

Introduction to Mobile Robotics

Summary

Wolfram Burgard, Maren Bennewitz, Diego
Tipaldi, Luciano Spinello



Probabilistic Robotics

Probabilistic Robotics

Key idea: Explicit representation of uncertainty

(using the calculus of probability theory)

- Perception = state estimation
- Action = utility optimization

Bayes Formula

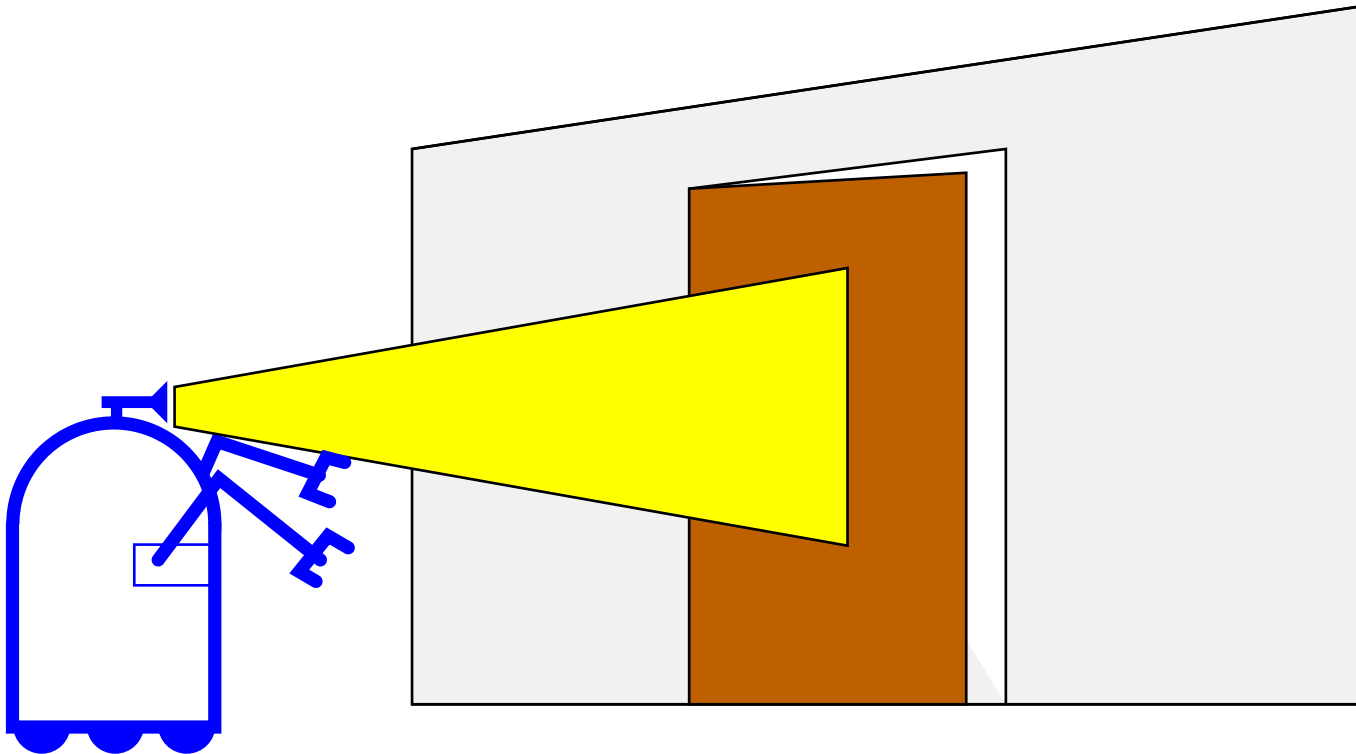
$$P(x, y) = P(x | y)P(y) = P(y | x)P(x)$$

\Rightarrow

$$P(x | y) = \frac{P(y | x) P(x)}{P(y)} = \frac{\text{likelihood} \cdot \text{prior}}{\text{evidence}}$$

Simple Example of State Estimation

- Suppose a robot obtains measurement z
- What is $P(open|z)$?



Causal vs. Diagnostic Reasoning

- $P(open|z)$ is **diagnostic**.
- $P(z|open)$ is **causal**.
- Often **causal** knowledge is easier to obtain.
- Bayes rule allows us to use causal knowledge:

count frequencies!

$$P(open|z) = \frac{P(z|open)P(open)}{P(z)}$$

z = observation
 u = action
 x = state

Bayes Filters

$$\boxed{Bel(x_t)} = P(x_t | u_1, z_1, \dots, u_t, z_t)$$

Bayes $= \eta P(z_t | x_t, u_1, z_1, \dots, u_t) P(x_t | u_1, z_1, \dots, u_t)$

Markov $= \eta P(z_t | x_t) P(x_t | u_1, z_1, \dots, u_t)$

Total prob. $= \eta P(z_t | x_t) \int P(x_t | u_1, z_1, \dots, u_t, x_{t-1})$
 $P(x_{t-1} | u_1, z_1, \dots, u_t) dx_{t-1}$

Markov $= \eta P(z_t | x_t) \int P(x_t | u_t, x_{t-1}) P(x_{t-1} | u_1, z_1, \dots, u_t) dx_{t-1}$

Markov $= \eta P(z_t | x_t) \int P(x_t | u_t, x_{t-1}) P(x_{t-1} | u_1, z_1, \dots, z_{t-1}) dx_{t-1}$

$$\boxed{= \eta P(z_t | x_t) \int P(x_t | u_t, x_{t-1}) Bel(x_{t-1}) dx_{t-1}}$$

Bayes Filters are Familiar!

$$Bel(x_t) = \eta P(z_t | x_t) \int P(x_t | u_t, x_{t-1}) Bel(x_{t-1}) dx_{t-1}$$

- Kalman filters
- Particle filters
- Hidden Markov models
- Dynamic Bayesian networks
- ...

Sensor and Motion Models

$$P(z | x, m)$$

$$P(x | x', u)$$

Motion Models

- Robot motion is inherently uncertain.
- How can we model this uncertainty?



Probabilistic Motion Models

- To implement the Bayes Filter, we need the transition model $p(x | x', u)$.
- The term $p(x | x', u)$ specifies a posterior probability, that action u carries the robot from x' to x .

Typical Motion Models

- In practice, one often finds two types of motion models:
 - **Odometry-based**
 - **Velocity-based (dead reckoning)**
- Odometry-based models are used when systems are equipped with wheel encoders.
- Velocity-based models have to be applied when no wheel encoders are given.
- They calculate the new pose based on the velocities and the time elapsed.

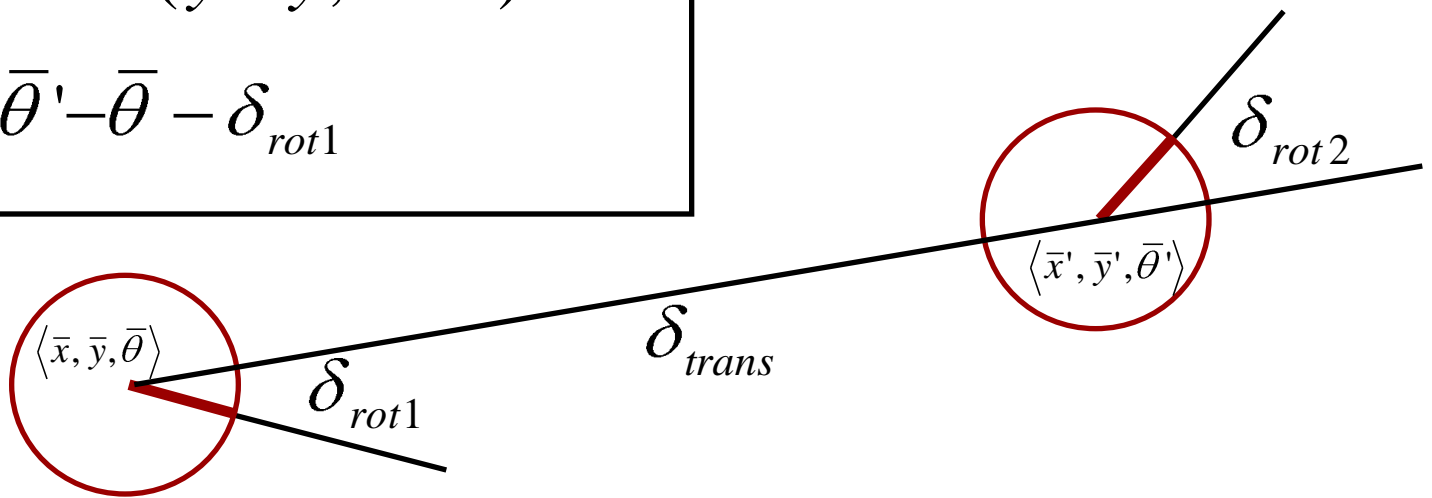
Odometry Model

- Robot moves from $\langle \bar{x}, \bar{y}, \bar{\theta} \rangle$ to $\langle \bar{x}', \bar{y}', \bar{\theta}' \rangle$.
- Odometry information $u = \langle \delta_{rot1}, \delta_{rot2}, \delta_{trans} \rangle$.

$$\delta_{trans} = \sqrt{(\bar{x}' - \bar{x})^2 + (\bar{y}' - \bar{y})^2}$$

$$\delta_{rot1} = \text{atan2}(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta}$$

$$\delta_{rot2} = \bar{\theta}' - \bar{\theta} - \delta_{rot1}$$



Sensors for Mobile Robots

- **Contact sensors:** Bumpers
- **Internal sensors**
 - Accelerometers (spring-mounted masses)
 - Gyroscopes (spinning mass, laser light)
 - Compasses, inclinometers (earth magnetic field, gravity)
- **Proximity sensors**
 - Sonar (time of flight)
 - Radar (phase and frequency)
 - Laser range-finders (triangulation, tof, phase)
 - Infrared (intensity)
- **Visual sensors:** Cameras
- **Satellite-based sensors:** GPS

Beam-based Sensor Model

- Scan z consists of K measurements.

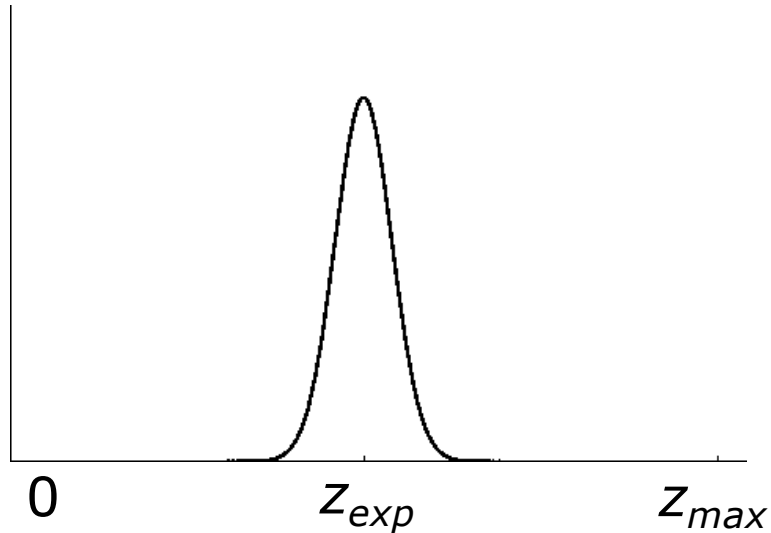
$$z = \{z_1, z_2, \dots, z_K\}$$

- Individual measurements are independent given the robot position.

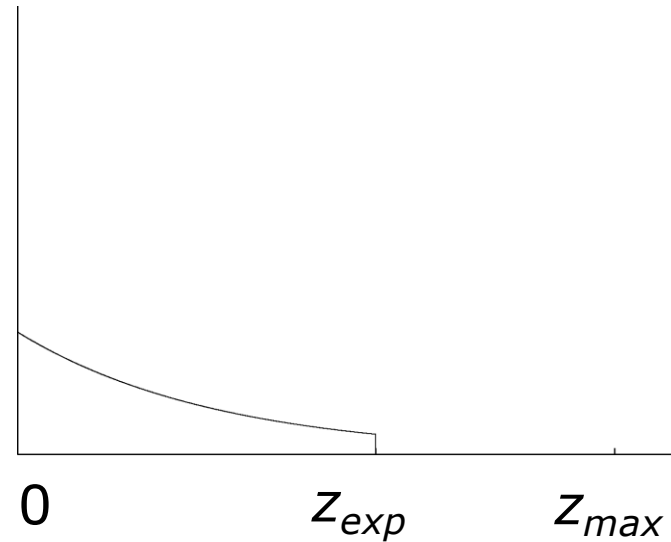
$$P(z | x, m) = \prod_{k=1}^K P(z_k | x, m)$$

Beam-based Proximity Model

Measurement noise



Unexpected obstacles

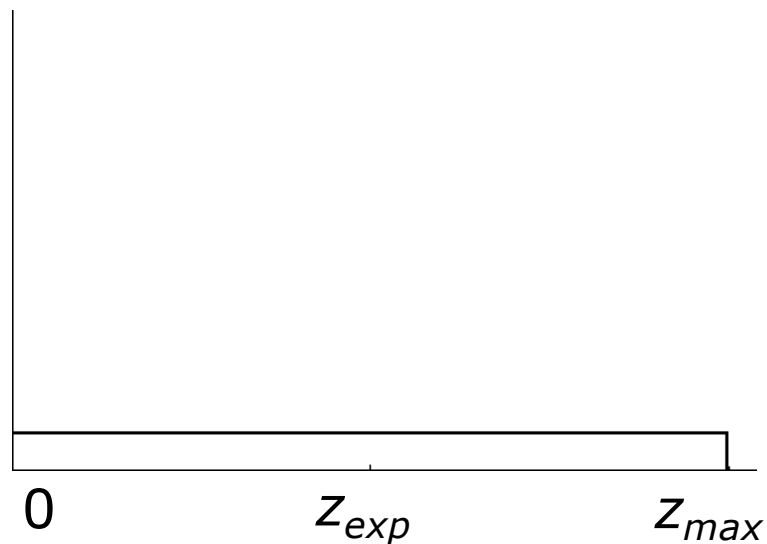


$$P_{hit}(z | x, m) = \eta \frac{1}{\sqrt{2\pi b}} e^{-\frac{1}{2} \frac{(z - z_{exp})^2}{b}}$$

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

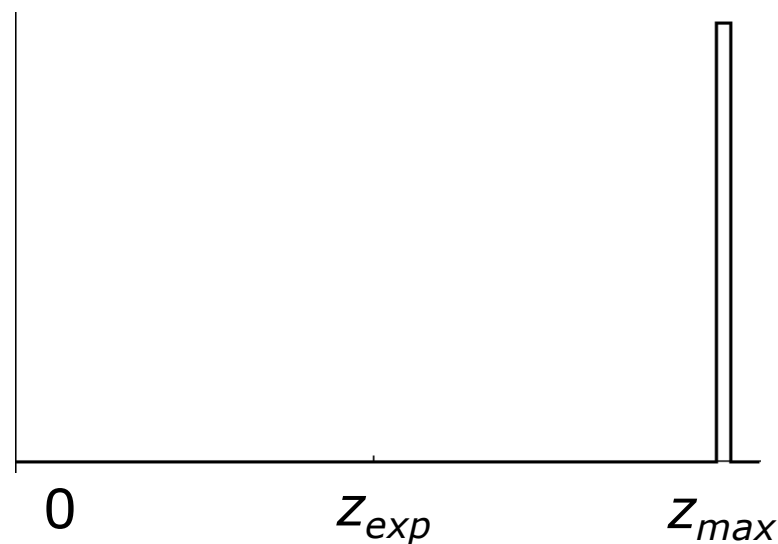
Beam-based Proximity Model

Random measurement



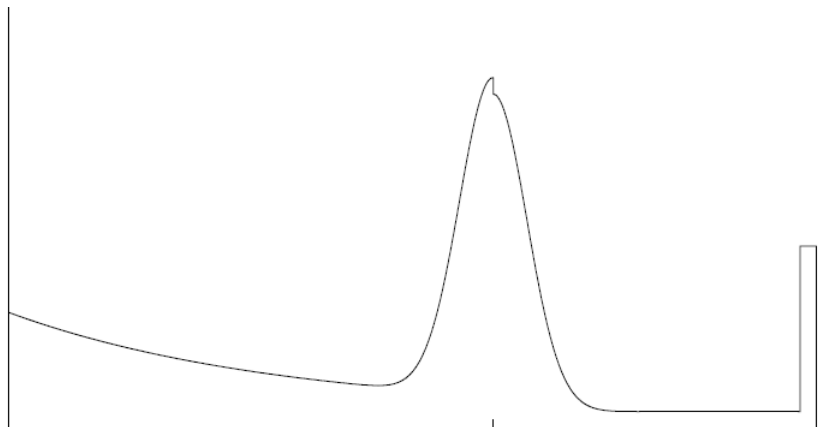
$$P_{rand}(z | x, m) = \eta \frac{1}{z_{max}}$$

Max range



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

Resulting Mixture Density



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

How can we determine the model parameters?

Bayes Filter in Robotics

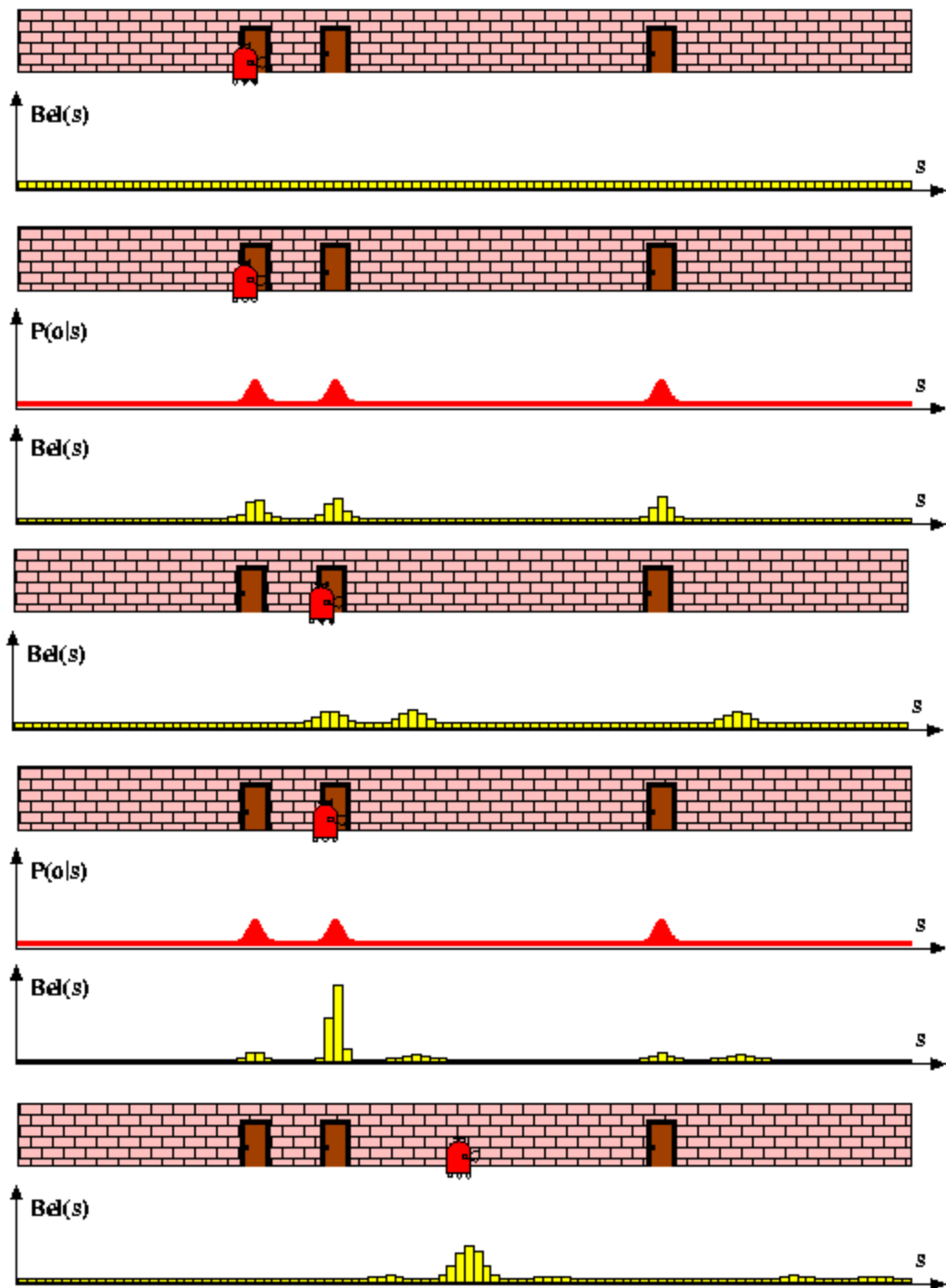
Bayes Filters in Action

- Discrete filters
- Kalman filters
- Particle filters

Discrete Filter

- The belief is typically stored in a histogram / grid representation
- To update the belief upon sensory input and to carry out the normalization one has to iterate over all cells of the grid

Piecewise Constant



Kalman Filter

- Optimal for linear Gaussian systems!
- Most robotics systems are **nonlinear**!
- Polynomial in measurement dimensionality k and state dimensionality n :

$$O(k^{2.376} + n^2)$$

Kalman Filter Algorithm

1. Algorithm **Kalman_filter**(μ_{t-1} , Σ_{t-1} , u_t , z_t):
2. Prediction:
3. $\bar{m}_t = A_t m_{t-1} + B_t u_t$
4. $\bar{S}_t = A_t S_{t-1} A_t^T + Q_t$
5. Correction:
6. $K_t = \bar{S}_t C_t^T (C_t \bar{S}_t C_t^T + R_t)^{-1}$
7. $m_t = \bar{m}_t + K_t (z_t - C_t \bar{m}_t)$
8. $S_t = (I - K_t C_t) \bar{S}_t$
9. Return μ_t , Σ_t

Extended Kalman Filter

- Approach to handle non-linear models
- Performs a linearization in each step
- **Not optimal**
- Can **diverge** if nonlinearities are large!
- Works surprisingly well even when all assumptions are violated!
- Same complexity than the KF

Particle Filter

- Basic principle
 - Set of state hypotheses (“particles”)
 - Survival-of-the-fittest
- Particle filters are a way to efficiently represent non-Gaussian distribution

Mathematical Description

- Set of weighted samples

$$S = \left\{ \left\langle s^{[i]}, w^{[i]} \right\rangle \mid i = 1, \dots, N \right\}$$

State hypothesis Importance weight

- The samples represent the posterior

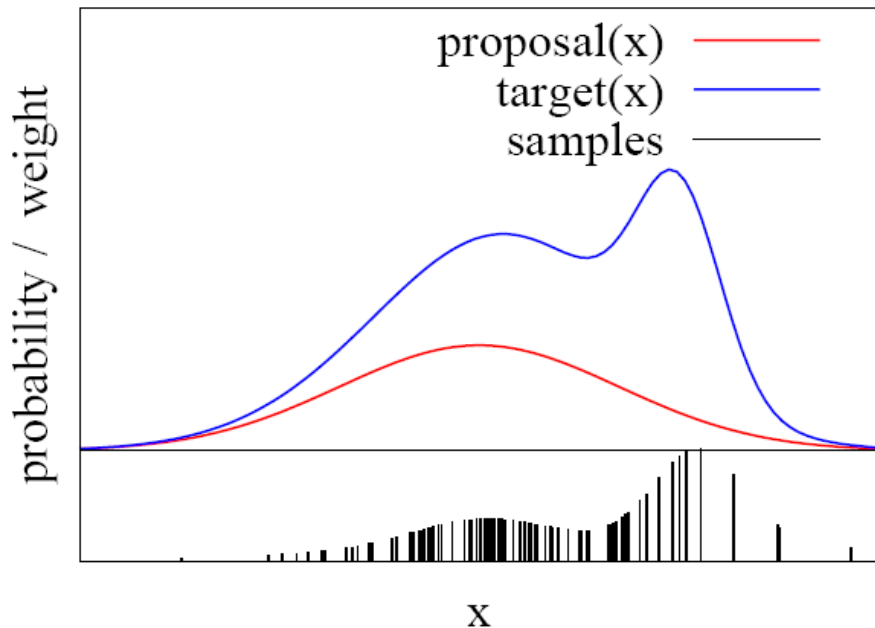
$$p(x) = \sum_{i=1}^N w_i \cdot \delta_{s^{[i]}}(x)$$

Particle Filter Algorithm in Brief

- Sample the next generation for particles using the proposal distribution
- Compute the importance weights :
weight = target distribution / proposal distribution
- Resampling: “Replace unlikely samples by more likely ones”

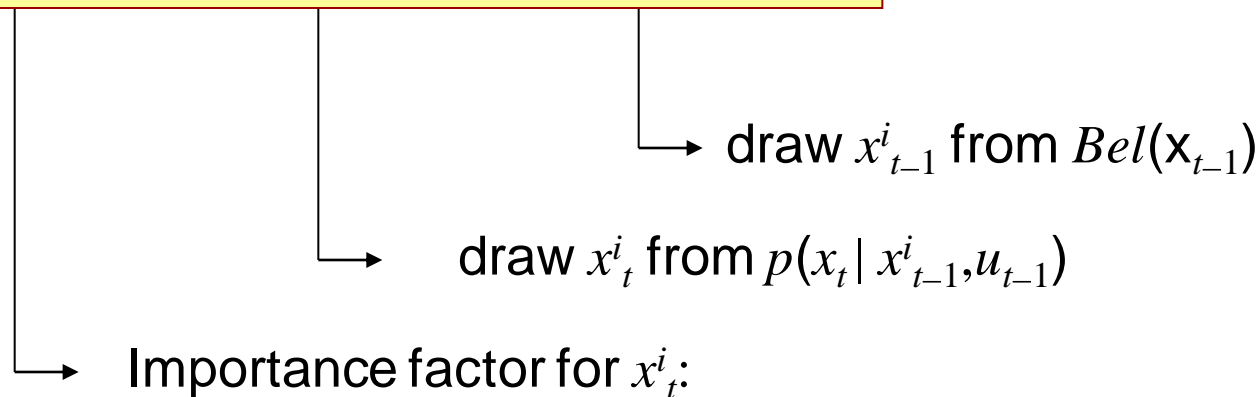
Importance Sampling Principle

- We can even use a different distribution g to generate samples from f
- By introducing an importance weight w , we can account for the “differences between g and f ”
- $w = f / g$
- f is often called target
- g is often called proposal
- Pre-condition:
 $f(x) > 0 \rightarrow g(x) > 0$



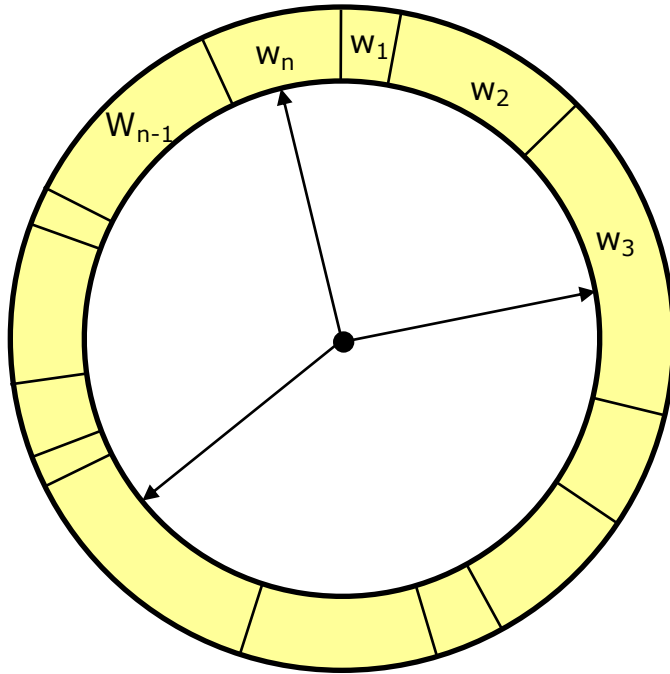
Particle Filter Algorithm

$$Bel(x_t) = \eta p(z_t | x_t) \int p(x_t | x_{t-1}, u_{t-1}) Bel(x_{t-1}) dx_{t-1}$$

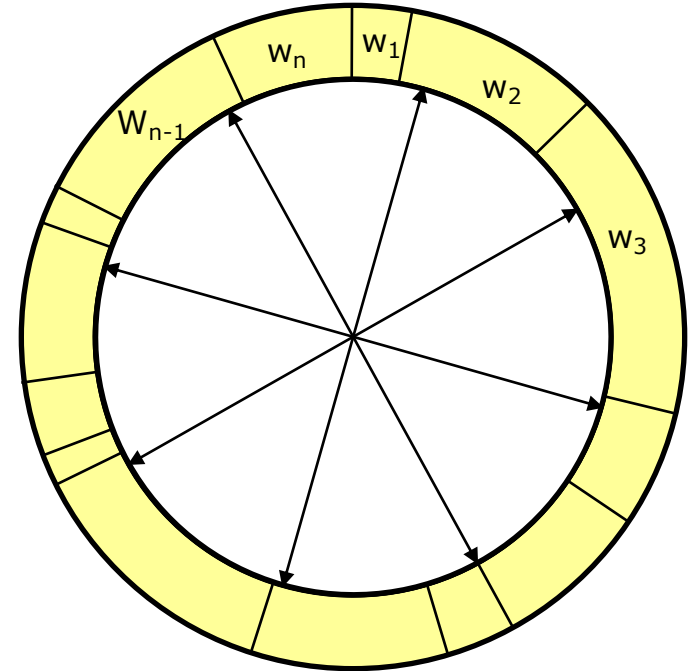


$$\begin{aligned} w_t^i &= \frac{\text{target distribution}}{\text{proposal distribution}} \\ &= \frac{\eta p(z_t | x_t) p(x_t | x_{t-1}^i, u_{t-1}) Bel(x_{t-1})}{p(x_t | x_{t-1}^i, u_{t-1}) Bel(x_{t-1})} \\ &\propto p(z_t | x_t) \end{aligned}$$

Resampling

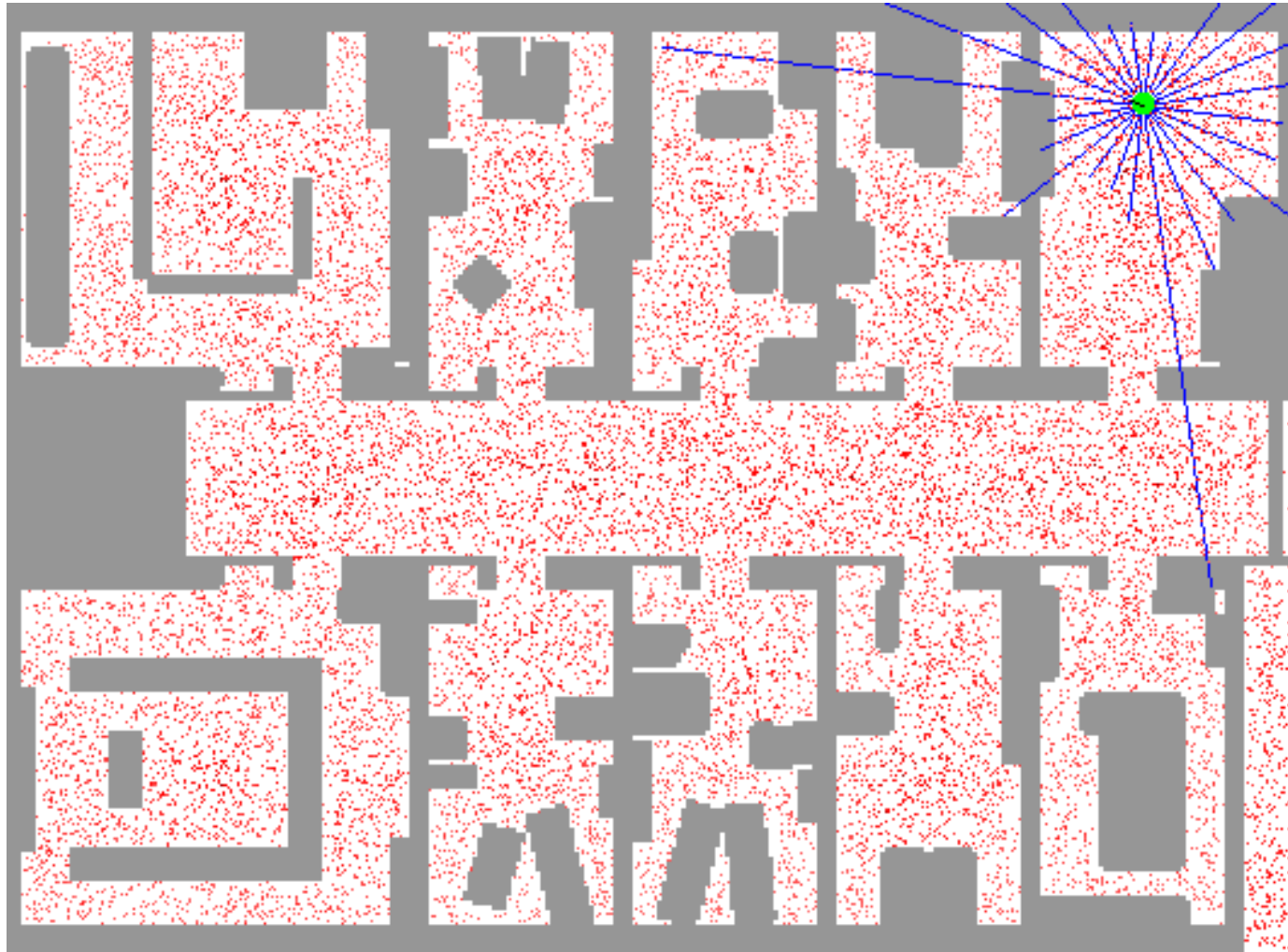


- Roulette wheel
- Binary search, $n \log n$



- Stochastic universal sampling
- Systematic resampling
- Linear time complexity
- Easy to implement, low variance

MCL Example



Mapping

Why Mapping?

- Learning maps is one of the fundamental problems in mobile robotics
- Maps allow robots to efficiently carry out their tasks, allow localization ...
- Successful robot systems rely on maps for localization, path planning, activity planning etc

Occupancy Grid Maps

- Discretize the world into equally spaced cells
- Each cells stores the probability that the corresponding area is occupied by an obstacle
- The cells are assumed to be conditionally independent
- If the pose of the robot is know, mapping is easy

Updating Occupancy Grid Maps

- Update the map cells using the **inverse sensor model**

$$Bel(m_t^{[xy]}) = 1 - \left(1 + \frac{P(m_t^{[xy]} | z_t, u_{t-1})}{1 - P(m_t^{[xy]} | z_t, u_{t-1})} \cdot \frac{1 - Bel(m_{t-1}^{[xy]})}{P(m_t^{[xy]})} \right)^{-1}$$

- Or use the **log-odds representation**

$$\bar{B}(m_t^{[xy]}) = \log odds(m_t^{[xy]} | z_t, u_{t-1}) - \log odds(m_t^{[xy]}) + \bar{B}(m_{t-1}^{[xy]})$$

$$\bar{B}(m_t^{[xy]}) := \log odds(m_t^{[xy]})$$

$$odds(x) := \left(\frac{P(x)}{1 - P(x)} \right)$$

Reflection Probability Maps

- Value of interest: $P(\text{reflects}(x,y))$
- For every cell count
 - $\text{hits}(x,y)$: number of cases where a beam ended at $\langle x,y \rangle$
 - $\text{misses}(x,y)$: number of cases where a beam passed through $\langle x,y \rangle$

$$\text{Bel}(m^{[xy]}) = \frac{\text{hits}(x, y)}{\text{hits}(x, y) + \text{misses}(x, y)}$$

SLAM

The SLAM Problem

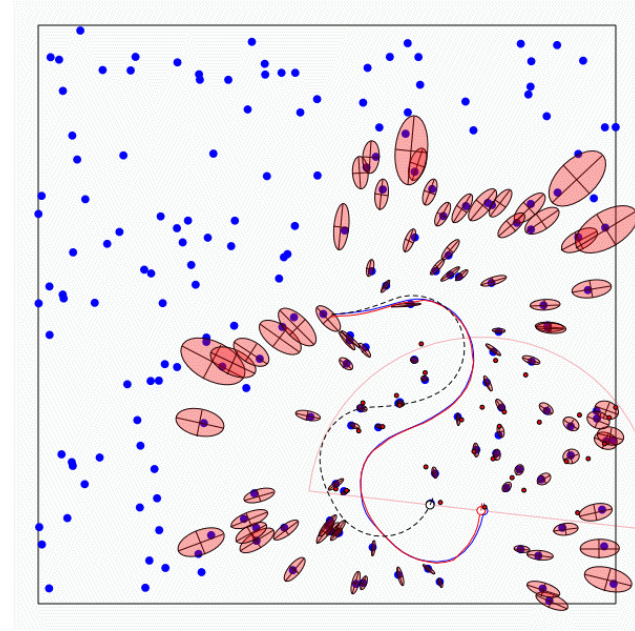
A robot is exploring an unknown, static environment.

Given:

- The robot's controls
- Observations of nearby features

Estimate:

- Map of features
- Path of the robot



Chicken-or-Egg

- 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
- Thus, SLAM is regarded as a hard problem in robotics
- A variety of different approaches to address the SLAM problem have been presented
- Probabilistic methods outperform most other techniques

SLAM:

Simultaneous Localization and Mapping

- Full SLAM: $p(x_{1:t}, m | z_{1:t}, u_{1:t})$

Estimates entire path and map!

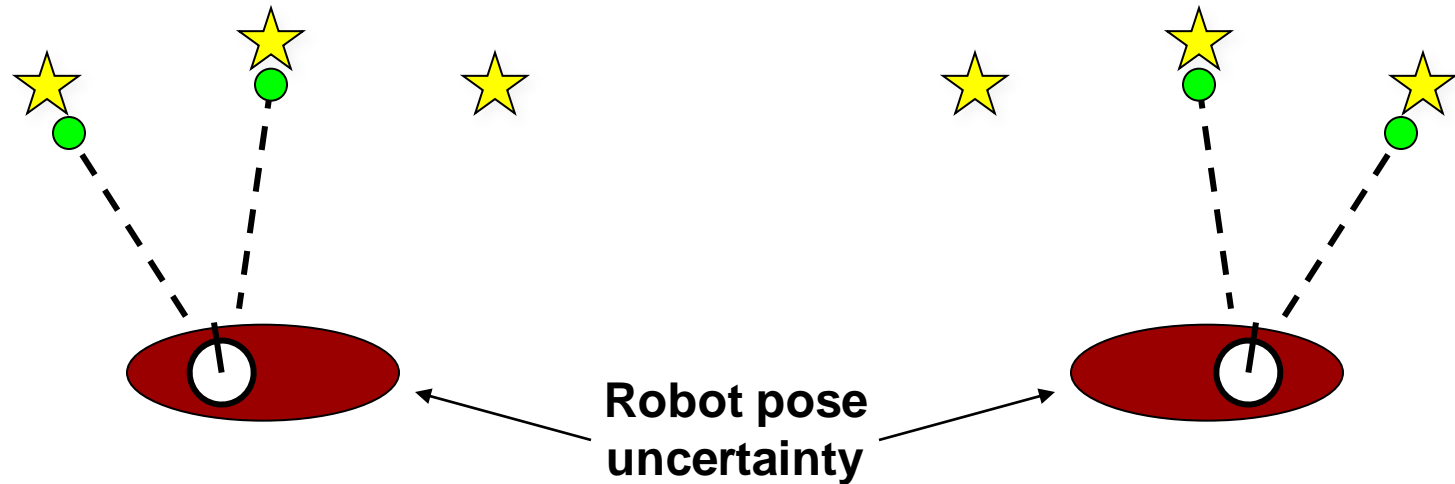
- Online SLAM:

$$p(x_t, m | z_{1:t}, u_{1:t}) = \int \int \dots \int p(x_{1:t}, m | z_{1:t}, u_{1:t}) dx_1 dx_2 \dots dx_{t-1}$$

Integrations typically done one at a time

Estimates most recent pose and 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

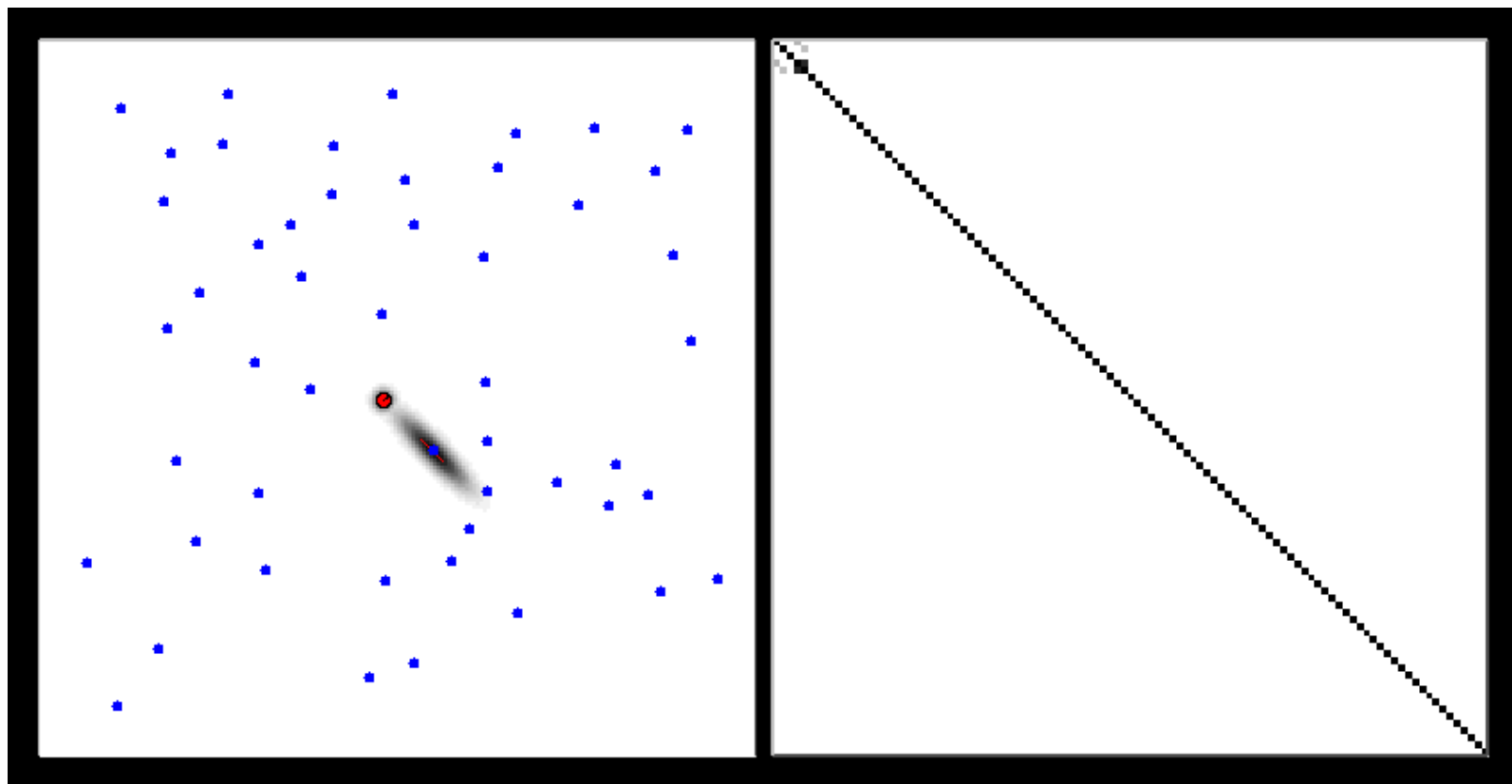
(E)KF-SLAM

- Map with N landmarks: $(3+2N)$ -dimensional Gaussian

$$Bel(x_t, m_t) = \begin{pmatrix} x \\ y \\ q \\ l_1 \\ l_2 \\ \vdots \\ l_N \end{pmatrix}, \begin{pmatrix} S_x^2 & S_{xy} & S_{xq} & S_{xl_1} & S_{xl_2} & \dots & S_{xl_N} \\ S_{xy} & S_y^2 & S_{yq} & S_{yl_1} & S_{yl_2} & \dots & S_{yl_N} \\ S_{xq} & S_{yq} & S_q^2 & S_{ql_1} & S_{ql_2} & \dots & S_{ql_N} \\ S_{xl_1} & S_{yl_1} & S_{ql_1} & S_{l_1}^2 & S_{l_1 l_2} & \dots & S_{l_1 l_N} \\ S_{xl_2} & S_{yl_2} & S_{ql_2} & S_{l_1 l_2} & S_{l_2}^2 & \dots & S_{l_2 l_N} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ S_{xl_N} & S_{yl_N} & S_{ql_N} & S_{l_1 l_N} & S_{l_2 l_N} & \dots & S_{l_N}^2 \end{pmatrix}$$

- Can handle hundreds of dimensions

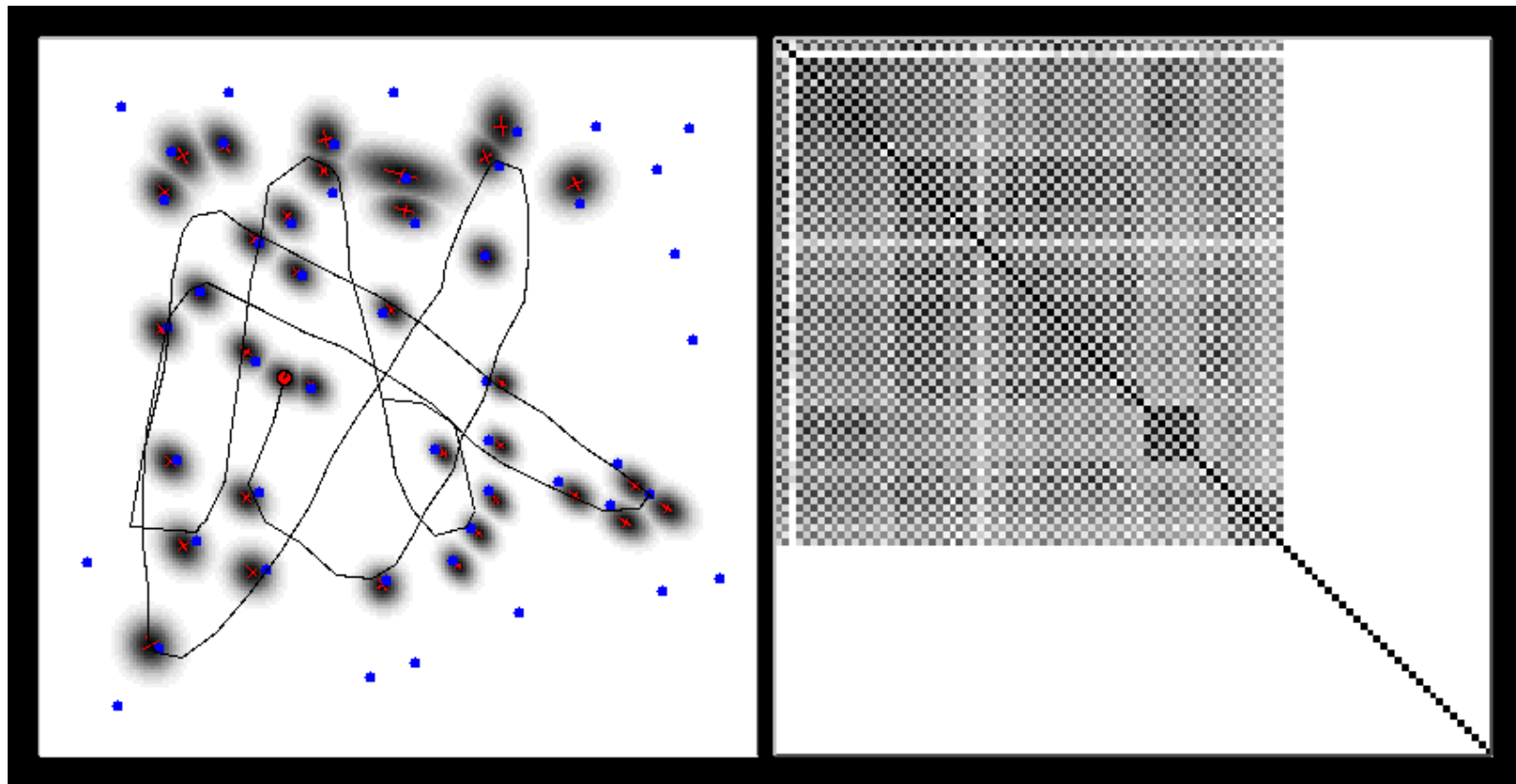
EKF-SLAM



Map

Correlation matrix

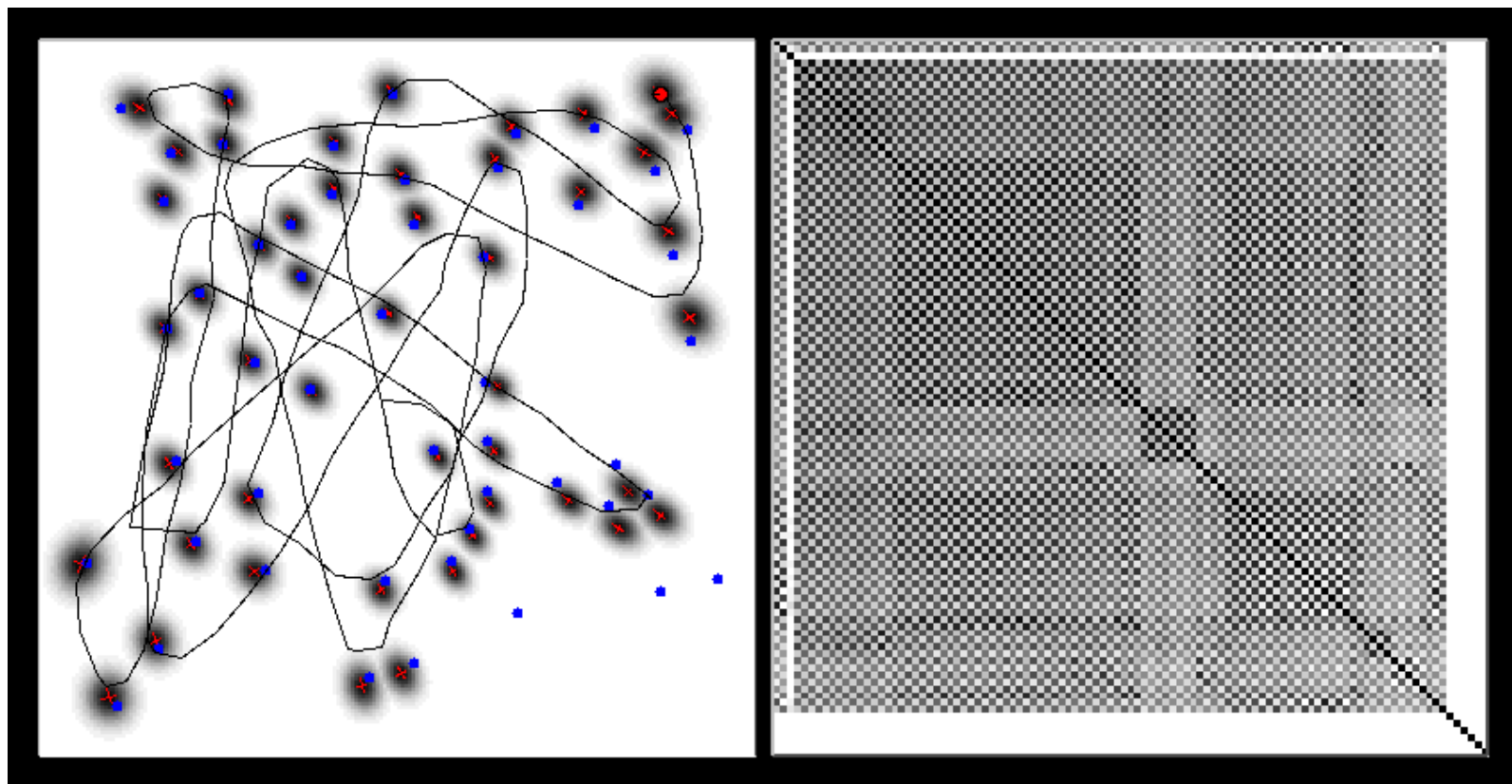
EKF-SLAM



Map

Correlation matrix

EKF-SLAM



Map

Correlation matrix

FastSLAM

- Use a particle filter for map learning
- Problem: the map is high-dimensional
- Solution: separate the estimation of the robot's trajectory from the one of the map of the environment
- This is done by means of a factorization in the SLAM posterior often called Rao-Blackwellization

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

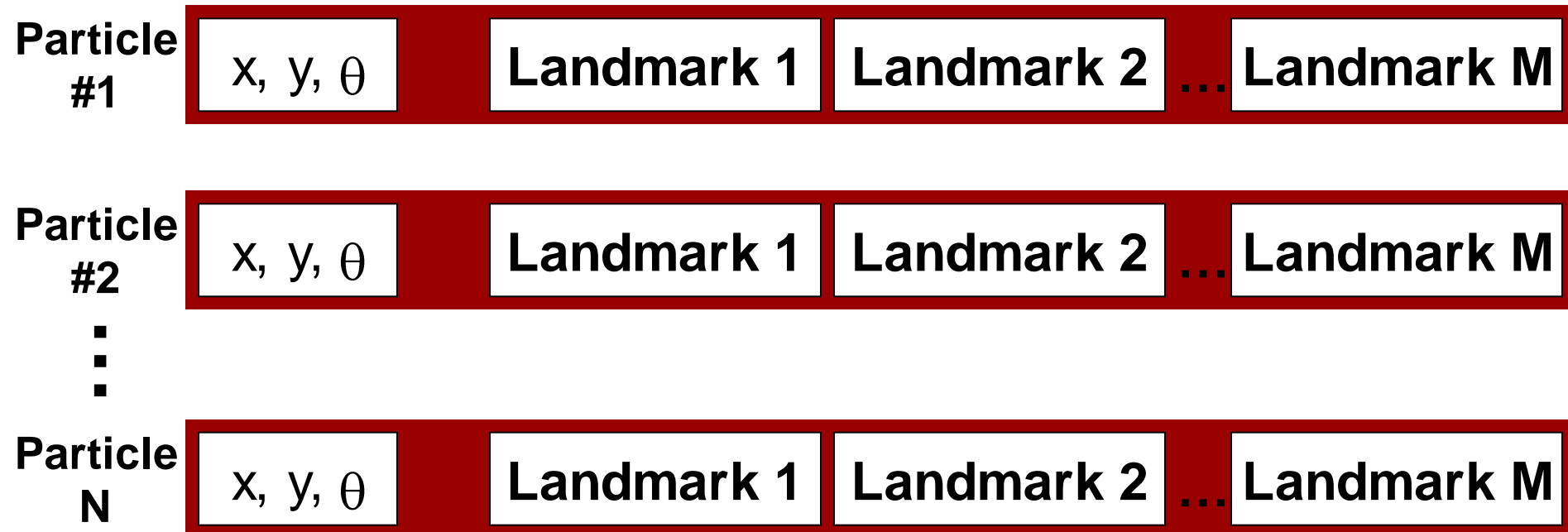
↑
Mapping with known poses

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

FastSLAM

- Rao-Blackwellized particle filtering based on landmarks
- Each landmark is represented by a 2x2 Extended Kalman Filter (EKF)
- Each particle therefore has to maintain M EKFs



Grid-based FastSLAM

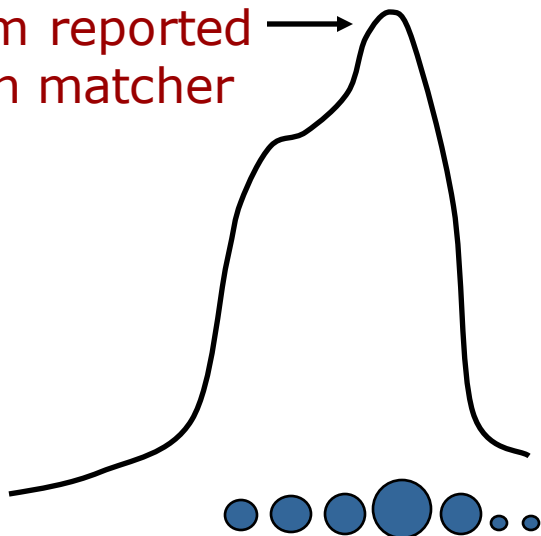
- Similar ideas can be used to learn grid maps
- To obtain a practical solution, an efficiently computable, informed proposal distribution is needed
- Idea: in the SLAM posterior, the observation model dominates the motion model (given an accurate sensor)

Proposal Distribution

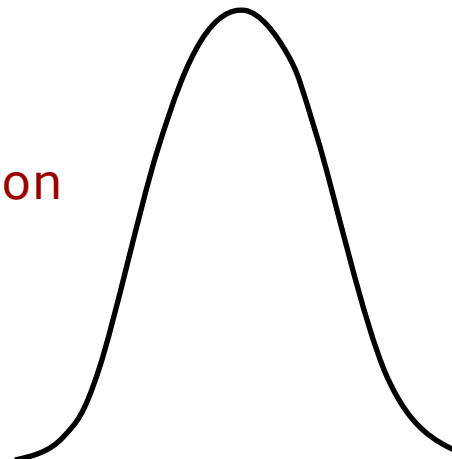
$$p(x_t | x_{t-1}^{(i)}, m^{(i)}, z_t, u_t) \simeq \frac{p(z_t | x_t, m^{(i)})}{\int_{x_t \in \{x | p(z_t | x, m^{(i)}) > \epsilon\}} p(z_t | x_t, m^{(i)}) dx_t}$$

Approximate this equation by a Gaussian:

maximum reported
by a scan matcher



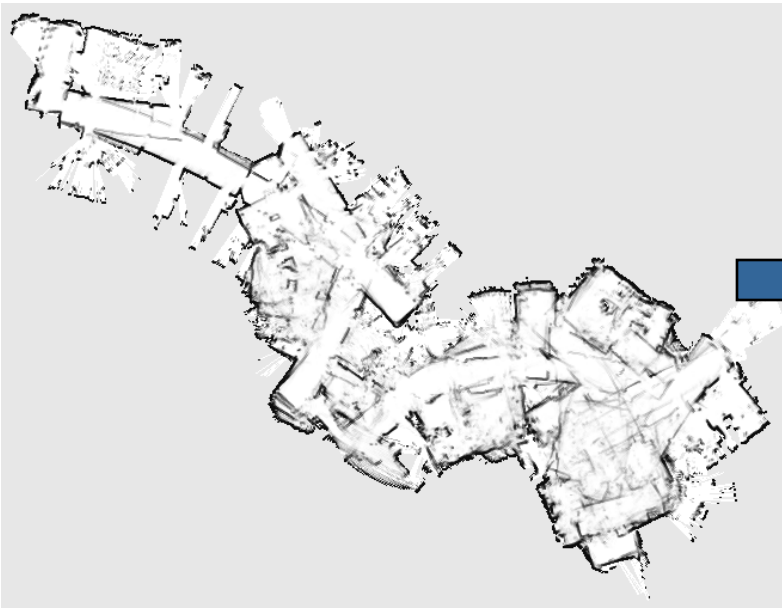
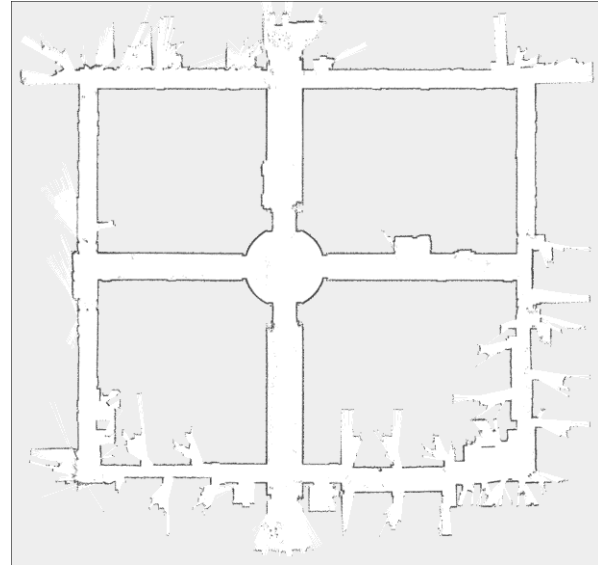
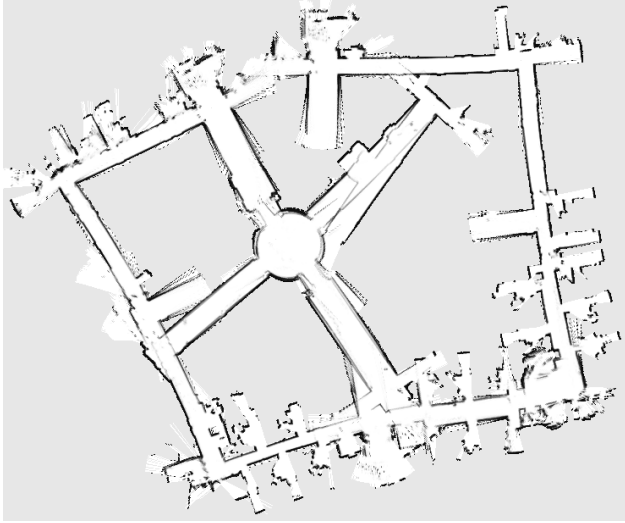
Gaussian
approximation



Draw next
generation of
samples

Sampled points around
the maximum

Typical Results



Robot Motion

Robot Motion Planning

Latombe (1991):

“...eminently necessary since, by definition, a robot accomplishes tasks by moving in the real world.”

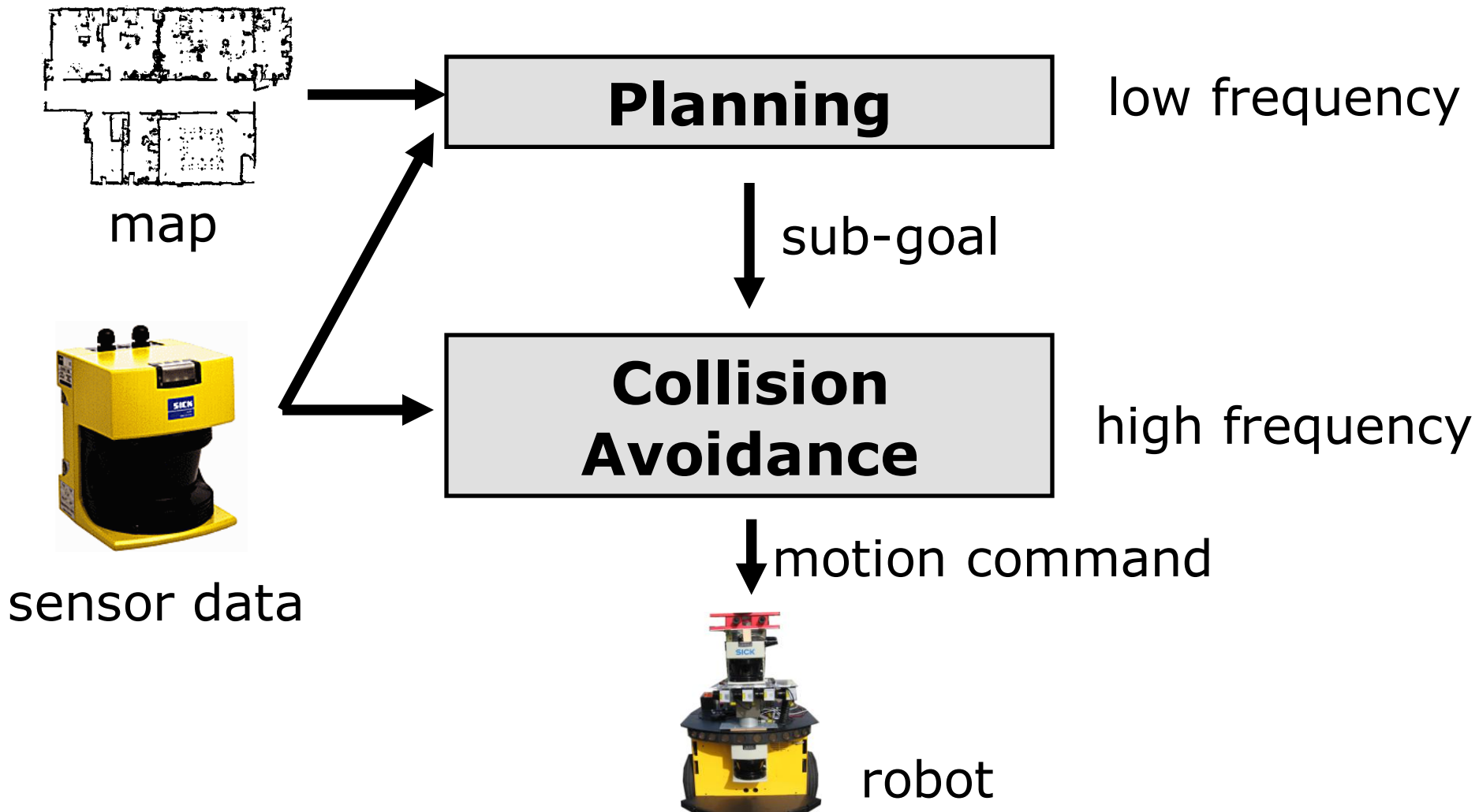
Goals:

- Collision-free trajectories.
- Robot should reach the goal location as fast as possible.

Two Challenges

- Calculate the optimal path taking potential uncertainties in the actions into account
- Quickly generate actions in the case of unforeseen objects

Classic Two-layered Architecture



Information Gain-based Exploration

- SLAM is typically **passive**, because it consumes incoming sensor data
- Exploration **actively guides the robot** to cover the environment with its sensors
- Exploration in combination with SLAM:
Acting under pose and map uncertainty
- Uncertainty should/needs to be taken into account when selecting an action
- Key question: **Where to move next?**

Mutual Information

- The mutual information I is given by the reduction of entropy in the belief

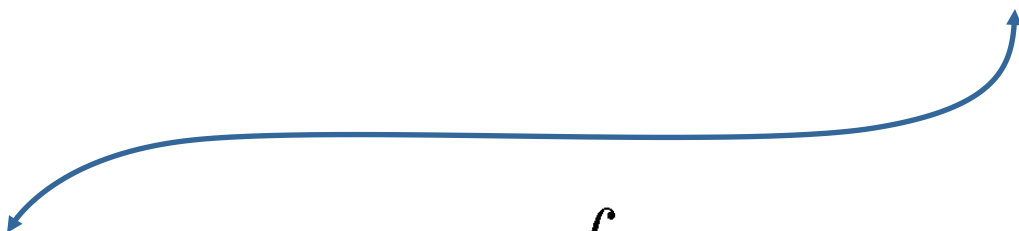
action to be carried
out

$$I(X, M; Z^a) = \text{“uncertainty of the filter”} - \text{“uncertainty of the filter after carrying out action } a\text{”}$$

Integrating Over Observations

- Computing the mutual information requires to integrate over potential observations

$$I(X, M; Z^a) = H(X, M) - H(X, M | Z^a)$$


$$H(X, M | Z^a) = \int_z p(z | a) H(X, M | Z^a = z) dz$$

↑
potential observation
sequences

Summary on Information Gain-based Exploration

- A decision-theoretic approach to exploration in the context of RBPF-SLAM
- The approach utilizes the factorization of the Rao-Blackwellization to efficiently calculate the expected information gain
- Reasons about measurements obtained along the path of the robot
- Considers a reduced action set consisting of exploration, loop-closing, and place-revisiting actions

The Exam is Approaching...

- This lecture gave a short overview over the most important topics addressed in this course
- For the exam, you need to know at least the basic formulas (e.g., Bayes filter, MCL eqs., Rao-Blackwellization, entropy, ...)

Good luck for the exam!