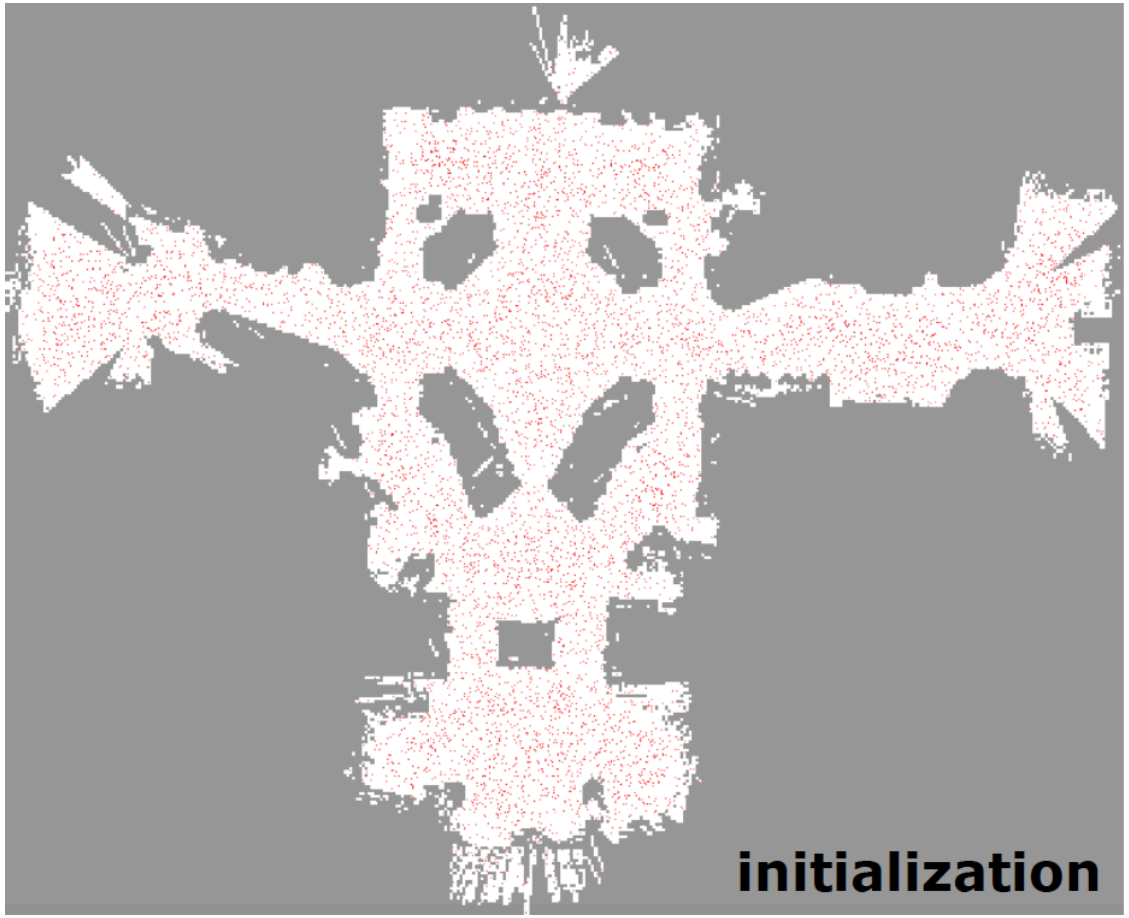
if $z > z_{\max}$

Resulting mixture density



$$P(z \mid x,m) = \begin{pmatrix} \alpha_{\text{hit}} \\ \alpha_{\text{unexp}} \\ \alpha_{\text{max}} \\ \alpha_{\text{rand}} \end{pmatrix}^{T} \cdot \begin{pmatrix} P_{\text{hit}}(z \mid x,m) \\ P_{\text{unexp}}(z \mid x,m) \\ P_{\text{max}}(z \mid x,m) \\ P_{\text{rand}}(z \mid x,m) \end{pmatrix}$$

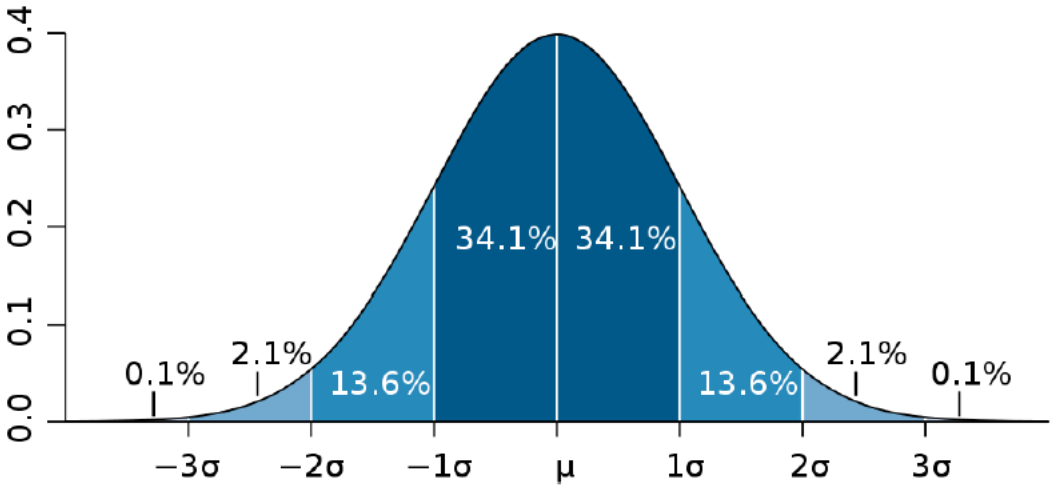
Particle Filter Motivation



Particle Filter Motivation

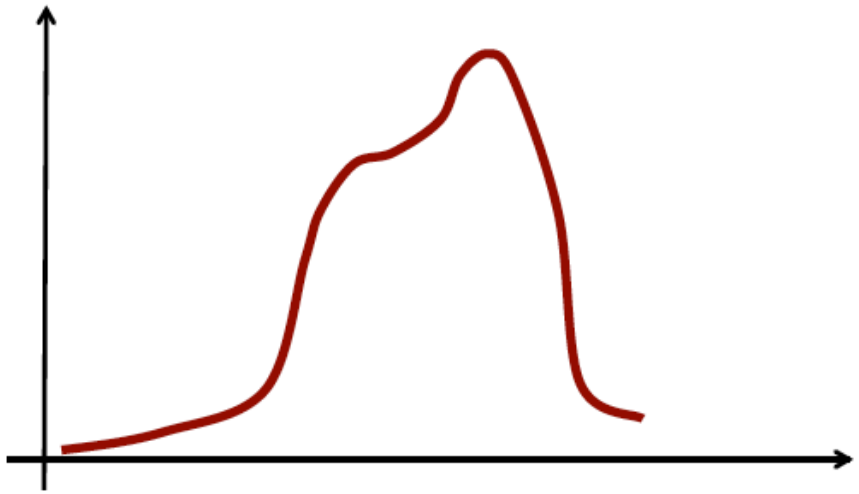
- The Kalman filter and its variants can only model Gaussian distributions

$$p(x) = \det(2\pi\Sigma)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$



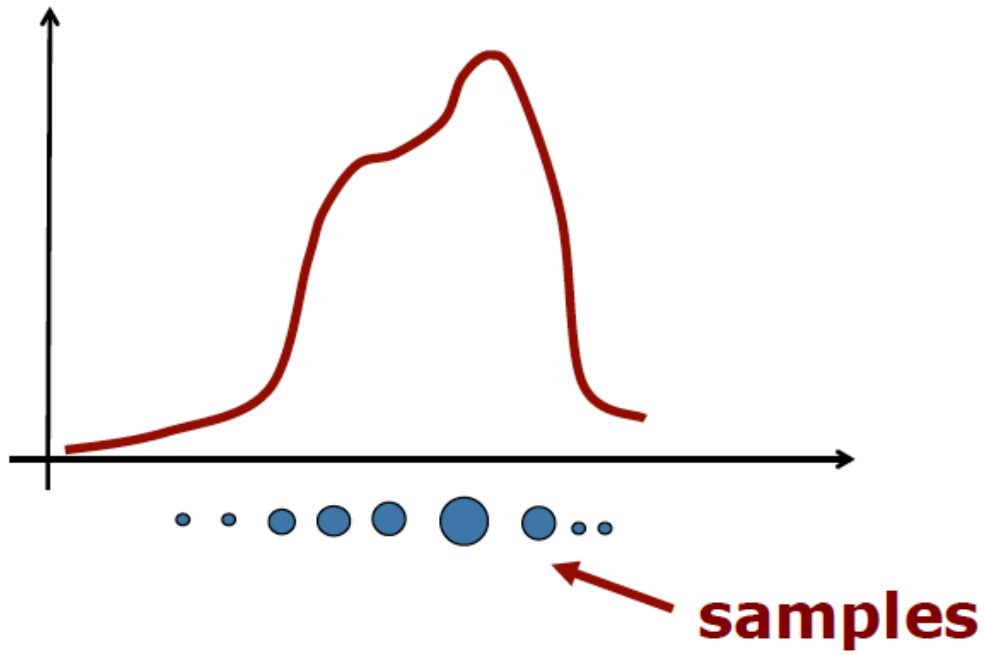
Particle Filter Motivation

- Goal: approach for dealing with arbitrary distributions



Key idea: Samples

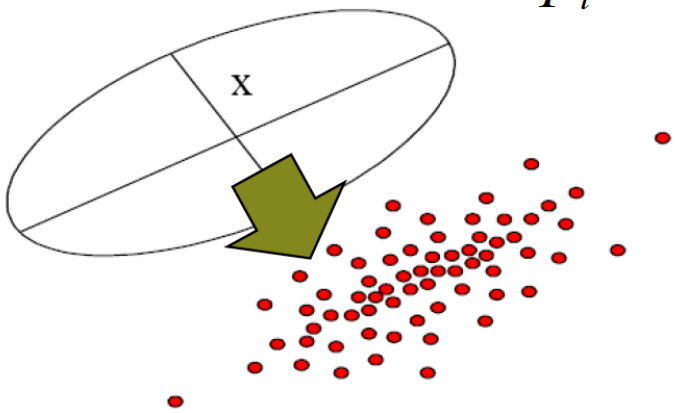
- Use multiple samples to represent arbitrary distributions



Particle Filtering

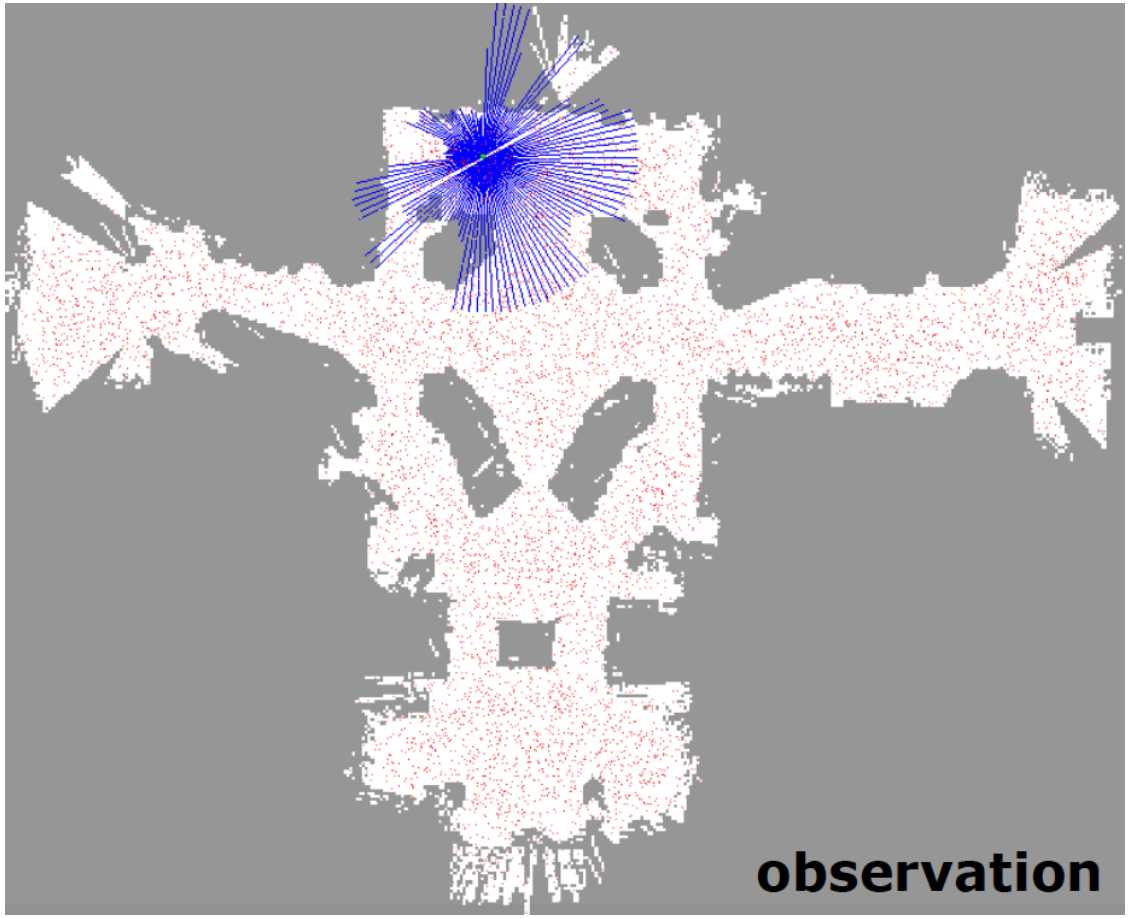
- **Particle Filtering**
  - Represents belief by a series of samples
  - Each Particle = a hypothesis of the state (= a suggested pose & map) with an associated weight
    (all weights should add up to 1)
  - Follows the “predict/measure/update” approach

$$p_i = \{y_{t_i}, w_i\}$$

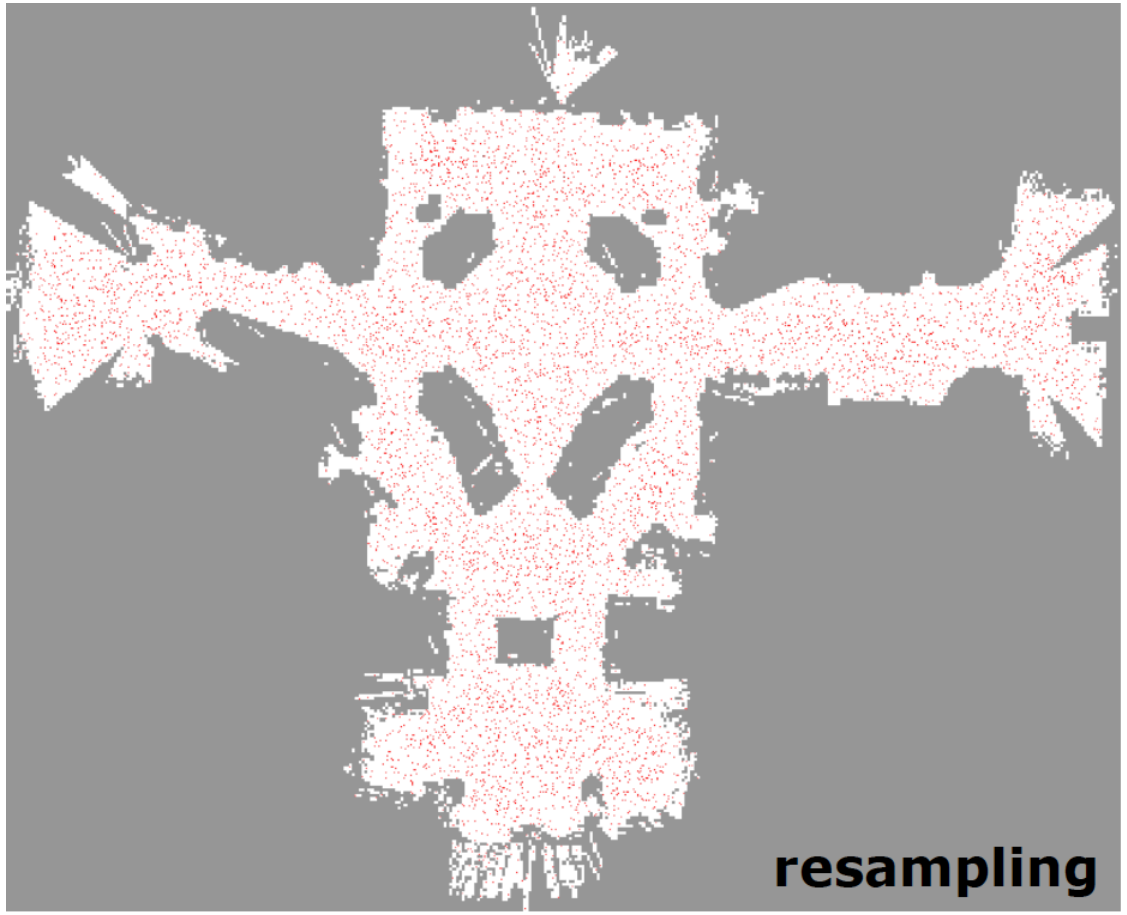


Probability distribution (ellipse) as particle set (red dots)

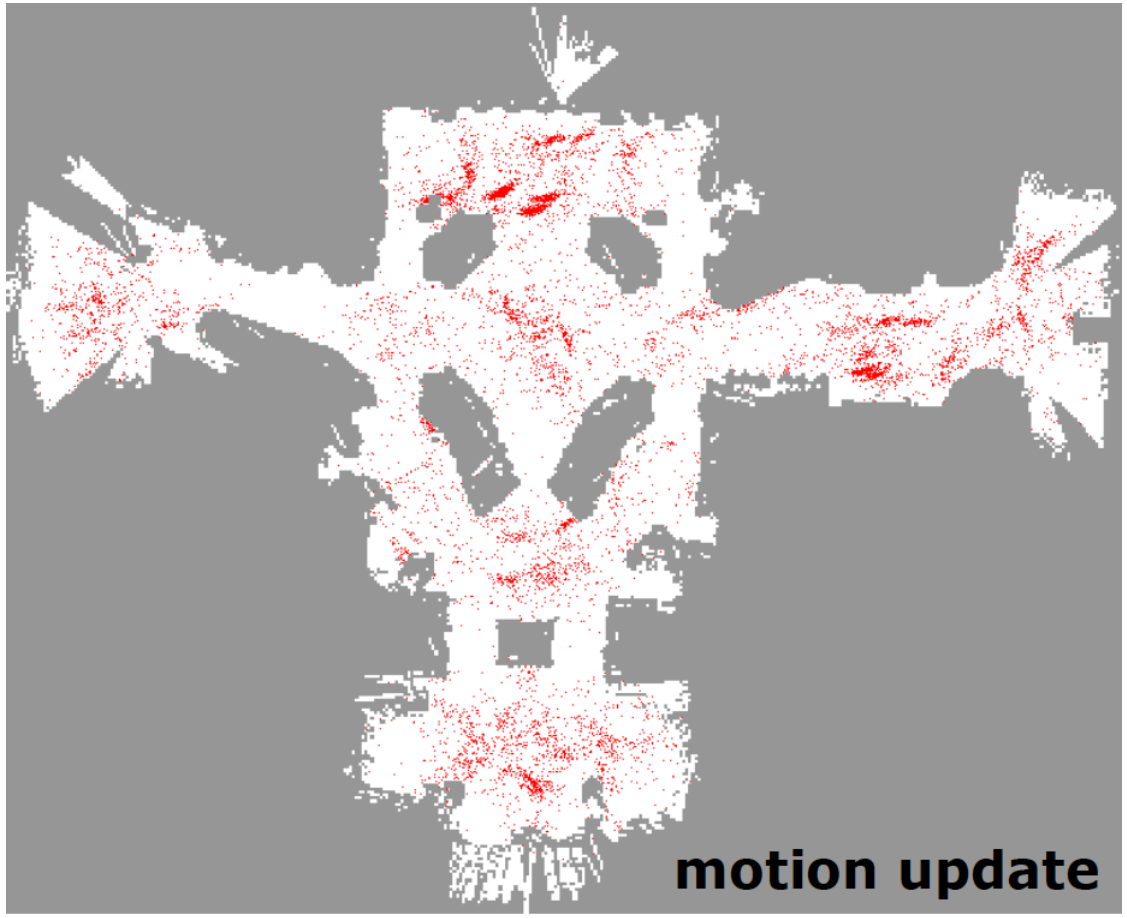
Particle Filter Example



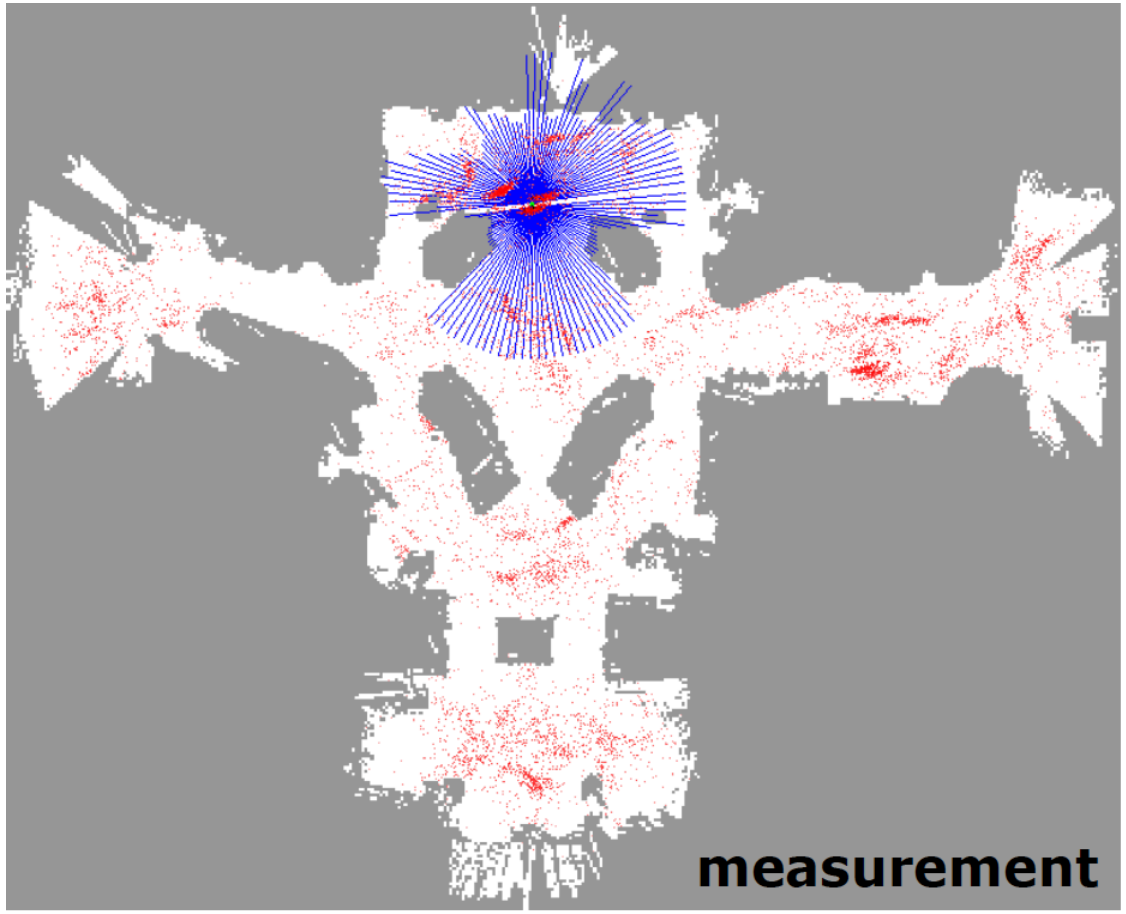
Particle Filter Example



Particle Filter Example

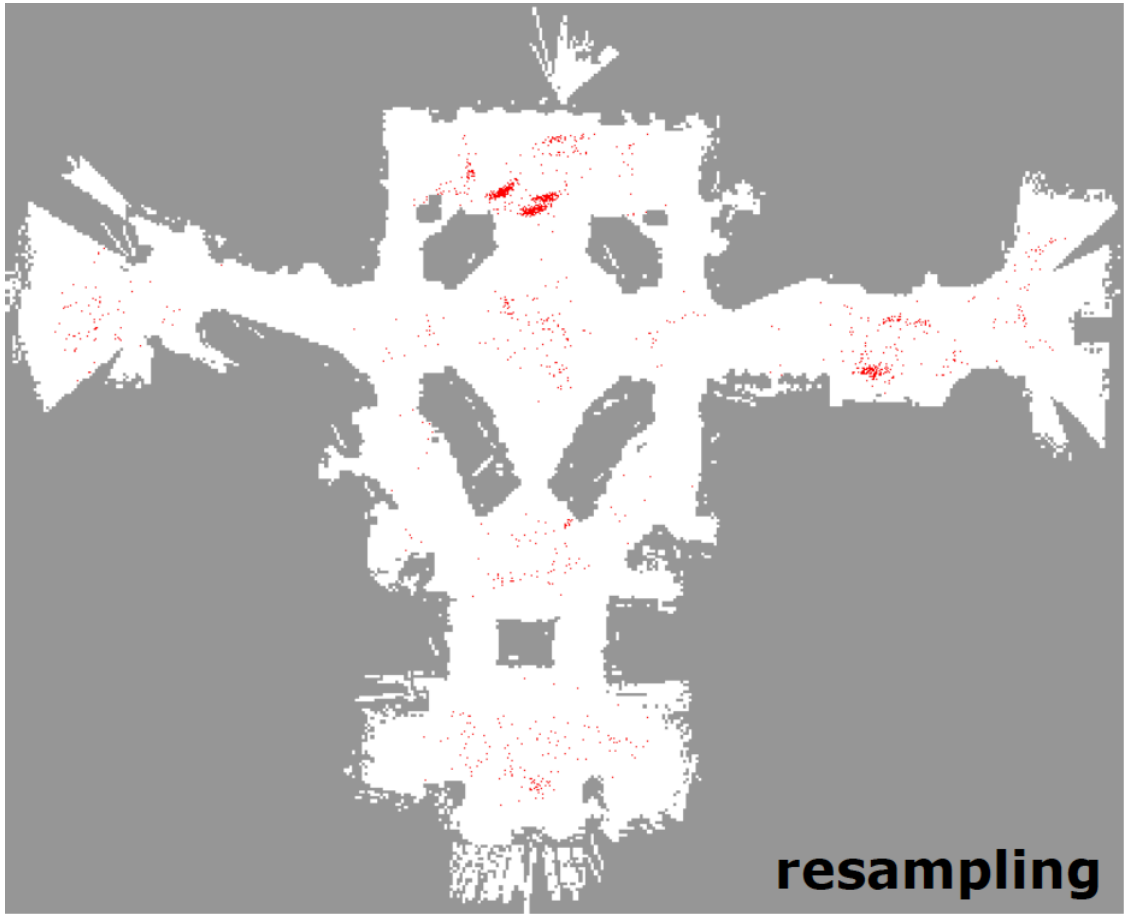


Particle Filter Example

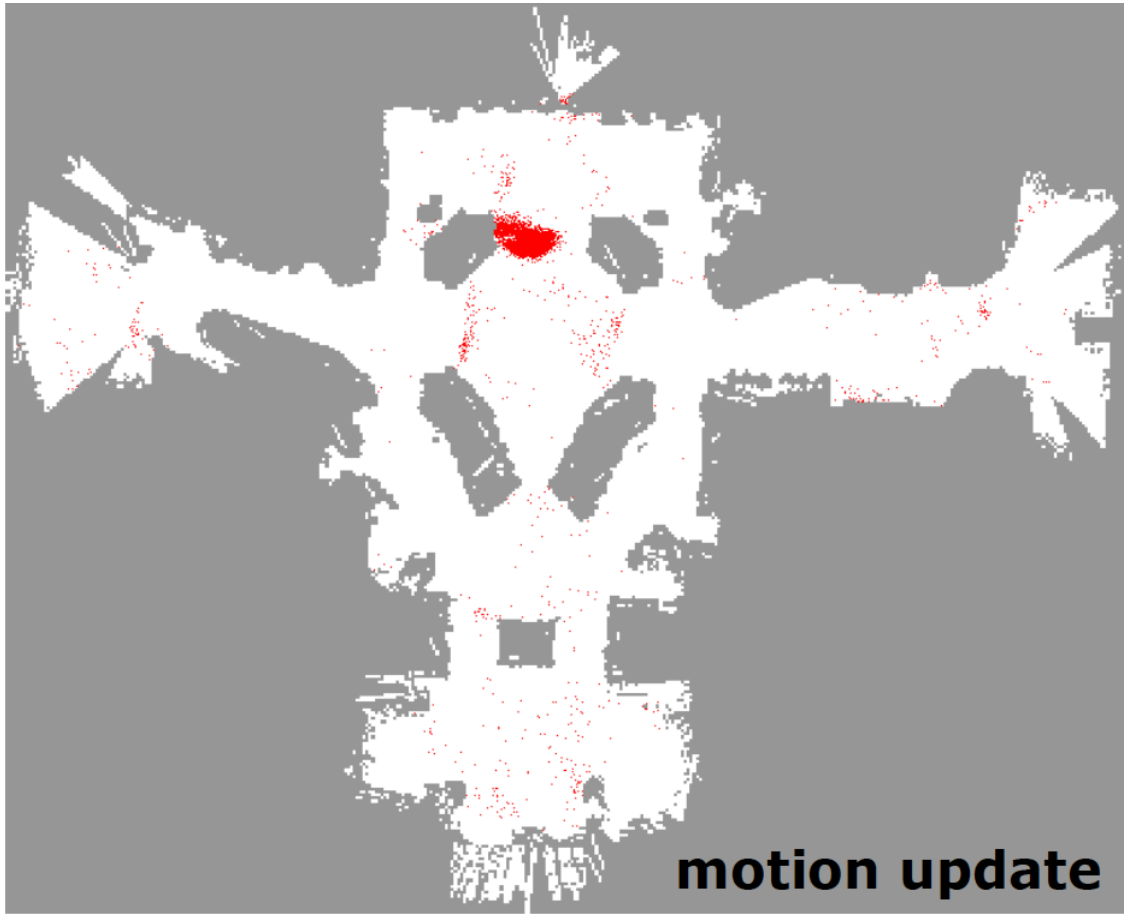


Particle Filter Example

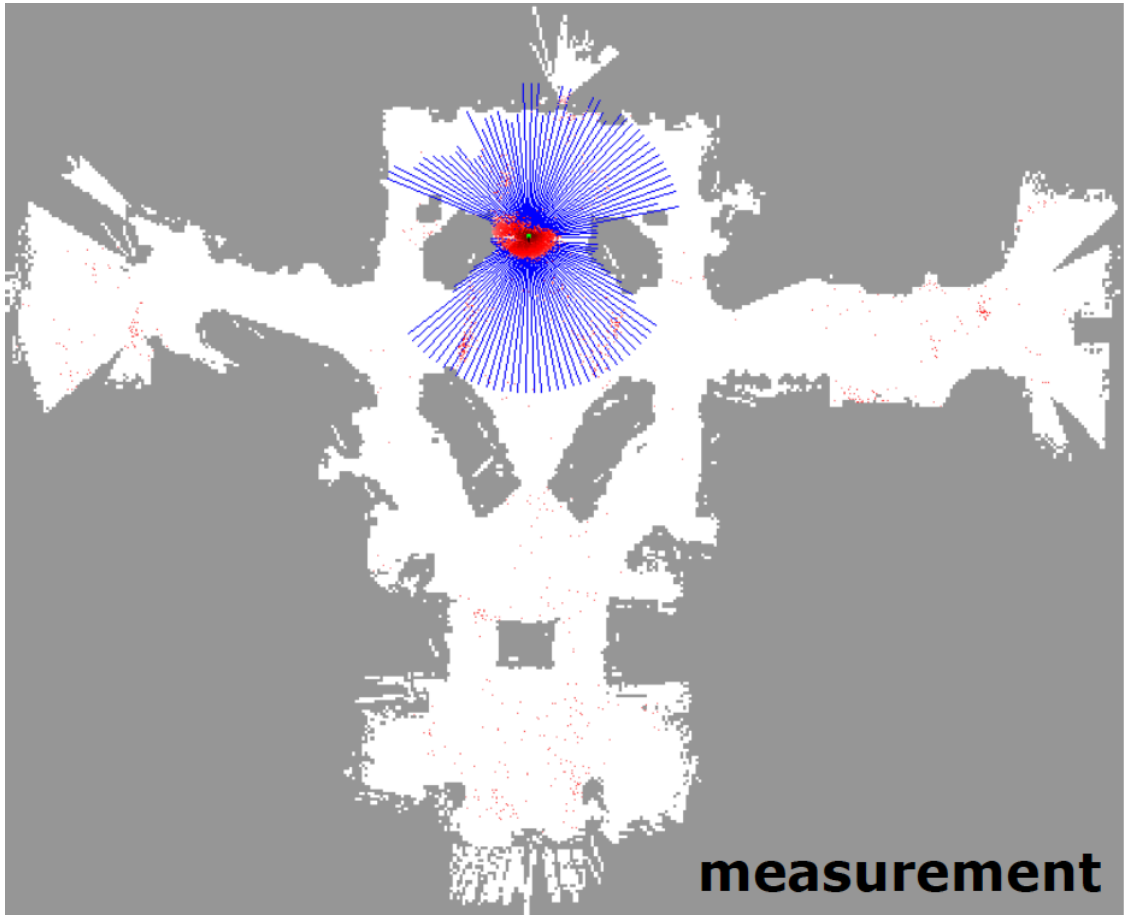
Particle Filter Example



Particle Filter Example

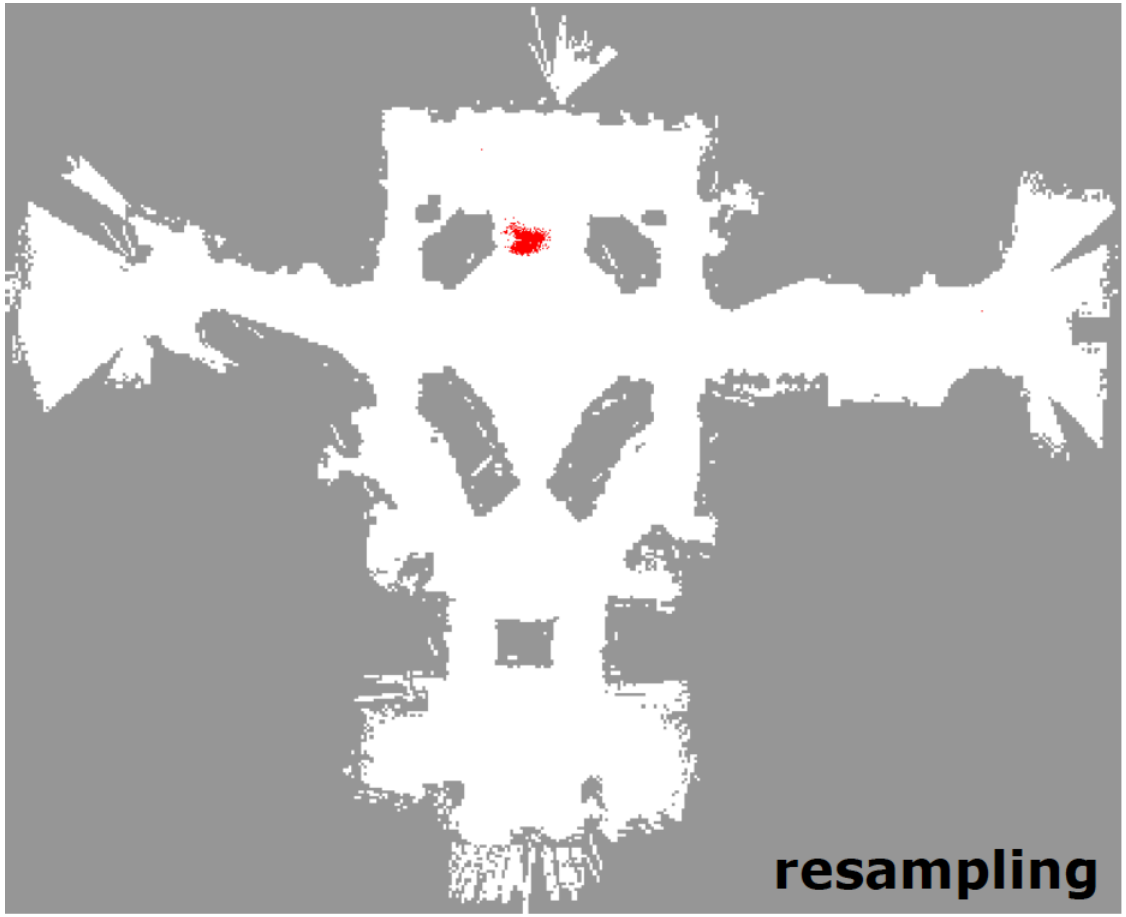


Particle Filter Example

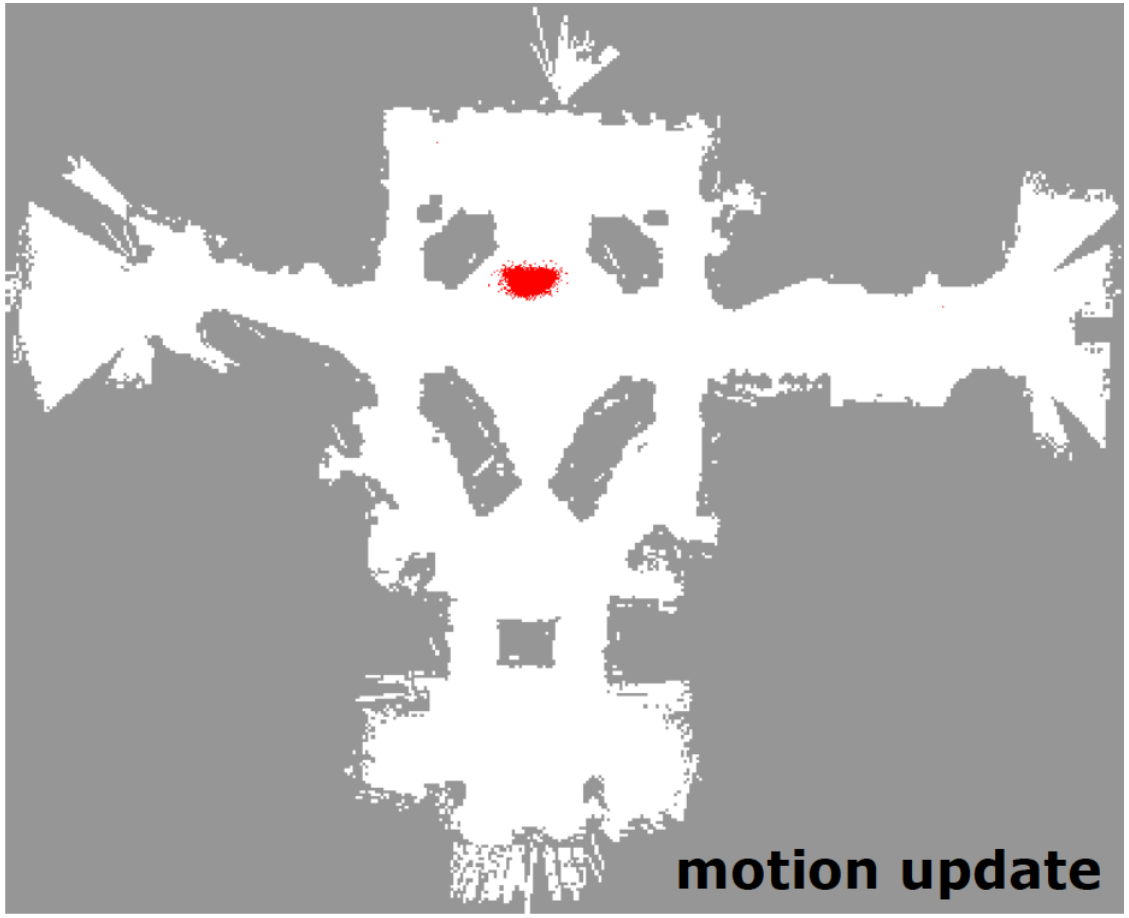


Particle Filter Example

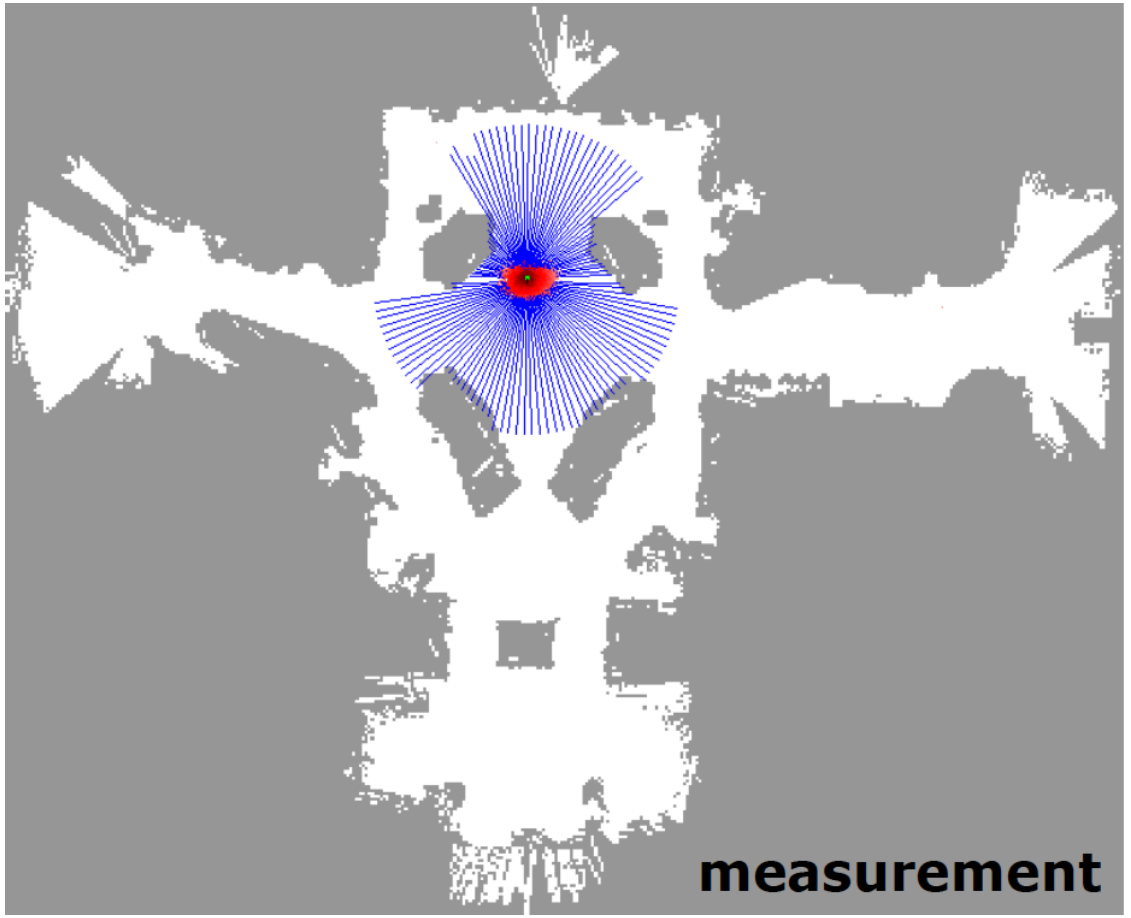
Particle Filter Example



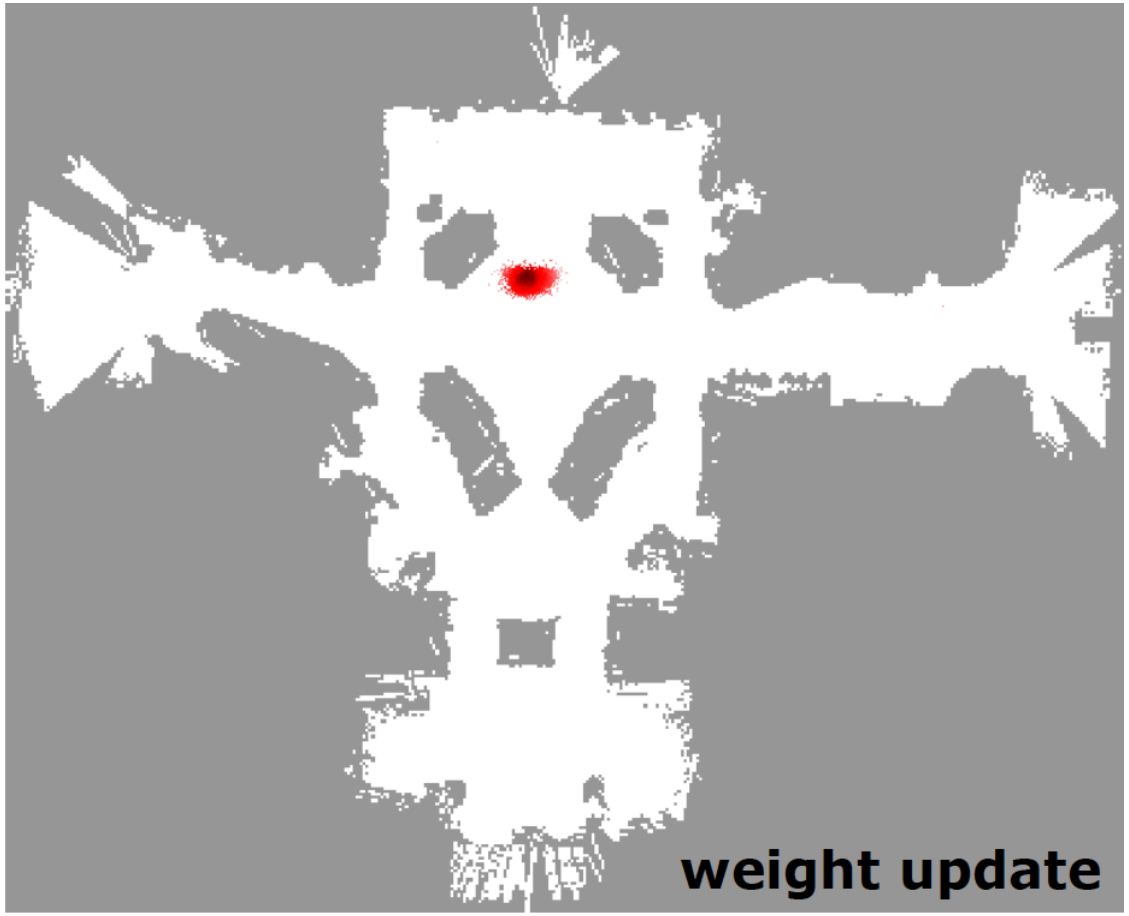
Particle Filter Example



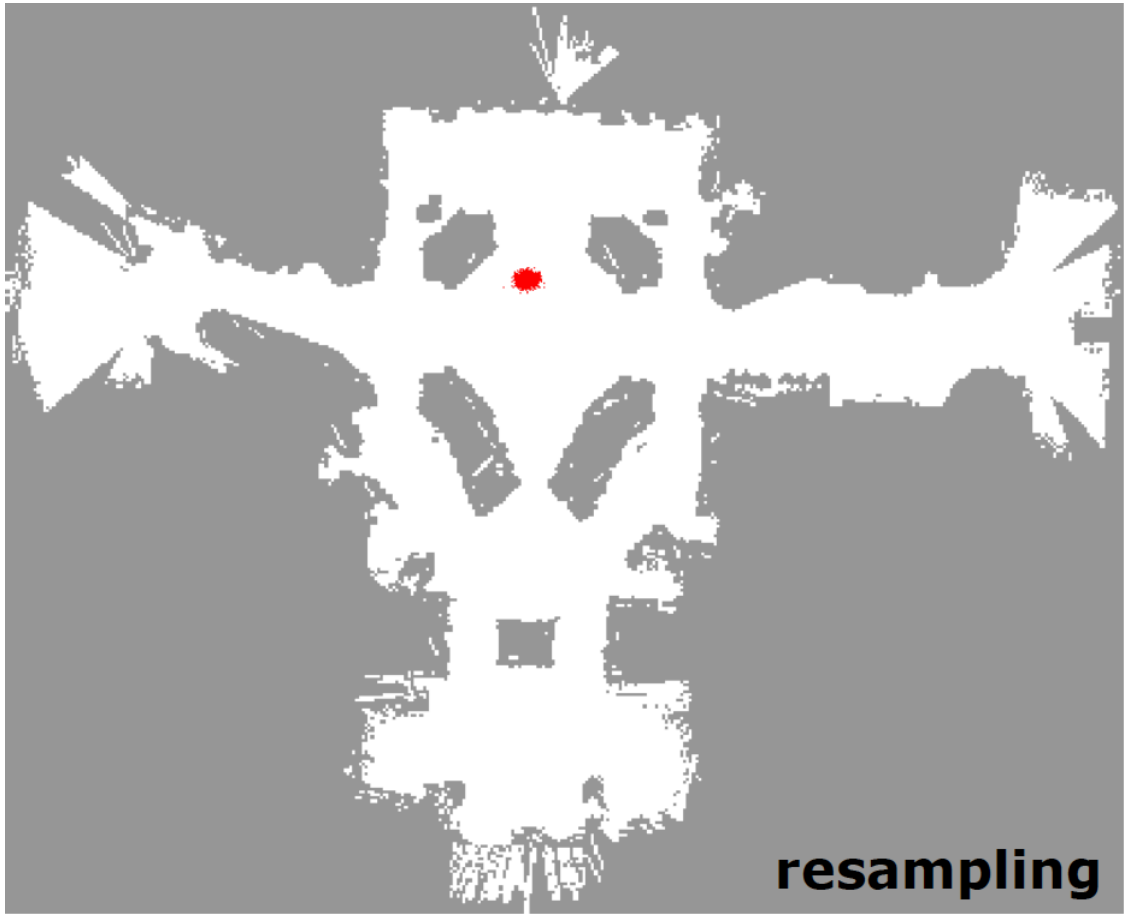
Particle Filter Example



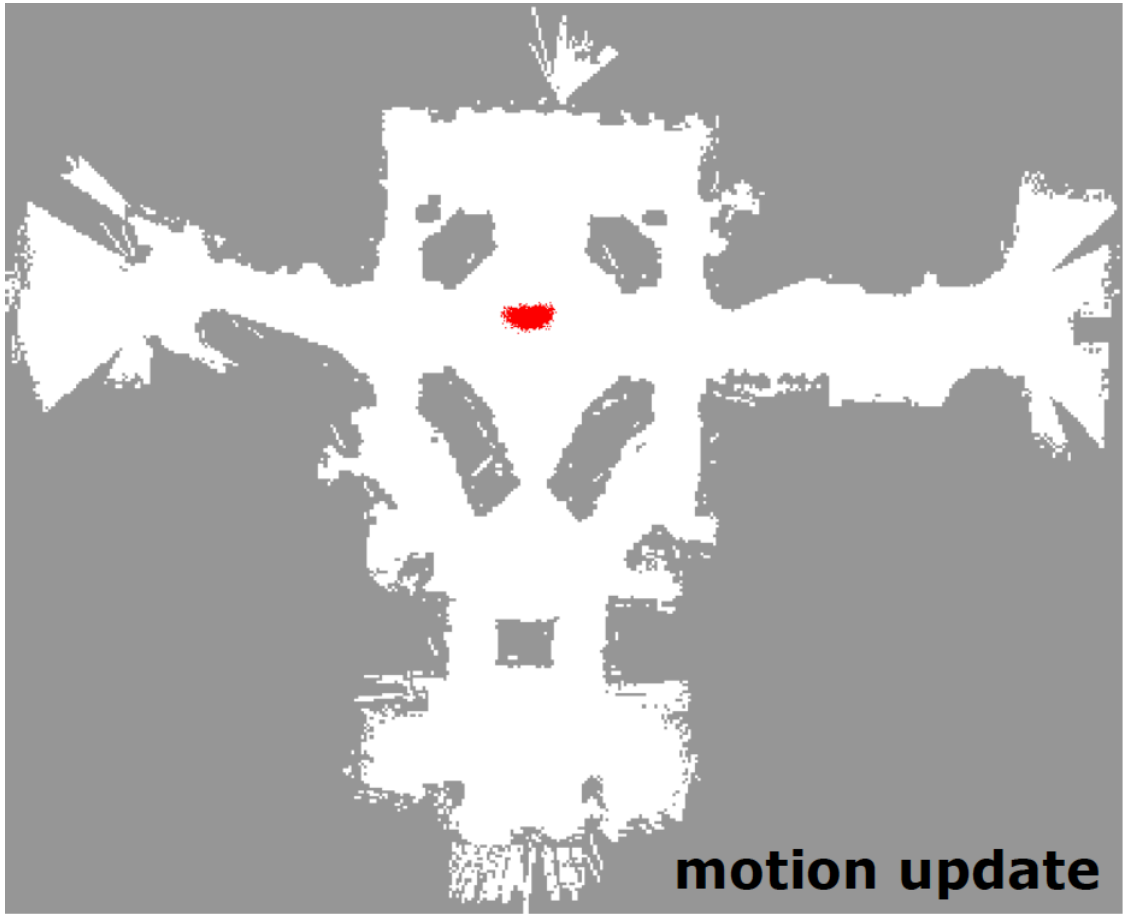
Particle Filter Example



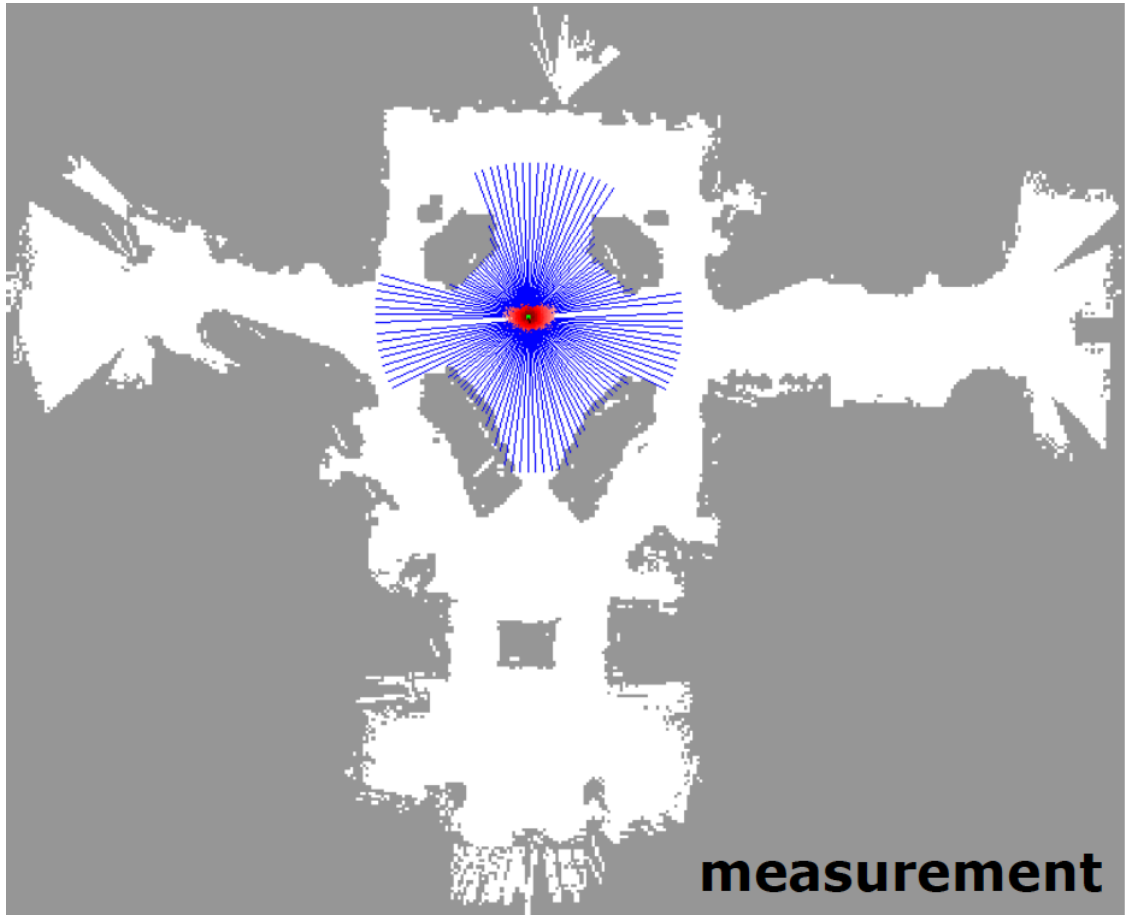
Particle Filter Example



Particle Filter Example



Particle Filter Example



Particle set

- Set of weighted samples

$$\mathcal{X} = \left\{ \left\langle x^{[j]}, w^{[j]} \right\rangle \right\}_{j=1,\ldots,J}$$

state hypothesis — **importance weight**

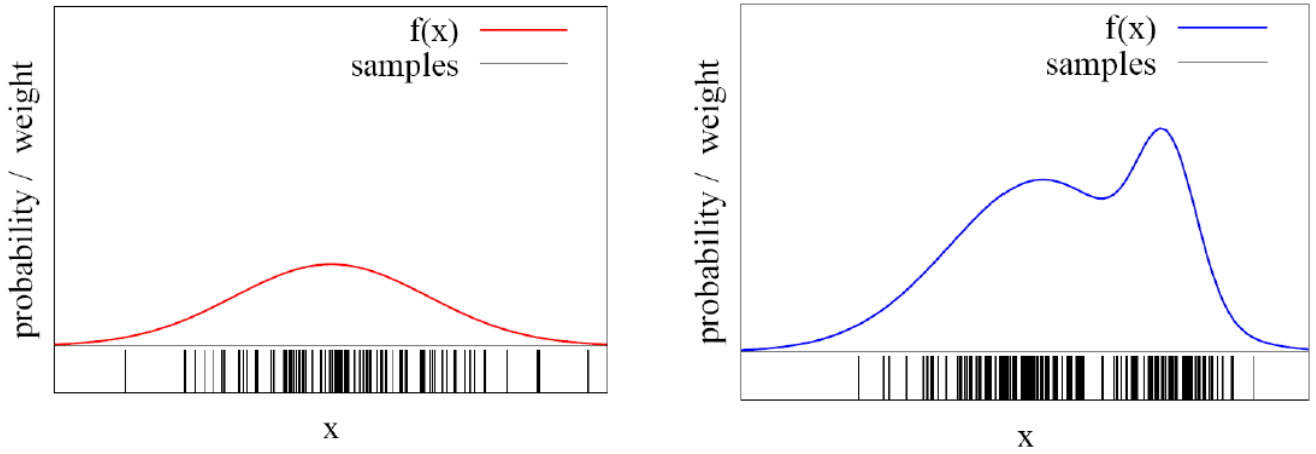
- The samples represent the posterior. Ideally:

$$p(x) = \sum_{j=1}^{J} w^{[j]} \, \delta_{x^{[j]}}(x)$$

with $J \rightarrow$ infinity!

Particles for approximation

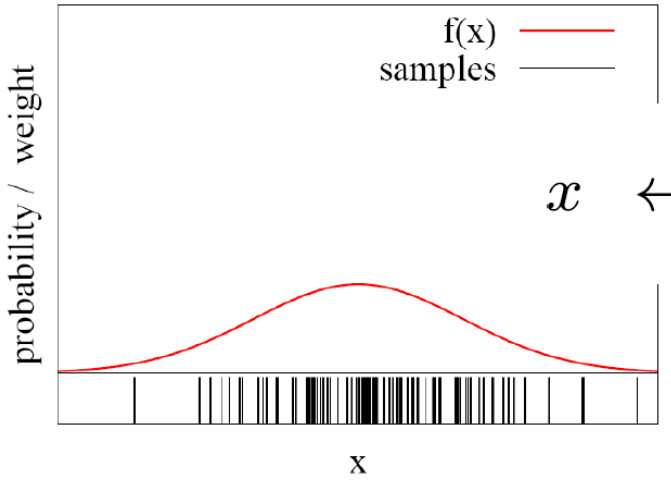
- Particles for function approximation



- The more particles fall into a region, the higher the probability of the region
  - How to obtain such samples?

Closed-form sampling?

- Only possible for a few distributions
- Example: Gaussian



$$x \leftarrow \frac{1}{2}\sum_{i=1}^{12} \text{rand}(-\sigma, \sigma)$$

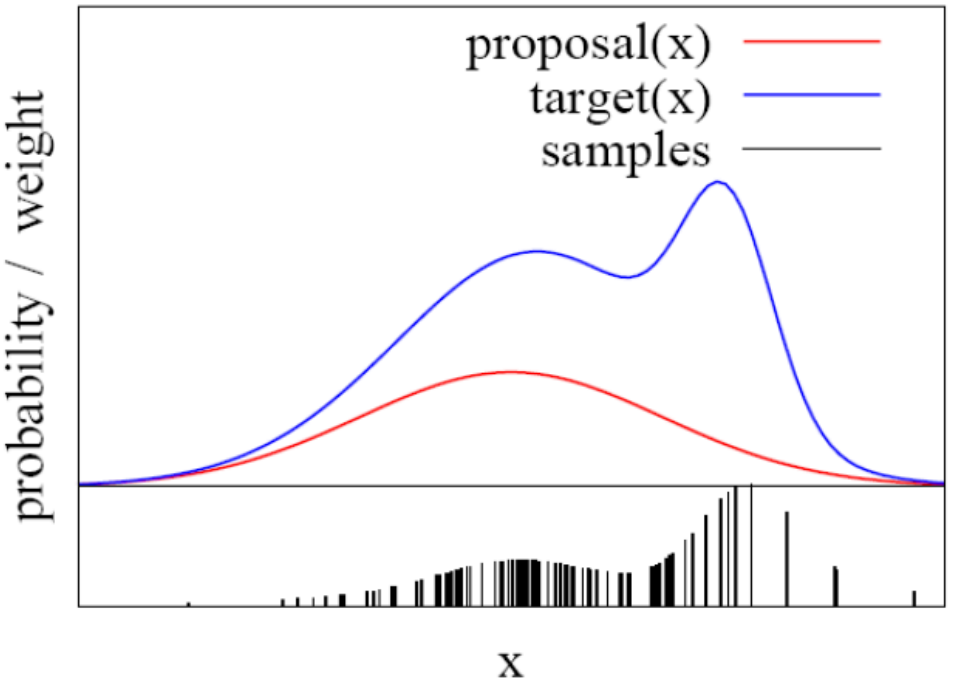
- How to sample from other distributions?

Importance sampling principle

- We can use a different distribution g to generate samples from f
- Account for the "differences between g and f" using a weight

$$w = f / g$$

- target f
- proposal g
- Pre-condition:
  $f(x)>0 \rightarrow g(x)>0$



Particle filter

- Recursive Bayes filter
- Non-parametric approach
- Models the distribution by samples
- Prediction: draw samples from proposal distribution
- Correction: weighting by the ratio of target and proposal

The more samples we use, the better the estimate!

Particle filter algorithm

- Sample the particles using the proposal distribution

$$x_t^{[j]} \sim \pi(x_t \mid \ldots)$$

- Then, compute the importance weights

$$w_t^{[j]} = \frac{target(x_t^{[j]})}{proposal(x_t^{[j]})}$$

- Finally, resampling: Draw sample i with probability $w_t^{[i]}$ and repeat J times
  - Replace unlikely particles with more likely ones

Particle filter algorithm

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

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

2: *for* $j = 1$ *to* J *do*

3: *sample* $x_t^{[j]} \sim \pi(x_t)$

4: $w_t^{[j]} = \frac{p(x_t^{[j]})}{\pi(x_t^{[j]})}$

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

6: *endfor*

7: *for* $j = 1$ *to* J *do*

8: *draw* $i \in 1, \ldots, J$ *with probability* $\propto w_t^{[i]}$

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

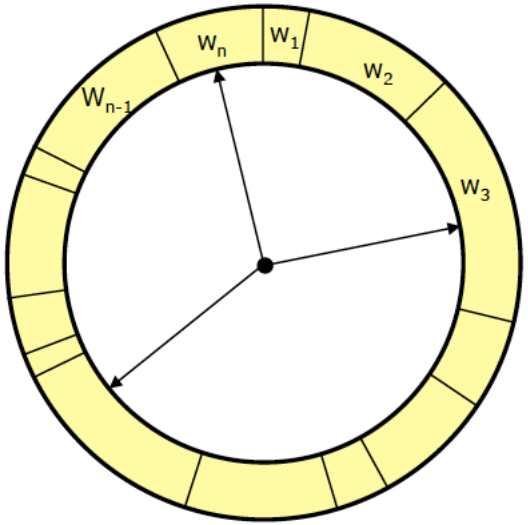
10: *endfor*

11: *return* $\mathcal{X}_t$

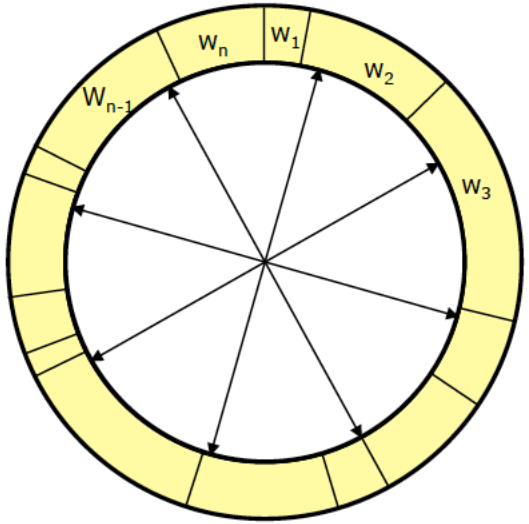
Resampling

- Draw sample i with probability $w_t^{[i]}$
  Repeat J times.
- Informally: “Replace unlikely samples by more likely ones”
- Survival of the fittest
- “Trick” to avoid that many samples cover unlikely states
- Needed as we have a limited number of samples

Resampling



- Roulette wheel
- Binary search
- O(J log J)



- Stochastic universal sampling
- Low variance
- O(J)

Monte-Carlo Localization

- Particle filter based solution to
  - Robot kidnapping problem
  - Global localization
- Each particle is a pose hypothesis
- Proposal is the motion model (move each particle by sample from motion model)

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

- Correction via the observation model

$$w_t^{[j]} = \frac{target}{proposal} \propto p(z_t \mid x_t, m)$$

Particle Filter for Localization

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

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

for $j = 1$ *to* J *do*

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

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

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

endfor

for $j = 1$ *to* J *do*

  draw $i \in 1, \ldots, J$ *with probability* $\propto w_t^{[i]}$

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

endfor

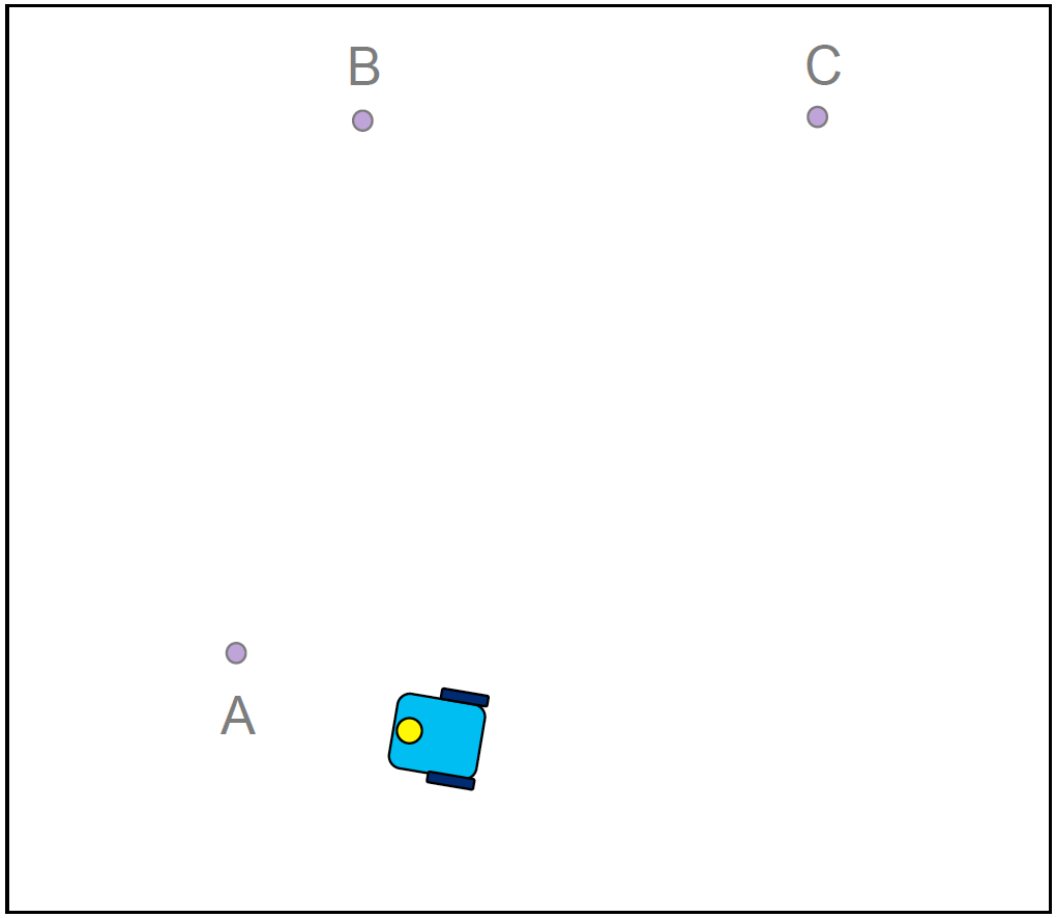
return $\mathcal{X}_t$

Particle filtering for full SLAM

- Besides state x, we also estimate map points m
- State-size is growing as robot explores environment

Particle filtering for full SLAM

- Use internal representations for
  - the positions of landmarks (map)
  - the robot pose parameters
- Assumption:
  - Agent's uncertainty at starting position is zero
- Initialize N particles at the origin, each with weight 1/N

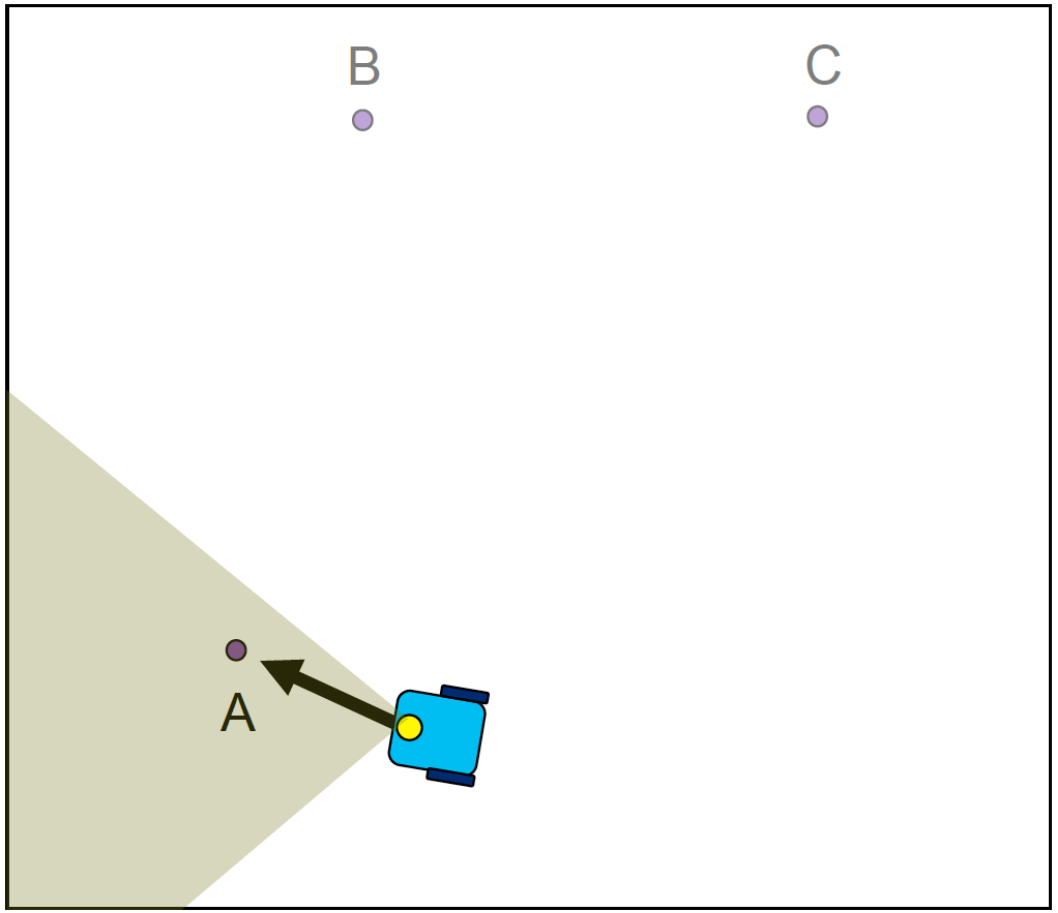


Start: agent has zero uncertainty

Particle filtering full SLAM

On every frame:

- **Predict** how the agent has moved
- **Measure**
- **Update** the internal representation



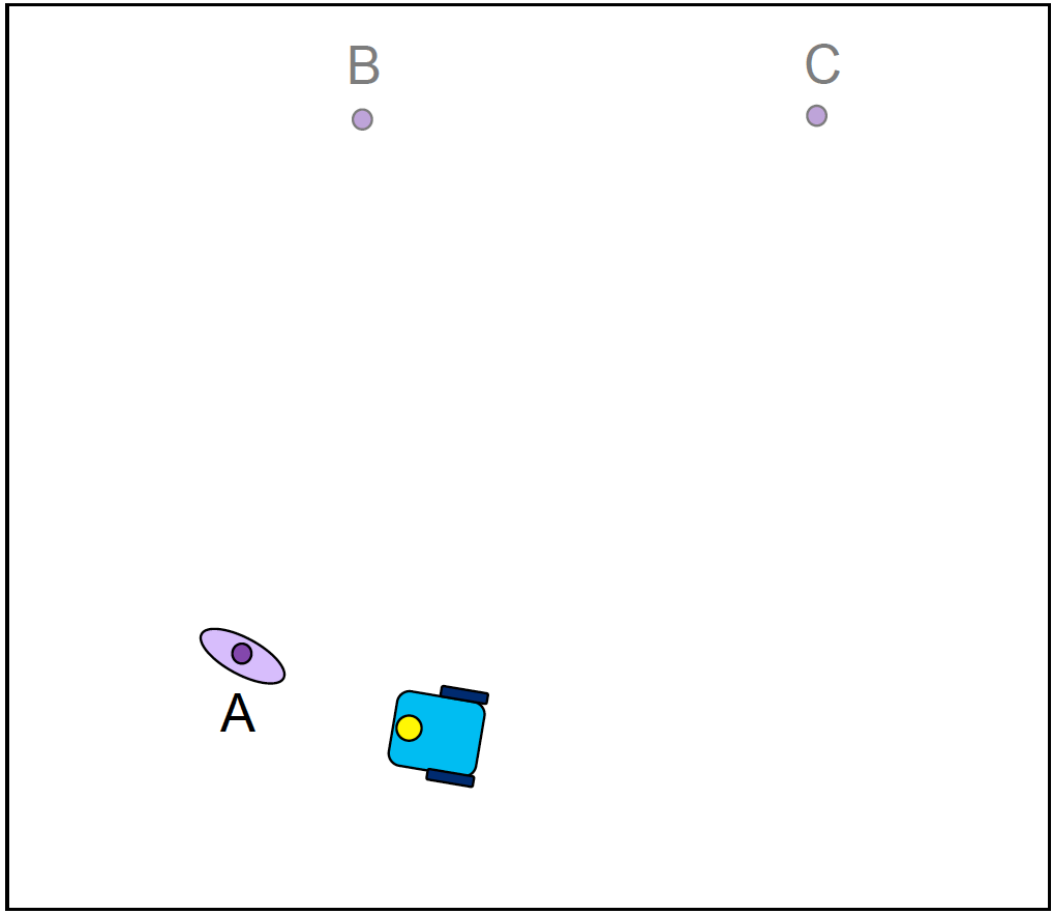
First measurement of feature A

Particle filtering

- The agent observes a feature which is mapped with an uncertainty related to the **measurement model**

  (augment entire state by map points, and—for each original particle—add lots of particles to reflect the uncertainty of the map point)

On every frame:
- **Predict** how the agent has moved
- **Measure**
- **Update** the internal representation

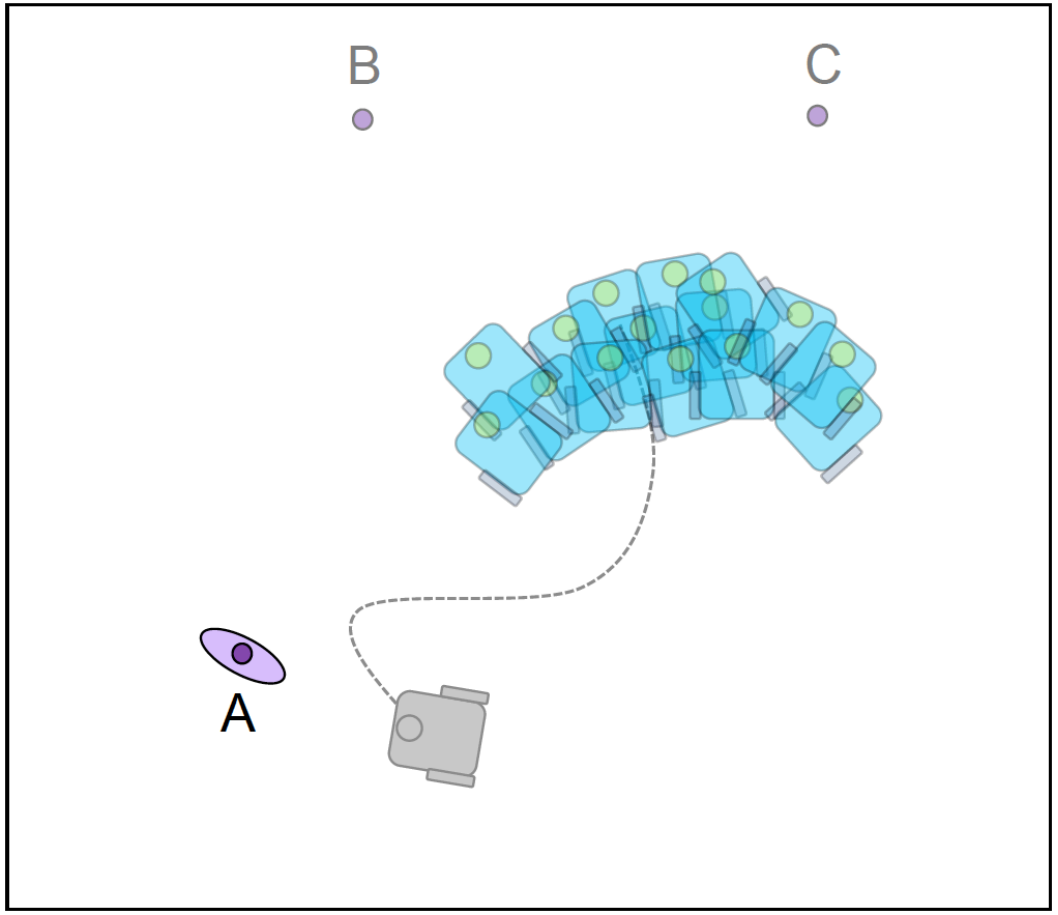


Particle filtering

- As the agent moves, its pose uncertainty increases, obeying the agent's **motion model**

- →Apply different sample from motion model distribution to each particle

On every frame:
- **Predict** how the agent has moved
- **Measure**
- **Update** the internal representation

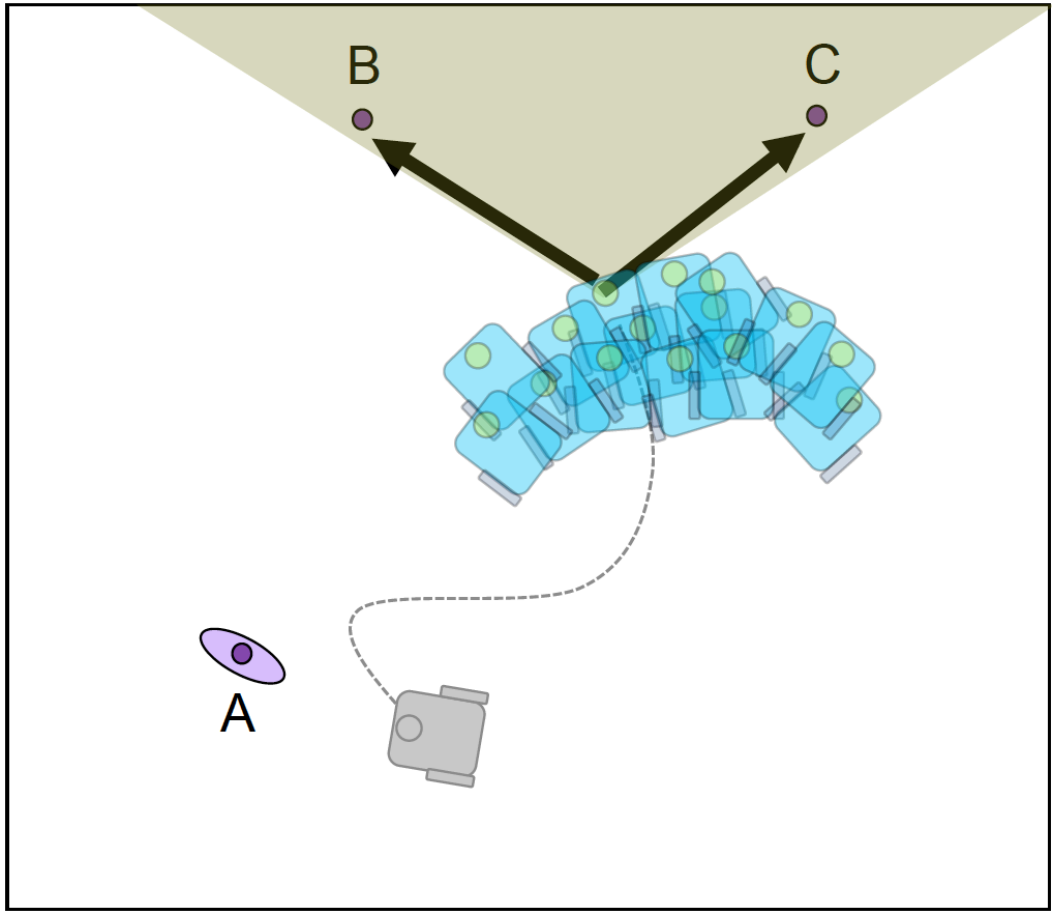


Agent moves forward: uncertainty grows

Particle filtering

- Agent observes two new features

On every frame:
- **Predict** how the agent has moved
- **Measure**
- **Update** the internal representation

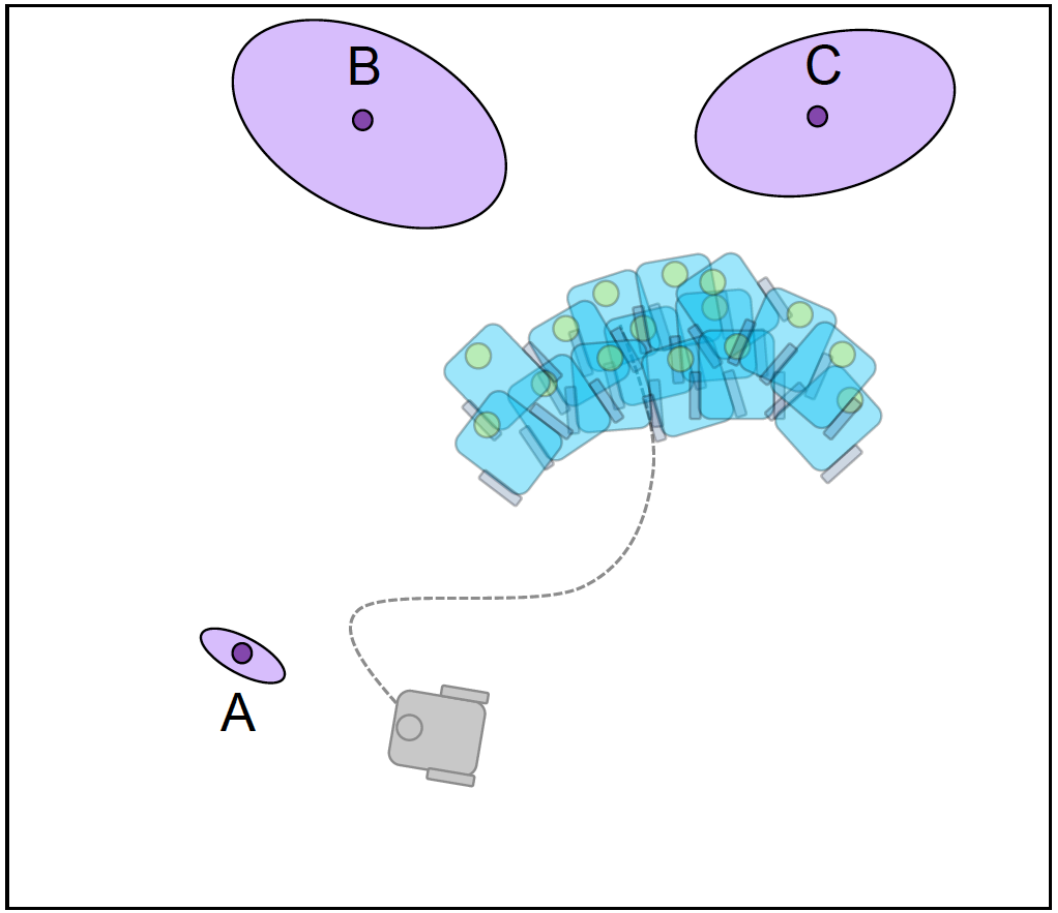


Agent makes first measurement of B and C

Particle filtering

- Their position uncertainty is encoded for each particle individually
  - Their position uncertainty results from the **combination** of the measurement error with the particle positions
    → Map uncertainty becomes **correlated** with the body pose uncertainty

On every frame:
- **Predict** how the agent has moved
- **Measure**
- **Update** the internal representation

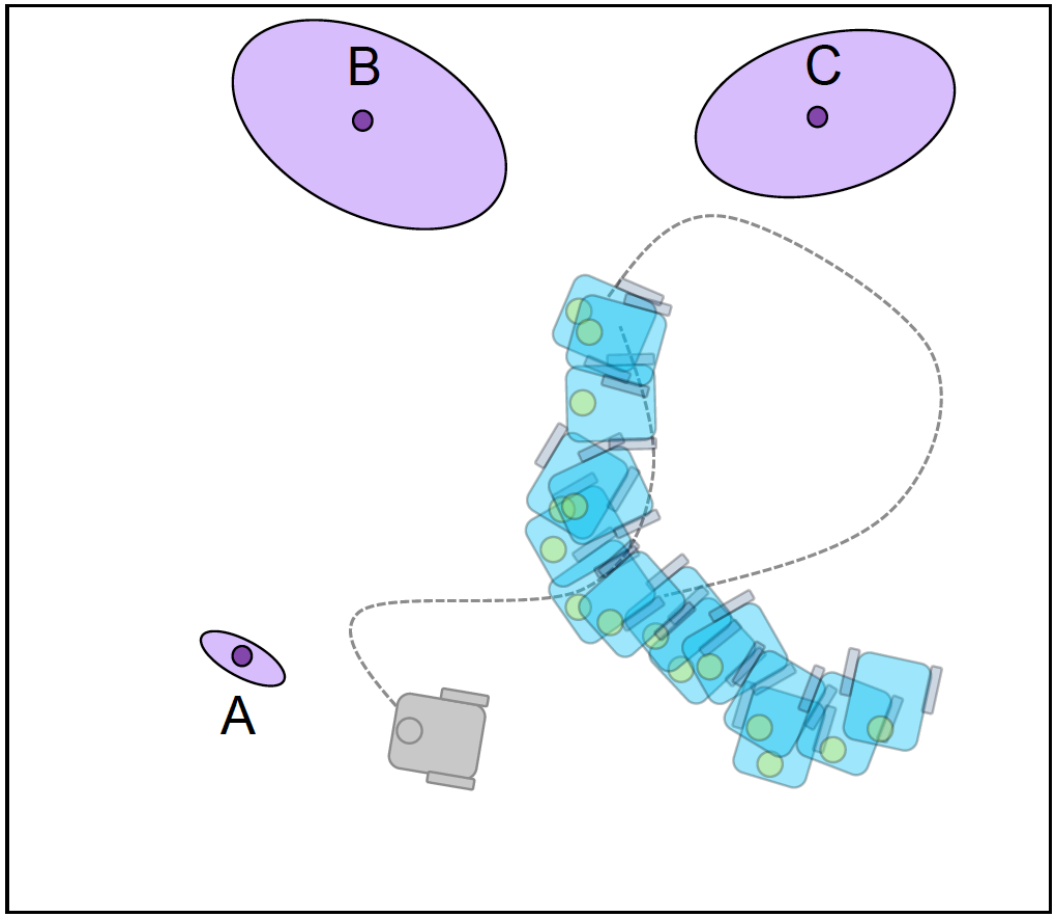


Agent makes first measurement of B and C

Particle filtering

- Agent moves again and its uncertainty increases (motion model)
- Apply motion model to each particle

On every frame:
- **Predict** how the agent has moved
- **Measure**
- **Update** the internal representation

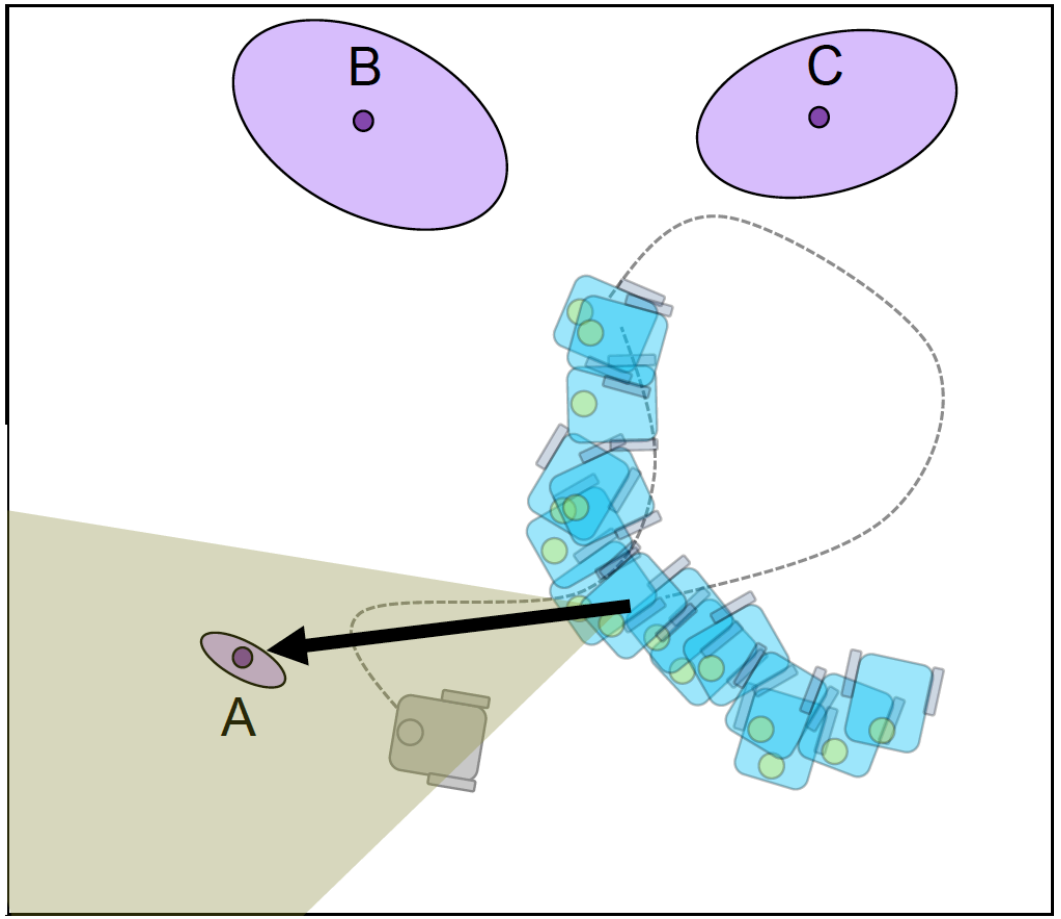


Agent moves again: uncertainty grows more

Particle filtering

- Agent re-observes an old feature
  → **Loop closure** detection

On every frame:
- **Predict** how the agent has moved
- **Measure**
- **Update** the internal representation

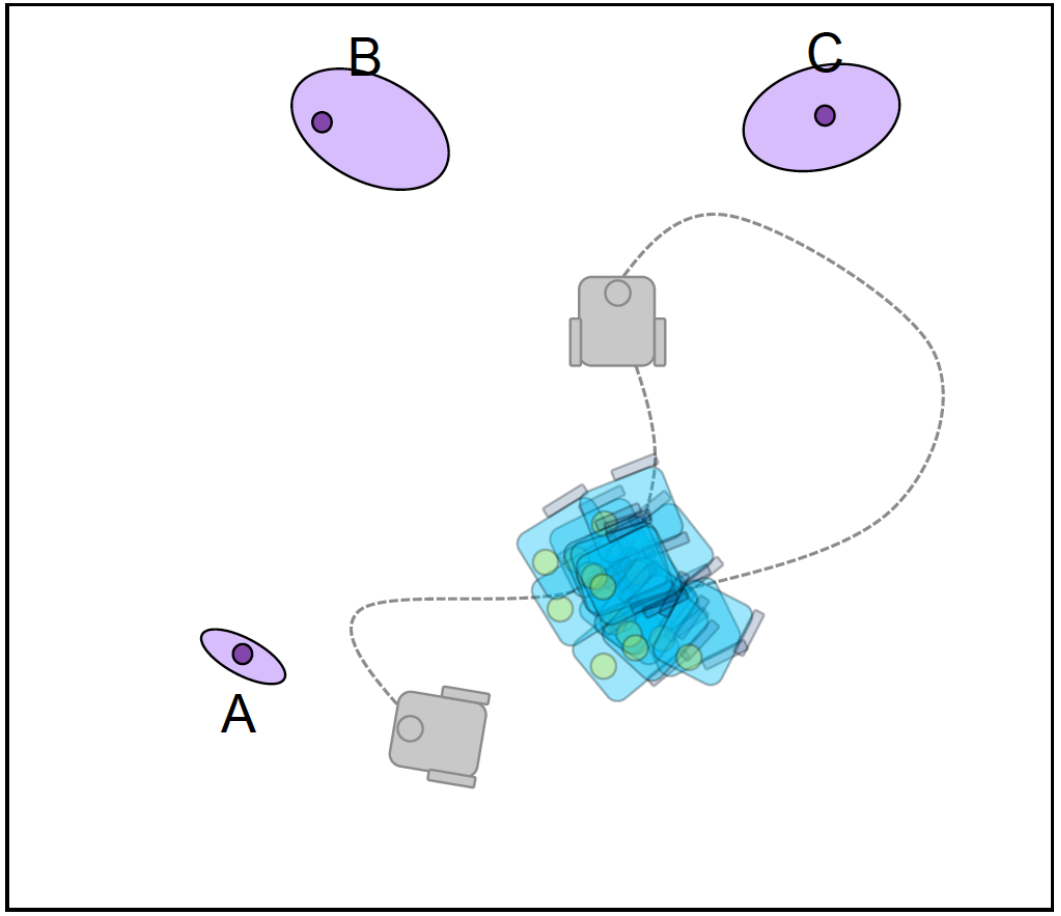


Agent re-measures A: “loop closure”
→Uncertainty shrinks

Particle filtering

- For each particle:
  - Compare particles' predicted measurements with obtained ones
  - Re-weigh such that particles with good predictions get higher weight & re-normalize particle weights
  - Re-sample according to likelihood

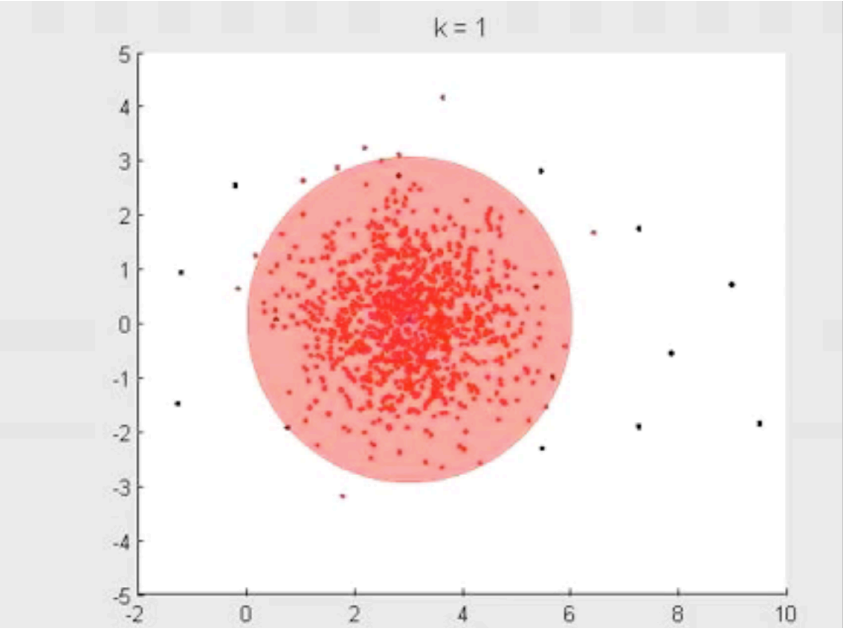
On every frame:
- **Predict** how the agent has moved
- **Measure**
- **Update** the internal representation



Agent re-measures A: "loop closure"
→Uncertainty shrinks

Particle filtering

- Particle Filtering
  - Represents belief by a series of samples
  - Each Particle = a hypothesis of the state with a associated weight (all weights should add up to 1)
  - Follow the “predict/measure/update” approach
- Pros
  - Noise densities can be from any distribution
  - Works for multi-modal distributions
  - Easy to implement
- Cons
  - Does not scale to high-dimensional problems
  - Requires many particles to have good convergence



Distribution in the agent’s position estimate:
- red dots – particle filtering
- red ellipse – EKF filtering

Summary: Particle filters

- Particle filters are non-parametric, recursive Bayes filters
- Posterior is represented by a set of weighted samples
- Proposal to draw the samples for t+1
- Weight to account for the differences between the proposal and the target
- Work well in low-dimensional spaces

Summary: PF Localization

- Particles are propagated according to the motion model
- They are weighted according to the likelihood of the observation
- Called: Monte-Carlo localization (MCL)
- MCL is the gold standard for mobile robot localization
- Good for global localization, bad for SLAM (curse of dimensionality)