Introduction to Mobile Robotics

Francisco Fonseca

2022/2023

These notes are based on the slides provided during the course and the book *Probabilistic Robotics* by Thrun, Burgard and Fox.

Contents

1 Locomotion - 03

Instantaneous Center of Curvature:

- For rolling motion to occur, each wheel has to move along its y-axis

Differential Drive:

$$ICC = [x - R\sin\theta, y + R\cos\theta]$$
$$\omega(R + \frac{l}{2}) = v_r$$
$$\omega(R - \frac{l}{2}) = v_l$$
$$R = \frac{l}{2}\frac{(v_l + v_r)}{(v_r - v_l)}$$
$$\omega = \frac{v_r - v_l}{l}$$
$$v = \frac{v_r + v_l}{2}$$

Differential Drive Motion Patterns:

- Foward motion:
$$v_l = v_R$$
- Left motion:
$$v_l < v_r \wedge v_l > 0$$
- Left circular motion:
$$v_l = 0 \wedge v_r > 0$$
- Left turn in place motion:
$$v_l = -v_r$$

Differential Drive Forward Kinematics:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} \cos(\omega\delta_t) & -\sin(\omega\delta_t) & 0 \\ \sin(\omega\delta_t) & \cos(\omega\delta_t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - ICC_x \\ y - ICC_y \\ \theta \end{bmatrix} \begin{bmatrix} ICC_x \\ ICC_y \\ \omega\delta_t \end{bmatrix}$$

$$\begin{cases} x(t) = \int_0^t v(t')\cos[\theta(t')]\,dt' \\ y(t) = \int_0^t v(t')\sin[\theta(t')]\,dt' \\ \theta(t) = \int_0^t \omega(t')\,dt' \end{cases}$$

In 1, we could replace $v(t')$ with $\frac{v_r(t')+v_l(t')}{2}$ and $\omega(t')$ with $\frac{v_r(t')-v_l(t')}{l}$.

Ackermann Drive:

The angle ϕ is the angle between the line that passes the back wheels and the ICC and the line that passes in the front wheel and the ICC. The d is the distance from the line of passing through the back wheels and the centre of the front wheel.

$$ICC = [x - R\sin\theta, y + R\cos\theta]$$
$$R = \frac{d}{\tan\phi}$$
$$\omega(R + \frac{l}{2}) = v_r$$
$$\omega(R - \frac{l}{2}) = v_l$$
$$R = \frac{l}{2}\frac{(v_l + v_r)}{(v_r - v_l)}$$
$$\omega = \frac{v_r - v_l}{l}$$

Mecanum Wheels Motion Patterns:

Assuming v_0, v_1, v_2 and v_3 in clock-wise order starting at the top left corner of a the velocities of each of a 4 wheeled robot:

$$\begin{cases} v_x = \frac{v_0 - v_1 + v_2 - v_3}{4} \\ v_y = \frac{v_0 + v_1 + v_2 + v_3}{4} \\ v_\theta = \frac{v_0 - v_1 - v_2 + v_3}{4} \\ v_{error} = \frac{v_0 + v_1 - v_2 - v_3}{4} \end{cases}$$

Different motion patterns can be obtained by the direction of spinning in each wheel. Forward motion will have all wheels spinning in a positive direction, right motion will have the first and third wheels positive and second and fourth negative, circular motion will have the first and fourth forward and second and third backward and to stop, you can set the first and second forward, and the third and fourth backwards.

Non-Holonomic Constraints:

- Non-holonomic constraints limit the possible incremental movements within the configuration space of the robot.
- Robots with differential drive or synchro-drive move on a circular trajectory and cannot move sideways.
- Mecanun-wheeled robots can move sideways (they have no non-holonomic constraints).

Holonomic vs. Non-Holonomic:

- Non-holonomic constraints reeduce the control space with respect to the current configuration *(moving sideways is impossible)*.
- Holonomic constraints reduce the configuration space *(a train on tracks)*.

The Synchro-drive, Differential drive and Ackermann drive have **non-holonomic contraints**.

The mecanum wheels is a drive without non-holonomic constraints.

Dead Reckoning and Odometry:

- Estimating the motion based on the issued controls/wheel encoder readings
- Integrated over time

2 Sensors - 04

Sensors of Wheeled Robots:

Allow for the perception of the environment.

- Active:
  - Ultrasound *(Time of flight)*
    - Signal profile [Polaroid]
    - Sources of error:
      - Opening angle
      - Crosstalk
      - Specular reflection
    - Parallel Operation: with 15 degrees opening angle, 24 sensors are needed to cover the whole 360 degrees area. To allow for frequent update the sensors have to be fired in parallel which increases the risk of crosstalk.
  - Lase range finder *(Time of flight)*
    - High precision
    - Wide field of view
    - Can be used for emergency stops
    - Laser data comes as an array of readings
    - Assume field of view of 180 degrees
    - First beams start at $-\frac{1}{2}$ of the FOV
    - Maximum range of usually 80m.
    - Time of Flight Sensors: $d = v\frac{t}{2}$ where v is the speed of the signal and t the time elapsed between broadcast of the signal and the reception of the echo.
  - Infrared *(Phase shift)*
    - Structured Light Sensors
    - Infrared projector illuminates the scene with a known light pattern
    - Scene is captured by an infrared sensor, depth is derived from pattern distortion
    - Pros: cheap, dense range image, precise at a range up to 5m
    - Cons: low operational range, sensitive to sunlight, dark surfaces and reflecting surfaces
- Passive:
  - Cameras *(Intensity-based)*
  - Tactiles *(Intensity-based)*
    - Bumper sensor *(spring + contact)*
    - Touch sensor

3 Probabilistic Robotics - 05

Explicit representation of uncertainty:

- Perception = state estimation
- Action = utility optimisation

Axioms of Probability Theory:

$P(A)$ denotes the probability that proposition A is true:

- $0 \leq P(A) \leq 1$
- $P(True) = 1$
- $P(A \vee B) = P(A) + P(B) - P(A \wedge B)$

Using the Axioms:

- $P(A \vee \neg A) = P(A) + P(\neg A) - P(A \wedge \neg A)$
- $P(True) = P(A) + P(\neg A) - P(False)$
- $1 = P(A) + P(\neg A) - 0$
- $P(\neg A) = 1 - P(A)$

Discrete Random Variables:

- X denotes a random variable
- X can take on a countable number of values in $\{x_1, x_2, ..., x_n\}$
- $P(X = x_i)$ or $P(x_i)$ is the probability that the random variable X takes on the value of x_i
- $P(\cdot)$ is called probability mass function

Continuous Random Variables:

- X takes on values in the continuum
- $p(X = x)$ or $p(x)$ is a probability density function:

$$P(x \in [a, b]) = \int_a^b p(x) * dx$$

Probability sums up to one:

- Discrete case: $\sum_x P(x) = 1$
- Continuous case: $\int_X P(x)\, dx = 1$

Joint and Conditional Probability:

- $P(X = x \text{ and } Y = y) = P(x, y)$
- If X and Y are independent then

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

- $P(x|y)$ is the probability of x given y:

$$P(x|y) = \frac{P(x, y)}{P(y)}$$

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

- If X and Y are independent then

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

Law of Total Probability:

Discrete case

$$P(x) = \sum_y P(x|y)P(y)$$

Continuous case

$$p(x) = \int p(x|y)P(y)\,dy$$

Marginalisation:

Discrete case

$$P(x) = \sum_y P(x, y)$$

Continuous case

$$p(x) = \int p(x, y)\,dy$$

Bayes Formula:

$$P(x, y) = P(x|y)P(y) = P(y|x)P(x)$$
$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$
$$\text{Posterior} = \frac{\text{Likelihood} * \text{Prior}}{\text{Evidence}}$$

Normalisation:

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$
$$\text{With } P(y) = \sum_x P(y|x)P(x) :$$
$$P(x|y) = \frac{P(y|x)P(x)}{\sum_x P(y|x)P(x)}$$
$$P(y) \text{ is independent of } x \text{ and thus constant for all } x :$$
$$P(x|y) = \eta P(y|x)P(x)$$

Bayes Rule with Background Knowledge:

$$P(x|y, a) = \frac{P(y|x, a)P(x|a)}{P(y|a)}$$

Conditional Independence:

- $P(x, y|z) = P(x|z)P(y|z)$
- Equivalent to $P(x|z) = P(x|z, y)$ and $P(y|z) = P(y|z, x)$
- Does not necessarily mean $P(x, y) = P(x)P(y)$
- Marginal independence does not mean independence

Causal vs. Diagnostic Reasoning:

- $P(open|z)$ is diagnostic
- $P(z|open)$ is causal
- In some situations, causal knowledge is easier to obtain
- Bayes rul allows us to usse causal knowledge:

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

Recursive Bayesian Updating:

$$P(x|\underbrace{z_1, \cdots, z_{n-1}}_{a}, z_n) = \frac{P(z_n|x, \overbrace{z_1, \cdots, z_{n-1}}^{a})P(x|\overbrace{z_1, \cdots z_{n-1}}^{a})}{P(z_n|\underbrace{z_1, \cdots, z_{n-1}}_{a})}$$

Markov Assumption:

z_n is independent of $z_1, \ldots, z_{n-1}$ given we know x:

$$\begin{aligned} P(x|\underbrace{z_1, \cdots, z_{n-1}}_{a}, z_n) &= \frac{P(z_n|x, \overbrace{z_1, \cdots, z_{n-1}}^{a})P(x|\overbrace{z_1, \cdots z_{n-1}}^{a})}{P(z_n|\underbrace{z_1, \cdots, z_{n-1}}_{a})} \\ &= \alpha P(z_n|x)P(x|a) \\ &= \alpha P(x) \prod_{i=1\ldots n} P(z_i|x) \end{aligned}$$

Modeling Actions:

- To incorporate the outcome of an action u into the current *belief*, we use the conditional probability $P(x|u, x')$
- This term specifies the probability that executing u changes the state from x' to x

Integrating the Outcome of Actions:

Discrete case

$$P(x|u) = \sum_y P(x|u, x')P(x', \cancel{u})$$

Continuous case

$$p(x|u) = \int p(x|u, x')P(x', \cancel{u})\, dy$$

We will make an independence assumption to get rid o the u in the second factor of the sum.

Bayes Filters Framework:

- Given:
  - Stream of obervations z and action data u:

$$d_t = \{u_1, z_1, \ldots, u_t, z_t\}$$

  - Sensor model $P(z|u)$
  - Action model $P(x|u, x')$
  - Prior probability of the system state $P(x)$
- Wanted:
  - Estimate of the state X of a dynamical system
  - The posterior of the state is also called belief:

$$Bel(x_t) = P(x_t|u_1, z_1, \ldots, u_t, z_t)$$

Markov Assumption:

$$P(z_t|x_{0:t}, z_{1:t-1}, u_{1:t}) = P(z_t|x_t)$$
$$P(x_t|x_{1:t-1}, z_{1:t-1}, u_{1:t}) = P(x_t|x_{t-1}, u_t)$$

Underlying Assumptions:

- Static world
- Independent noise
- Perfect model, no approximation errors

Bayes Filters:

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

Bayes

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

Markov

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

Total Probability

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

Markov

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

Markov

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

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

Algorithm 1 Bayes_filter($Bel(x), d$)

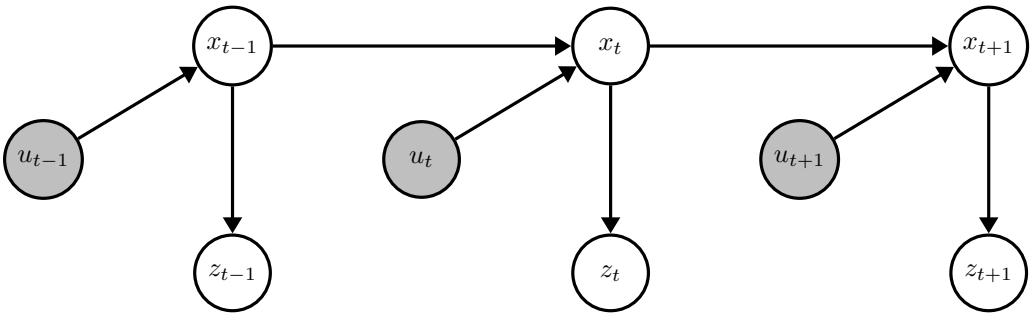
$\eta = 0$
if d is a perceptual data item z **then**
  for all x **do**
    $Bel'(x) = P(z|x)Bel(x)$
    $\eta = \eta + Bel'(x)$
  end for
  for all x **do**
    $Bel'(x) = \eta^{-1}Bel'(x)$
  end for
else if d is an action data item u **then**
  for all x **do**
    $Bel'(x) = \int P(x|u, x')Bel'(x)\ dx'$
  end for
end if
Return $Bel'(x)$

Bayes Filters are Familiar:

- Kalman filters
- Particle filters
- Hidden Markov models
- Dynamic Bayesian networks
- Partially Observable Markov Decision Processes (POMDPs)

4 Probabilistic Motion Models - 06

Dynamic Bayesian Network for Controls, States, and Perceptions:



Probabilistic Motion Models:

- To implement the Bayes Filter, we need the transition model $p(x_t|x_{t-1}, u_t)$
- The term $p(x_t|x_{t-1}, u_t)$ specifies a posterior probability, that action u_t carries the robot from x_{t-1} to x_t

Coordinate Systems:

- Six parameters can describe the configuration of a typical wheeled robot in 3D
- These are the three-dimensional Cartesian coordinates plus the three Euler angles for roll, pitch, and yaw
- For simplicity, we consider robots operating on a planar surface
- The state space of such systems is three-dimensional $(x.y, \theta)$

Typical Motion Models:

- In practice, these are the two most often used 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 applies when no wheel encoders are given
- Velocity-based models calculate the new pose based on the velocities and the time elapsed

Wheel encoders:

These modules provide $+5V$ output when they *see* white, and a $0V$ output when they *see* black.

These disks are manufactured from high-quality laminated colour plastic to offer a very crisp black-to-white transition. This enables a wheel encoder sensor to easily see the transitions.

Dead Reckoning:

- Derived from *deduced reckoning*
- Mathematical procedure for determining the present location of a vehicle
- Achieved by calculating the current pose of the vehicle based on its velocities and the time elapsed
- Historically used to log the position of ships

Reason for Motion Errors of Wheeled Robots:

- Ideal case - perfectly horizontal floor
- Different wheel diameters
- Bump, Carpet, etc

Odometry Model:

- Robot moves from $\langle \bar{x}, \bar{y}, \bar{\theta} \rangle$ to $\langle \bar{x}', \bar{y}', \bar{\theta}' \rangle$

- Odometry information $u = \langle \delta_{rot1}, \delta_{rot2}, \delta_{trans} \rangle$

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

***atan2* Function**: Extends the inverse tangent and correctly copes with the signs of x and y:

$$\text{atan2}(y, x) = \begin{cases} \text{atan}(\frac{y}{x}), & \text{if } x > 0 \\ \text{sign}(y)(\pi - \text{atan}(\frac{y}{x})), & \text{if } x < 0 \\ 0, & \text{if } x = y = 0 \\ \text{sign}(y)\frac{\pi}{2}, & \text{if } x = 0, y \neq 0 \end{cases}$$

Noise Model for Odometry:

- The measured motion is given by the true motion corrupted with noise:

$$\begin{aligned} \hat{\delta}_{trans} &= \delta_{trans} + \varepsilon_{\alpha_3|\delta_{trans}|+\alpha_4(|\delta_{rot1}|+|\delta_{rot2}|)} \\ \hat{\delta}_{rot1} &= \delta_{rot1} + \varepsilon_{\alpha_1|\delta_{rot1}|+\alpha_2|\delta_{trans}|} \\ \hat{\delta}_{rot2} &= \delta_{rot2} + \varepsilon_{\alpha_1|\delta_{rot2}|+\alpha_2|\delta_{trans}|} \end{aligned}$$

Typical Distributions for Probabilistic Motion Models:

- Normal distribution:

$$\varepsilon_{\sigma^2}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{x^2}{\sigma^2}}$$

- Triangular distribution:

$$\varepsilon_{\sigma^2}(x) = \begin{cases} 0, & \text{if } |x| > \sqrt{6\sigma^2} \\ \frac{\sqrt{6\sigma^2} - x}{6\sigma^2} \end{cases}$$

Algorithm 2 motion_model_odometry$(x, x', \bar{x}, \bar{x}')$

1: $\delta_{trans} = \sqrt{(\bar{x}' - \bar{x})^2 + (\bar{y}' - \bar{y})^2}$
2: $\delta_{rot1} = \text{atan2}(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta}$
3: $\delta_{rot2} = \bar{\theta}' - \bar{\theta} - \delta_{rot1}$
4: $\hat{\delta}_{trans} = \sqrt{(x' - x)^2 + (y' - y)^2}$
5: $\hat{\delta}_{rot1} = \text{atan2}(y' - y, x' - x) - \theta$
6: $\hat{\delta}_{rot2} = \theta' - \theta - \hat{\delta}_{rot1}$
7: $p_1 = prob(\delta_{trans} - \hat{\delta}_{trans}, \alpha_3|\delta_{trans}| + \alpha_4(|\delta_{rot1}| + |\delta_{rot2}|))$
8: $p_2 = prob(\delta_{rot1} - \hat{\delta}_{rot1}, \alpha_1|\delta_{rot1}| + \alpha_2|\delta_{trans}|)$
9: $p_3 = prob(\delta_{rot2} - \hat{\delta}_{rot2}, \alpha_1|\delta_{rot2}| + \alpha_2|\delta_{trans}|)$
10: Return $p_1 p_2 p_3$

Application:

- Repeated application of the motion model for short movements
- Typical banana-shaped distributions obtained for the 2d-projection of the 3d posterior

How to Sample from Normal or Triangular Distributions:

Algorithm 3 sample_normal_distribution(b)

1: Return $\frac{1}{2}\sum_{i=1}^{12} rand(-b, b)$

Algorithm 4 sample_triangular_distribution(b)

1: Return $\frac{\sqrt{6}}{2}[rand(-b, b) + rand(-b, b)]$

Rejection Sampling:

- Sampling from arbitrary distributions
- Sample x from a uniform distribution from $[-b, b]$
- Sample y from $[0, max(f)]$

$$\begin{cases} \text{keep the sample } x, & \text{if } f(x) > y \\ \text{reject it} \end{cases}$$

Algorithm 5 sample_distribution(f,b)

1: **repeat**
2: $\quad x = rand(-b, b)$
3: $\quad rand(0, max\{f(x)|x \in [-b, b]\})$
4: **until** $y \leq f(x)$
5: Return x

Sample Odometry Motion Model:

- $x = \langle x, y, \theta \rangle$
- $u = \langle \delta_{rot1}, \delta_{rot2}, \delta_{trans} \rangle$

Algorithm 6 sample_motion_model(u, x)

1: $\hat{\delta}_{rot1} = \delta_{rot1} + sample\_normal\_distribution(\alpha_1|\delta_{rot1}| + \alpha_2|\delta_{trans}|)$
2: $\hat{\delta}_{rot2} = \delta_{rot2} + sample\_normal\_distribution(\alpha_1|\delta_{rot2}| + \alpha_2|\delta_{trans}|)$
3: $\hat{\delta}_{trans} = \delta_{trans} + sample\_normal\_distribution(\alpha_3|\delta_{trans}| + \alpha_4(|\delta_{rot1}| + |\delta_{rot2}|))$
4: $x' = x + \hat{\delta}_{trans} \cos(\theta \hat{\delta}_{rot1})$
5: $y' = y + \hat{\delta}_{trans} \sin(\theta + \hat{\delta}_{rot1})$
6: $\theta' = \theta + \hat{\delta}_{rot1} + \hat{\delta}_{rot2}$
7: Return $\langle x', y', \theta' \rangle$

Velocity-Based Model:

Algorithm 7 motion_model_velocity(x_t, u_t, x_{t_1})

1: $= \frac{1}{2} \frac{(x-x')\cos\theta + (y-y')\sin\theta}{(y-y')\cos\theta + (x-x')\sin\theta}$
2: $x^* = \frac{x+x'}{2} + (y - y')$
3: $y^* = \frac{y+y'}{2} + (x' - x)$
4: $r^* = \sqrt{(x - x*)^2 + (y - y*)^2}$
5: $\Delta\theta = \text{atan2}(y' - y^*, x' - x^*) - \text{atan2}(y - y^*, x - x^*)$
6: $\hat{v} = \frac{\Delta\theta}{\Delta t} r^*$
7: $\hat{\omega} = \frac{\Delta\theta}{\Delta t}$
8: $\hat{\gamma} = \frac{\theta' - \theta}{\Delta t} - \hat{\omega}$
9: Return $prob(v - \hat{v}, \alpha_1 v^2 + \alpha_2 \omega^2) prob(\omega - \hat{\omega}, \alpha_3 v^2 + \alpha_4 \omega^2) prob(\hat{\gamma}, \alpha_5 v^2 + \alpha_6 \omega^2)$

Sampling from Velocity Model:

Algorithm 8 sample_motion_model_velocity(u_t, x_{t-1})

1: $\hat{v} = v + sample\_normal\_distribution(\alpha_1 v^2 + \alpha_2 \omega^2)$
2: $\hat{\omega} = \omega + sample\_normal\_distribution(\alpha_3 v^2 + \alpha_4 \omega^2)$
3: $\hat{\gamma} = sample\_normal\_distribution(\alpha_5 v^2 + \alpha_6 \omega^2)$
4: $x' = x - \frac{\hat{v}}{\hat{\omega}} \sin\theta + \frac{\hat{v}}{\hat{\omega}} \sin(\theta + \hat{\omega}\Delta t)$
5: $y' = y + \frac{\hat{v}}{\hat{\omega}} \cos\theta - \frac{\hat{v}}{\hat{\omega}} \cos(\theta + \hat{\omega}\Delta t)$
6: $\theta' = \theta + \hat{\omega}\Delta t + \hat{\gamma}\Delta t$
7: Return $(x', y', \theta')^T$

Map-Consistent Motion Model:

If there is a barrier of some kind, we can add the map as background knowledge m which will mean $p(x'|u, x) \neq p(x'|u, x, m)$ and we do the approximation $p(x'|u, x, m) = \eta p(x'|m) p(x'|u, x)$.

5 Probabilistic Sensor Models - 07

Sensors for Mobile Robots:

- Contact sensors - Bumpers
- Proprioceptive 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)
  - LiDAR (triangulation, time of flight, phase)
  - Light-based (intensity)
- Visual sensors: Cameras (monocular, stereo, sheet of light)
- Exteroceptive:
  - GPS
  - Active beacons

Proximity Sensors:

- The central task is to determine $P(z|x)$, i.e., the probability of a measurement z given that the robot is at position x (and given the map).

Types of Range Measurement Models:

- Beam-based model:
  - tries to explain the measurement
  - model parameters can be learned
  - requires ray-casting
- Scan-based model:
  - tries to be fast
  - ignores the fact that the measurement is a ray

Independece Assumption of Both Models:

- Scan z consists of K measurements
$$z = \{z_1, z_2, \ldots, z_K\}$$
- Individual measurements are independent given the robot position
$$P(z|x,m) = \prod_{k=1}^{K} P(Z_K|x,m)$$

Typical Errors of Range Measurements:

- Beams reflected by obstacles
- Beams reflected by people / caused by crosstalk
- Random measurements
- Maximum range measurements

Proximity Measurement:

- Measurement can be caused by:
  - a known obstacle
  - crosstalk
  - an unexpected obstacle
  - missing all obstacles
- Noise is due to uncertainty in:
  - measuring distance to known obstacle
  - position of known obstacles
  - position of additional obstacles
  - missed obstacles

Beam-based Model:

- Considers beams individually
- Uses the following approximation:

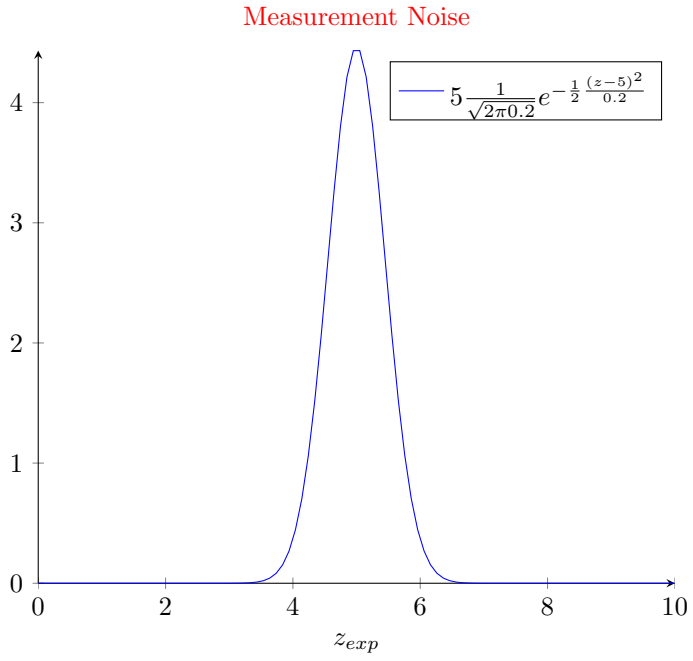
$$P(z|x) = P(z|x,m) \approx P(z|z_{exp}(x,m)) = P(z|z_{exp})$$

- $z_{exp}(x,m)$ equals the distance to the closest obstacle in direction of measurement (obtained by ray casting)

Measurement Noise:

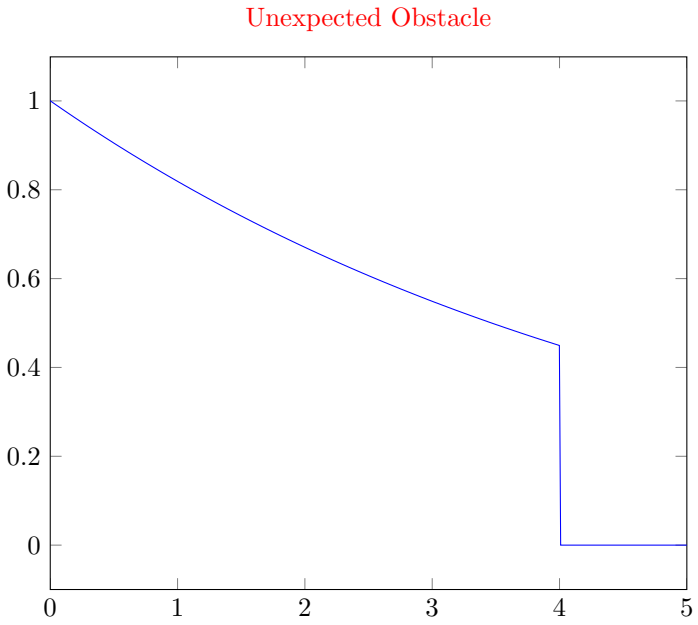
$$P_{hit}(z|x,m) = \eta \frac{1}{\sqrt{2\pi b}} e^{\frac{1}{2}\frac{(z-z_{exp})^2}{b}}$$

Below an exmaple with $z_{exp} = 5, b = 0.2, \eta = 5$



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

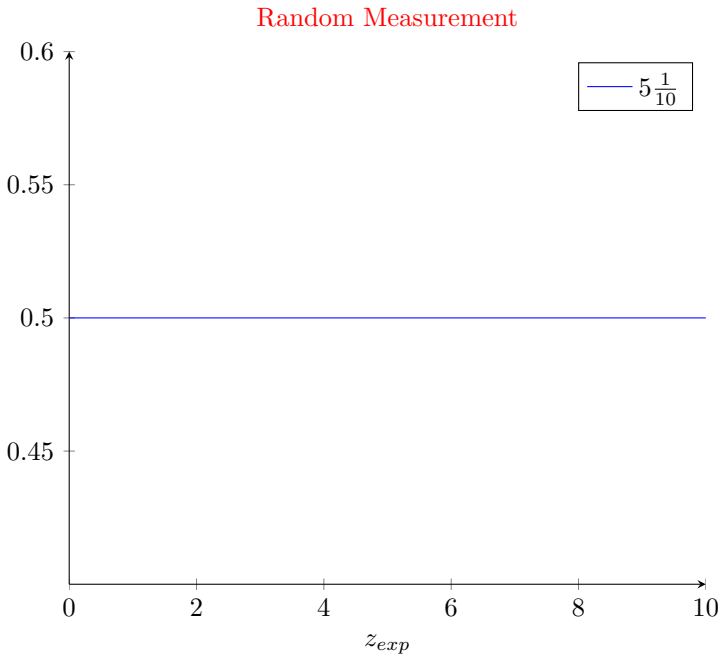
Below an example with $z_{exp} = 4, \lambda = 0.2, \eta = 5$



Random measurement:

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

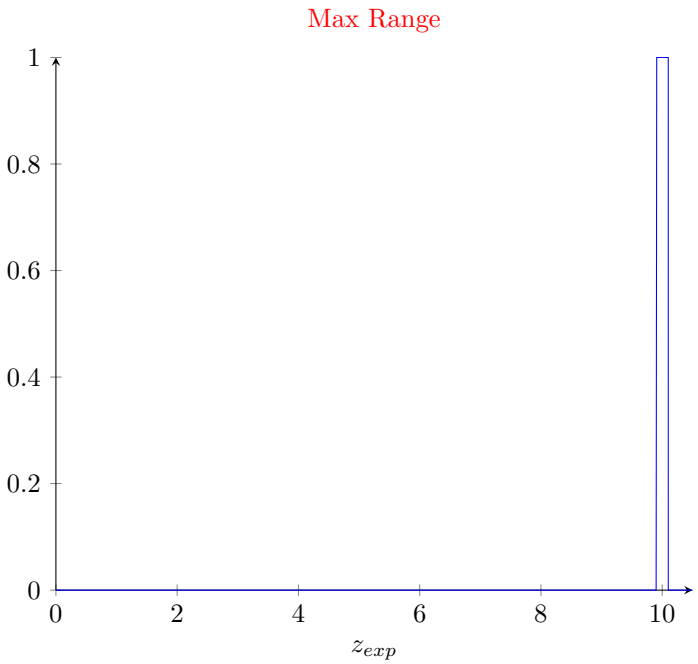
Below an exmaple with $z_{exp} = 5, z_{max} = 10, \eta = 5$



Max range:

$$P_{max}(z|x,m) = \begin{cases} 1 & z = z_{max} \\ 0 & otherwise \end{cases}$$

Below an exmaple with $z_{max} = 10$



Resulting Mixture Density:

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

Approximation:

- Maximize log likelihood of the data $P(z|z_{exp})$.
- Search space of $n-1$ parameters and deterministically compute the $n-th$ parameter to satisfy normalisation constraint.
  - Hill climbing
  - Gradient descent
  - Genetic algorithms

Scan-based Model:

Main idea is to check the end point instead of following along the beam.

- Probability is a mixture of:
  - a Gaussian distribution with mean at distance to closest obstacle
  - a uniform distribution for random measurements
  - a small uniform distribution for max range measurements
- Can be efficiently stored in a two-dimensional *likelihood field*
- Independence between different beams is assumed

Properties of Scan-based Model:

- Highly efficient uses $2D$ tables only
- Likelihood field is smooth w.r.t. small changes in robot position
- Allows gradient descent, scan matching
- Ignores physical properties of beams

Additional Models of Proximity Sensors:

- Map matching (sonar, laser): generate small, local maps from sensor data and match local maps against global model
- Scan matching (laser): map is represented by scan endpoints, match scan into this map
- Features (sonar, laser, vision): Extract features such as doors and hallways from sensor data

Landmarks:

- Active beacons (radio, GPS)
- Passive (visual, retro-reflective)
- Standard approach is triangulation
- Sensor provides:
  - distance, or
  - bearing, or
  - distance and bearing

Probabilistic Model:

$$z = \langle I, d, \alpha \rangle$$
$$x = \langle x, y, \theta \rangle$$

Algorithm 9 landmark_detection_model(z, x, m)

1: $\hat{d} = \sqrt{(m_x(i) - x*)^2 + (m_y(i) - y*)^2}$
2: $\hat{\alpha} = \text{atan2}(m_y(i) - y, m_x(i) - x) - \theta$
3: $p_{det} = prob(\hat{d} - d, \varepsilon_d) prob(\hat{\alpha} - \alpha, \varepsilon_\alpha)$
4: Return p_{det}

6 Bayes Filter - Discrete Filters - 08

Implemntation:

- To update the belief upon sensory input and to carry out the normalisation, one has to iterate over all cells of the grid
- Especially when the belief is pealed (which is generally the case during position tracking), one wants to avoid updating irrelevant aspects of the state space
- One approach is to update only sub-spaces of the state space
- This, however, requires monitoring the relevant sub-spaces
- To identify localisation errors, a typical approach is to monitor the likelihood of the observations given the active sub-states
- To efficiently update the belief upon robot motions, one typically assumes a bounded Gaussian model for the motion uncertainty
- This reduced the update cost from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$, where n is the number of states
- The update can also be realised by shifting the data in the grid according to the measured motion
- In a second step, the grid is then converted using a separable Gaussian Kernel

$$\begin{array}{ccc} 1/16 & 1/8 & 1/16 \\ \boxed{1/8} & \boxed{1/4} & \boxed{1/8} \\ 1/16 & 1/8 & 1/16 \end{array} \cong \begin{array}{c} 1/4 \\ \boxed{1/2} \\ 1/4 \end{array} + \begin{array}{ccc} 1/4 & \boxed{1/2} & 1/4 \end{array} \tag{1}$$

  - Fewer arithmetic operations
  - Easier to implement

Tree-based Representation:

Idea: Represent density using a variant of octrees. Efficient in space and time as well as multi-resolution.

7 Bayes Filter - Particle Filter and Monte Carlo Localization - 09

Particle Filter:

- Discrete filter:
  - Discretize the continuous state space
  - High memory complexity
  - Fixed resolution (does not adapt to the belief)
- Particle filters are a way to efficiently represent non-Gaussian distributions
- Basic principle:
  - Set of state hypotheses (*particles*)
  - Survival-of-the-fittest

Mathematical Description:

- Set (actually a multi-set) of weighted samples:

$$S = \left\{ \left\langle \underbrace{s^{[i]}}_{\text{state hypothesis}}, \underbrace{w^{[i]}}_{importance weight} \right\rangle | i = 1, \ldots, N \right\}$$

- The samples represent the posterior

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

$$\delta_{s^{[i]}}(x) = \begin{cases} 1 & \text{if } x = s^{[i]} \\ 0 & otherwise \end{cases}$$

Function Approximation:

- Particle sets can be used to approximate functions
- The more particles fall into an interval, the higher the probability of that interval

Bayes Filter with Particle Sets:

- Measurement update

$$Bel(x) \leftarrow p(z|x)\bar{Bel}(x)$$

$$= p(z|x) \sum_i w_i \delta_{s^{[i]}}(x) = \sum_i p(z|s^{[i]}) w_i \delta_{s^{[i]}}(x)$$

- Motion update

$$\bar{Bel}(x) \leftarrow \int p(x|u, x') Bel(x') \, dx'$$

$$= \int p(x|u, x') \sum_i w_i \delta_{s^{[i]}}(x') \, dx' = \sum_i p(x|u, s^{[i]}) w_i$$

Particle Filter Algorithm:

- 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*

Algorithm 10 particle_filter(S_{t-1}, u_t, z_t)

$S_t = \emptyset$
$\eta = 0$
for $i = 1, \ldots, N$ **do**
    Sample index $j(i)$ from the discrete distribution given by w_{t-1}
    Sample x_t^i from $p(x_t|x_{t-1}, u_t)$ using $x_{t-1}^{j(i)}$ and u_t
    $w_t^i = p(z_t|x_t^i)$
    $\eta = \eta + w_t^i$
    $S_t = S_t\{< x_t^i, w_t^i >\}$
end for
for $i = 1, \ldots, N$ **do**
    $w_t^i = \frac{w_t^i}{\eta}$
end for
Return S_t

Resampling:

- Given: Set S of weighted samples
- Wanted: Random sample, where the probability of drawing x_i is given by w_i
- Typically done n times with replacement to generate new sample set S'
- Roulette wheel: binary search, $\mathcal{O}(n \log(n))$
- Stochastic universal sampling: Systematic resampling, linear time complexity, $\mathcal{O}(n)$, easy to implement, low variance

Algorithm 11 systematic_resampling(S, n)

$S' = \emptyset$
$c_1 = w^1$
for $i = 2, \ldots, n$ **do**
    $c_i = c_{i-1} + w^i$
end for
$u_1 \sim U]0, n^{-1}], i = 1$
for $j = 1, \ldots, n$ **do**
    while $u_j > c_i$ **do**
        $i = i + 1$
    end while
    $S' = S' \cup \left\{ < x^i, n^{-1} > \right\}$
    $u_{j+1} = u_j + n^{-1}$
end for
Return S'

Mobile Robot Localization:

- Each particle is a potential pose of the robot
- Proposal distribution is the motion model of the robot (prediction step)
- The observation model is used to compute importance weight (correction step

Robot Localization using Particle Filters:

- Each particle is a potential pose of the robot
- The set of weighted particles approximates the posterior belief about the robot's pose (target distribution)
- Particles are drawn from the motion model (proposal distribution)
- Particles are weighted according to the observation model (sensor model)
- Particles are resampled according to the particle weights
- Why use resample:
  - Finite number of particles
  - Without resampling, the filter will likely lose track of the *good* hypotheses
  - Resampling ensures that particles stay in the meaningful area of the state space

Limitations:

- The approach is able:
  - to track the pose of a mobile robot
  - to globally localize the robot
- Deal with localization errors like the kidnapped robot problem:
  - Randomly insert a fixed number of samples with randomly chosen poses
  - Allows considering that the robot can be teleported at any point in time to an arbitrary location
  - Another option is to insert the samples inverse proportional to the average likelihood of the observations (the lower this likelihood, the higher the probability that the current estimation is wrong)

8 Bayes Filter - Kalman Filter - 10

Kalman Filter:

- Bayes filter with Gaussians
- Developed in the late 1950s
- Most relevant Bayes filter variant in practice
- Applications range from economics, weather forecasting, and satellite navigation to robotics and many more

Properties of Gaussians:

We stay Gaussian as long as we start with Gaussians and perform only linear transformations.

Discrete Kalman Filter:

Estimates the state x of a discrete-time controlled process that is governed by the linear stochastic difference equation:

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t$$

With a measurement:

$$z_t = C_t x_t + \delta_t$$

Components of a Kalman Filter:

- A_t Matrix $(n \times n)$ that describes how the state evolves from $t-1$ to t without controls or noise
- B_t Matrix $(n \times l)$ that describes how the control u_T changes the state from $t-1$ to t
- C_t Matrix $(k \times n)$ that describes how map the state x_t to an observation z_t
- ε_t δ_t Random variables representing the process and measurement noise that are assumed to be independent and normally distributed with covariance Q_t and R_t repectively

Linear Gaussian Systems:

Initialization: Initial belief is normally distributed:

$$Bel(x_0) = N(x_0; \mu_0, \Sigma_0)$$

Dynamics: Dynamics are linear functions of the state and the control plus additive noise, from which we get:

$$p(x_t|u_t, x_{t-1}) = N(x_t; A_t x_{t-1} + B_t \mu_t; Q_t)$$

$$\bar{\mu}_t = A_t \mu_{t-1} + B_t \mu_t$$

$$\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + Q_t$$

Observations: Observations are a linear function of the state plus additive noise, from which we get:

$$p(z_t|x_t) = N(z_t; C_t x_t; R_t)$$

$$\mu_t = \bar{\mu}_t + K_t(z_t - C_t \bar{\mu}_t)$$

$$\Sigma_t = (I - K_t C_t)\bar{\Sigma}_t$$

with $K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + R_t)^{-1}$

Algorithm 12 kalman_filter($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$)

1: Prediction:
2: $\bar{\mu}_t = A_t \mu_{t-1} + B_t \mu_t$
3: $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + Q_t$
4: Correction:
5: $K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + R_t)^{-1}$
6: $\mu_t = \bar{\mu}_t + K_t(z_t - C_t \bar{\mu}_t)$
7: $\Sigma_t = (I - K_t C_t)\bar{\Sigma}_t$
8: Return μ_t, Σ_t

Kalman Filter Characteristics:

- Only two parameters describe belief about the state of the system
- Highly efficient: Polynomial in the measurement dimensionality k and state dimensionality n: $\mathcal{O}(k^{2.376} + n^2)$
- Optimal for linear Gaussian systems
- However: most robotics systems are nonlinear
- Can only model unimodal beliefs

9 Bayes Filter - Extended Kalman Filter - 11

Nonlinear Dynamic Systems:

Most realistic problems involve nonlinear functions:

$$\cancel{x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t} \implies x_t = g(u_t, x_{t-1})$$

$$\cancel{z_t = C_t x_t + \delta_t} \implies z_t = h(x_t)$$

Non-Gaussian Distributions:

- The non-linear functions lead to non-Gaussian distributions
- Kalman filter is not applicable anymore
- Solve this with local linearization

EKF Linearization: First Order Taylor Expansion:

- Prediction:

$$g(u_t, x_{t-1}) \cup g(u_t, \mu_{t-1}) + G_t(x_t - \mu_{t-1})$$

- Correction:

$$h(x_t) \cup h(\bar{\mu}_t) + H_t(x_t - \bar{\mu}_t)$$

- Where G_t and H_t are jacobian matrices

Jacobian Matrix:

- In general, a $n \times m$ non-square matrix
- Given a vector-valued function $f(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_m(x) \end{bmatrix}$, the jacobian matrix is defined as:

$$F_x = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

- It is the orientation of the tangent plane to the vector-valued function at a given point
- Generalizes the gradient of a scalar-valued function

Algorithm 13 extended_kalman_filter$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$

1: Prediction:
2: $\bar{\mu}_t = g(u_t, \mu_{t-1})$
3: $\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + Q_t$
4: Correction:
5: $K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + R_t)^{-1}$
6: $\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$
7: $\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$
8: Return μ_t, Σ_t

EKF Characteristics:

- The EKF is an ad-hoc solution to deal with non-linearities
- It performs local linearization in each step
- It works well in practice for moderate non-linearities (ex: landmark localization)
- There exist better ways for dealing with non-linearities, such as the unscented Kalman filter, called UKF
- Unile the KF, the EKF, in general, is not an optimal estimator
- It is optimal if the measurement and the motion models are both linear, in which case the ELF reduces to the KF

10 Grid Maps and Mapping with Known Poses - 12

The General Problem of Mapping:

- Formally, mapping involves, given the sensor data $d = \{u_1, z_1, \ldots, u_t, z_t\}$, to calculate the most likely map $m^* = \text{argmax}_m P(m|d)$
- Formally, mapping with known poses involves given the measurements and the poses, $m^* = \text{argmax}_m P(m|z_1, \ldots, z_t, x_1, \ldots, x$ to calculate the most likely map

Grid Maps:

- We discretize the world into cells
- The grid structure is rigid
- Each cell is assumed to be occupied or free
- It is a non-parametric model
- It does not rely on a feature detector
- It requires substantial memory resources

Occupancy Probability:

- Each cell is a binary random variable that models the occupancy
- Cell is occupied: $p(m_i) = 1$
- Cell is not occupied $p(m_i) = 0$
- No information $p(m_i) = 0.5$
- The environment is assumed to be static
- The cells (the random variables) are independent of each other

Representation: The product of the probability distributions of the individual cells gives the probability distribution of the map:

$$\underbrace{p(m)}_{\text{map - four-dimensional vector}} = \underbrace{\prod_i p(m_i)}_{\text{cells - four independent cells}}$$

Estimating a Map from Data: Given sensor data $z_{1:t}$ and the poses $x_{1:t}$ of the sensor, estimate the map:

$$p(m) = \prod_i \underbrace{p(m_i|z_{1:t}, x_{1:t})}_{\text{binary random variable}}$$

$\Longrightarrow$Binary Bayes filter (for a static state)

Static State Binary Bayes Filter

$$
\begin{aligned}
p(m_i|z_{1:t}, x_{1:t}) &\overset{\text{Bayes rule}}{=} \frac{p(z_t|m_i, z_{1:t-1}, x_{1:t})p(m_i|z_{1:t-1}, x_{1:t})}{p(z_t|z_{1:t-1}, x_{1:t})} \\
&\overset{\text{Markov}}{=} \frac{p(z_t|m_i, x_t)p(m_i|z_{1:t-1}, x_{1:t-1})}{p(z_t|z_{1:t-1}, x_{1:t})} \\
&\overset{\text{Bayes rule}}{=} \frac{p(m_i|z_t, x_t)p(z_t|x_t)p(m_i|z_{1:t-1}, x_{1:t-1})}{p(m_i|x_t)p(z_t|z_{1:t-1}, x_{1:t})} \\
&\overset{\text{Markov}}{=} \frac{p(m_i|z_t, x_t)p(z_t|x_t)p(m_i|z_{1:t-1}, x_{1:t-1})}{p(m_i)p(z_t|z_{1:t-1}, x_{1:t})}
\end{aligned}
$$

Do exactly the same for the opposite:

$$
p(\neg m_i|z_{1:t}, x_{1:t}) \overset{\text{the same}}{=} \frac{p(\neg m_i|z_t, x_t)p(z_t|x_t)p(\neg m_i|z_{1:t-1}, x_{1:t-1})}{p(\neg m_i)p(z_t|z_{1:t-1}, x_{1:t})}
$$

By computing the ratio of both probabilities, we obtain:

$$
\begin{aligned}
\frac{p(m_i|z_{1:t}, x_{1:t})}{p(\neg m_i|z_{1:t}, x_{1:t})} &= \frac{\frac{p(m_i|z_t,x_t)\cancel{p(z_t|x_t)}p(m_i|z_{1:t-1},x_{1:t-1})}{p(m_i)\cancel{p(z_t|z_{1:t-1},x_{1:t})}}}{\frac{p(\neg m_i|z_t,x_t)\cancel{p(z_t|x_t)}p(\neg m_i|z_{1:t-1},x_{1:t-1})}{p(\neg m_i)\cancel{p(z_t|z_{1:t-1},x_{1:t})})}} \\
&= \frac{p(m_i|z_t, x_t)p(m_i|z_{1:t-1}, x_{1:t-1})p(\neg m_i)}{p(\neg m_i|z_t, x_t)p(\neg m_i|z_{1:t-1}, x_{1:t-1})p(m_i)} \\
&= \underbrace{\frac{p(m_i|z_t, x_t)}{1-p(m_i|z_t, x_t)}}_{\text{uses } z_t} \underbrace{\frac{p(m_i|z_{1:t-1}, x_{1:t-1})}{1-p(m_i|z_{1:t-1}, x_{1:t-1})}}_{\text{recursive term}} \underbrace{\frac{1-p(m_i)}{p(m_i)}}_{\text{prior}}
\end{aligned}
$$

Also written as:

$$
Bel(m_t^i) = \Big[1 + \frac{1-p(m_i|z_t, x_t)}{p(m_i|z_t, x_t)} \frac{1-Bel(m_t^i)}{Bel(m_t^i)} \frac{p(m_i)}{1-p(m_i)}\Big]^{-1}
$$

Log Odds Notation:

- Log odds ratio is defined as:

$$
l(x) = \log \frac{p(x)}{1-p(x)}
$$

- With the ability to retrieve $p(x)$

$$
p(x) = 1 - \frac{1}{1 + exp(l(X))}
$$

Occupancy Mapping in Log Odds Form:

- The product turns into a sum

$$
l(m_i|z_{1:t}, x_{1:t}) =
$$

- Also written as:

$$
l_{t,i} = inv\_sensor\_model(m_i, x_t, z_t) + l_{t-1,i} - l_0
$$

Algorithm 14 occupancy_grid_mapping($\{l_{t-1,i}\}, x_t, z_t$)

1: **for** all cells m_i **do**
2: **if** m_i in perceptual field of z_t **then**
3: $l_{t,i} = inv\_sensor\_model(m_i, x_t, z_t) + l_{t-1,i} - l_0$
4: **else**
5: $l_{t,i} = l_{t-1,i}$
6: **end if**
7: **end for**
8: Return $\{l_{t,i}\}$

Occupancy Grid Mapping:

- Developed in the mid 80's by Moravec and Elfes
- Originally developed for noisy sonars
- Also called *mapping with known poses*

Alternative: Counting Model:

- For every cell count:
  - hits(x, y): number of cases where a beam ended at $< x, y >$
  - misses(x, y): number of cases where a beam passed through $< x, y >$

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

- Value of interest: $P(reflects(x, y))$

The Measurement Model:

- Pose at time $t : x_t$
- Beam n of scan at time $t : z_{t,n}$
- Maximum range reading: $\zeta_{t,n} = 1$
- Beam reflected by an object: $\zeta_{t,n} = 0$
- Max range: *first $z_{t,n}$(measured distance in #cells) -1 cells covered by the beam must be free*
- Otherwise: *last cell reflected beam, all other free*

$$p(z_t|x_t, m) = \begin{cases} \prod_{k=0}^{z_{t,n}-1}(1 - m_{f(x_t,n,k)}) & \text{if } \zeta_{t,n} = 1 \\ m_{f(x_t,n,z_{t,n})} \prod_{k=0}^{z_{t,n}-1}(1 - m_{f(x_t,n,k)}) & \text{if } \zeta_{t,n} = 0 \end{cases}$$

Computing the Most Likely Map:

- Compute values for m that maximize:

$$m^* = \text{argmax}_m P(m|z_1, \ldots, z_t, x_1, \ldots, x_t)$$

- Assuming a uniform prior probability for $P(m)$, this is equivalent to maximizing:

$$m^* = \text{argmax}_m P(z_1, \ldots, z_t|m, x_1, \ldots, x_t)$$

$$= \text{argmax}_m \prod_{t=1}^{T} P(z_t|m, x_t) \text{ since } z_t \text{ independent and only depend on } x_t$$

$$= \text{argmax}_m \sum_{t=1}^{T} \ln P(z_t|m, x_t)$$

$$= \text{argmax}_m \overbrace{\sum_{j=1}^{J}}^{\text{cells}} \sum_{t=1}^{T} \overbrace{\sum_{n=1}^{N}}^{\text{beams}} \Big(\overbrace{I(f(x_t, n, z_{t,n}) = j)}^{\text{beam } n \text{ end in cell } j} \cdot (1 - \zeta_{t,n}) \cdot \ln(m_j) + \sum_{k=0}^{z_{t,n}-1} \overbrace{I(f(x_t, n, k) = j)}^{\text{beam } n \text{ traversed cell } j} \cdot \ln(1 - m_j) \Big)$$

- Define:

$$\alpha_j = \sum_{t=1}^{T} \sum_{n=1}^{N} I(f(x_t, n, z_{t,n}) = j) \cdot (1 - \zeta_{t,n})$$

$$\beta_j = \sum_{t=1}^{T} \sum_{n=1}^{N} \sum_{k=0}^{z_{t,n}-1} I(f(x_t, n, k) = j)$$

Meaning of α_j and β_j:

α_j corresponds to the number of times a beam that is not a maximum range beam ended in cell j ($hits(j)$)

β_j corresponds to the number of times a beam traversed cell j without ending in it ($misses(j)$)

Meaning we can write:

$$m^* = \text{argmax}_m \sum_{j=1}^{J} \Big(\alpha_j \ln(m_j) + \beta_j \ln(1 - m_j) \Big)$$

As the m_j's are independent of each other we can maximize this sum by maximizing it for every j.

If we ser $\frac{\partial}{\partial m_j} = \frac{\alpha_j}{m_j} - \frac{\beta_j}{1-m_j} = 0$ we obtain $m_j = \frac{\alpha_j}{\alpha_j + \beta_j}$

Computing the most likely map reduces to counting how often a cell has reflected a measurement and how often a cell was traversed by a beam.

Difference between Occupancy Grid Maps and Counting:

- The counting model determines how often a cell reflects a beam
- The occupancy model represents whether or not a cell is occupied by an object
- Although a cell might be occupied by an object, the reflection probability of this object might be very small

11 SLAM: Simultaneous Localization and Mapping - 13

What is SLAM?

- Estimate the pose of a robot and the map of the environment at the same time
- SLAM is hard, because:
  - a map is needed for localization
  - a good pose estimate is needed for mapping
- **Localization**: inferring location given a map
- **Mapping**: inferring a map given localizations
- **SLAM**: learning a map and localizing the robot simultaneously

Different Variants of the SLAM Problem:

- Depending on the representation
  - Feature-based (landmarks or features)
  - Dense (2D/3D grids, signed distance functions, ...)
- Depending on the sensor
  - Vision
  - Proximity sensors
- Depending on the sensor information
  - Range only
  - Bearing only
  - Range and bearing
- Depending on the algorithm
  - Filtering-based
  - Optimization-based

Feature-Based SLAM: Given:

- The robot's controls $U_{1:k} = \{u_1, u_2, \ldots, u_k\}$
- Relative observations $Z_{1:k} = \{z_1, z_2, \ldots, z_k\}$

Wanted:

- Map of features $m = \{m_1, m_2, \ldots, m_n\}$
- Path of the robot $X_{1:k} = \{x_1, x_2, \ldots, x_k\}$

Absolute robot poses and landmark positions.

Relative measurements of landmarks.

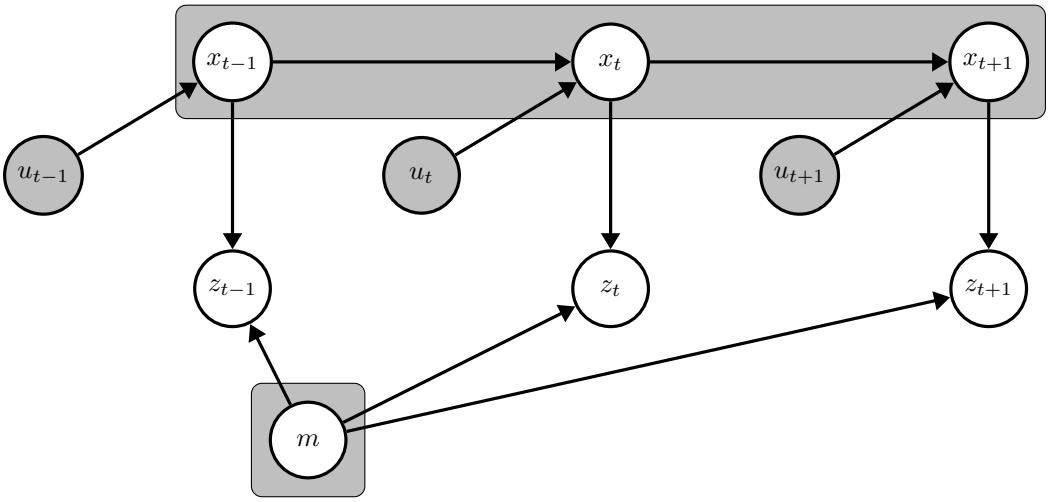
Why SLAM is Hard:

- Robot path and map are both **unknown**
- Errors in map and pose estimates correlated
- The **mapping between observations and landmarks is unknown**
- Picking **wrong** data associations can have **catastrophic** consequences (divergence)

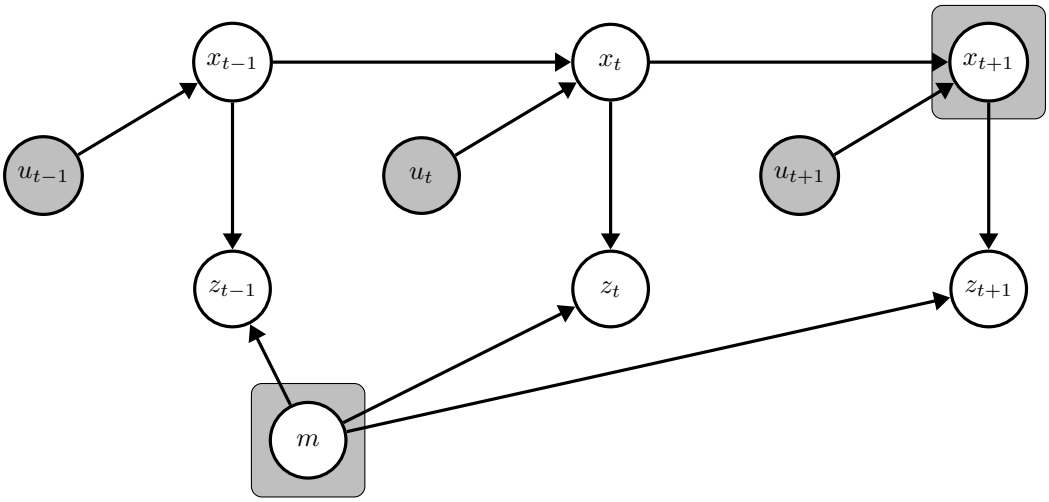
SLAM: Simultaneous Localization and Mapping:

- Full SLAM - $p(x_{0:t}, m|z_{1:t}, u_{1:t})$ - estimates entire path and map
- Online SLAM - $p(x_t, m|z_{1:t}, u_{1:t}) = \int\int\cdots\int p(x_{1:t}, m|z_{1:t}, u_{1:t})\, dx_1\, dx_2\, \ldots\, dx_{t-1}$ - estimates most recent pose and map
- Integrations (marginalization) typically done recursively, one at a time

Graphical Model of Full SLAM:



Graphical Model of Online SLAM:



EKF SLAM: State Representation:

- **Localization**:

3x1 pose vector
3x3 covariance matrix

$$x_k = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} \quad \Sigma_k = \begin{bmatrix} \sigma_x^2 & \sigma_{xy}^2 & \sigma_{x\theta}^2 \\ \sigma_{yx}^2 & \sigma_y^2 & \sigma_{y\theta}^2 \\ \sigma_{\theta x}^2 & \sigma_{\theta y}^2 & \sigma_\theta^2 \end{bmatrix}$$

- **SLAM**: Landmarks **simply extend** the state. **Growing** state vector and covariance matrix.

$$x_k = \begin{bmatrix} x_k \\ m_1 \\ m_2 \\ \vdots \\ m_n \end{bmatrix} \quad \Sigma_k = \begin{bmatrix} \Sigma_R & \Sigma_{RM_1} & \Sigma_{RM_2} & \cdots & \Sigma_{RM_n} \\ \Sigma_{M_1R} & \Sigma_{M_1} & \Sigma_{M_1M_2} & \cdots & \Sigma_{M_1M_n} \\ \Sigma_{M_2R} & \Sigma_{M_2M_1} & \Sigma_{M_2} & \cdots & \Sigma_{M_2M_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \Sigma_{M_nR} & \Sigma_{M_nM_1} & \Sigma_{M_nM_2} & \cdots & \Sigma_{M_n} \end{bmatrix}$$

Map with n landmarks: $(3+2n)$ - dimensional Gaussian, can handle a large number of dimensions:

$$\underbrace{\begin{bmatrix} x \\ y \\ \theta \\ m_{1,x} \\ m_{1,y} \\ \vdots \\ m_{n,x} \\ m_{n,y} \end{bmatrix}}_{\mu} \quad \underbrace{\begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{x\theta} & \sigma_{xm_{1,x}} & \sigma_{xm_{1,y}} & \cdots & \sigma_{xm_{n,x}} & \sigma_{xm_{n,y}} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{y\theta} & \sigma_{ym_{1,x}} & \sigma_{ym_{1,y}} & \cdots & \sigma_{ym_{n,x}} & \sigma_{ym_{n,y}} \\ \sigma_{\theta x} & \sigma_{\theta y} & \sigma_{\theta\theta} & \sigma_{\theta m_{1,x}} & \sigma_{\theta m_{1,y}} & \cdots & \sigma_{\theta m_{n,x}} & \sigma_{\theta m_{n,y}} \\ \sigma_{m_{1,x}x} & \sigma_{m_{1,x}y} & \sigma_{\theta} & \sigma_{m_{1,x}m_{1,x}} & \sigma_{m_{1,x}m_{1,y}} & \cdots & \sigma_{m_{1,x}m_{n,x}} & \sigma_{m_{1,x}m_{n,y}} \\ \sigma_{m_{1,y}x} & \sigma_{m_{1,y}y} & \sigma_{\theta} & \sigma_{m_{1,y}m_{1,x}} & \sigma_{m_{1,y}m_{1,y}} & \cdots & \sigma_{m_{1,y}m_{n,x}} & \sigma_{m_{1,y}m_{n,y}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \sigma_{m_{n,x}x} & \sigma_{m_{n,x}y} & \sigma_{\theta} & \sigma_{m_{n,x}m_{1,x}} & \sigma_{m_{n,x}m_{1,y}} & \cdots & \sigma_{m_{n,x}m_{n,x}} & \sigma_{m_{n,x}m_{n,y}} \\ \sigma_{m_{n,y}x} & \sigma_{m_{n,y}y} & \sigma_{\theta} & \sigma_{m_{n,y}m_{1,x}} & \sigma_{m_{n,y}m_{1,y}} & \cdots & \sigma_{m_{n,y}m_{n,x}} & \sigma_{m_{n,y}m_{n,y}} \end{bmatrix}}_{\Sigma}$$

EKF SLAM: Filter Cycle:

- State prediction (odometry)
- Measurement prediction
- Measurement
- Data association
- Update
- Integration of new landmarks

EKF SLAM: State Prediction: Odometry:

$$\hat{x}_R = f(x_r, u)$$
$$\hat{\Sigma}_R = F_x \Sigma_R F_x^T + F_u U F_u^T$$

Robot-landmark cross-covariance prediction: $\hat{\Sigma}_{RM_i} = F_x \Sigma_{RM_i}$

EKF SLAM: Measurement Prediction: Global-to-local frame transform h: $\hat{z}_k = h\hat{x}_k$

EKF SLAM: Obtained Measurement: (x, y)-point landmarks:

$$z_k = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} z \\ z_2 \end{bmatrix}$$
$$R_k = \begin{bmatrix} R_1 & 0 \\ 0 & R_2 \end{bmatrix}$$

EKF SLAM: Data Association: Associates predicted measurements $\hat{z}_k^i$ with observation z_k^i

$$v_k^{ij} = z_k^i - \hat{z}_k^i$$
$$S_k^{ij} = R_k^j + H^i \hat{\Sigma}_k H^{iT}$$

EKF SLAM: Update Step: The usual Kalman filter expressions:

$$K_k = \hat{\Sigma}_k H^T S_k^{-1}$$
$$x_k = \hat{x}_k + K_k v_k$$
$$C_k = (I - K_k H)\hat{\Sigma}_k$$

EKF SLAM: New Landmarks: State augmented by:

$$m_{n+1} = g(x_r, z_j)$$
$$\Sigma_{M_{n+1}} = G_R \Sigma_R G_R^T + G_z R_j G_z^T$$

Cross-covariances:

$$\Sigma_{M_{n+1} M_i} = G_R \Sigma_{RM_i}$$
$$\Sigma_{M_{n+1} R} = G_R \Sigma_R$$

EKF SLAM: Correlations Matter:

- If we neglected cross-correlations ($\Sigma_{RM_i} = 0_{3\times 2}$, $\Sigma_{M_i M_{i+1}} = 0_{2\times 2}$)
- Landmark and robot uncertainties would become overly optimistic
- Data association would fail
- Multiple map entries of the same landmark
- Inconsistent map

SLAM: Loop Closure:

- **Recognizing an already mapped area**, typically after a long exploration path (the robot *closes a loop*)
- Structurally identical to data association, but:
  - high levels of ambiguity
  - possibly useless validation gates
  - environment symmetries
- Typically, uncertainties **collapse** after a loop closure (whether the closure was correct or not)
- By revisiting already mapped areas, uncertainties in robot and landmark estimates can be **reduced**
- This can be exploited when **exploring** an environment for the sake of better maps
- Exploration: the problem of **where to acquire new information**

KF-SLAM Properties (Linear Case):

- The **determinant** of any sub-matrix of the map covariance matrix **decreases monotonically** as successive observations are made
- When a new landmark is initialized, its **uncertainty** is maximal
- Landmark uncertainty **decreases monotonically** with each new observation
- In the limit, the landmark estimates become **fully correlated**
- In the limit, the **covariance** associated with any single landmark location estimate is determined only by the **initial covariance in the vehicle location estimate**

EKF-SLAM: Complexity:

- Cost per step: quadratic in n, the number of landmarks: $\mathcal{O}(n^2)$
- Total cost to build map with n landmarks: $\mathcal{O}(n^3)$
- Memory consumption: $\mathcal{O}(n^2)$
- Problem: becomes computationally challenging for very large maps
- There exist variants to circumvent these problems

EKF-SLAM Characteristics:

- The first SLAM solution
- Convergence proof for linear Gaussian case
- Can diverge if nonlinearities are large
- Can only deal with a single mode
- Successful in medium-scale scenes with landmarks
- Approximations exists to reduce the computational challenges

Particle Filters for SLAM:

- Feature-based representations
- Grid-based representations
- Represent belief by random samples
- Estimation of non-Gaussian, nonlinear processes
- Sampling Importance Resampling (SIR) principle:
  - Draw the new generation of particles
  - Assign an importance weight to each particle Resample
- Typical application scenarios are tracking, localization, ...

Localization vs. SLAM:

- A particle filter can be used to solve both problems
- Localization: state space $< x, y, \theta >$
- SLAM: state space $< x, y, \theta, map >$
  - for landmark maps $=< l_1, l_2, \ldots, l_m >$
  - for grid maps $=< c_{11}, c_{12}, \ldots, c_{1n}, c_{21}, \ldots, c_{nm} >$
- Problem: the number of particles needed to represent a posterior grows exponentially with the dimension of the state space
- Use dependencies, in the SLAM context, the map depends on the poses of the robot and we know how to build a map given the position of the sensor is known

Factored Posterior (landmarks):

The SLAM posterior equals the robot path posterior times the landmark positions:

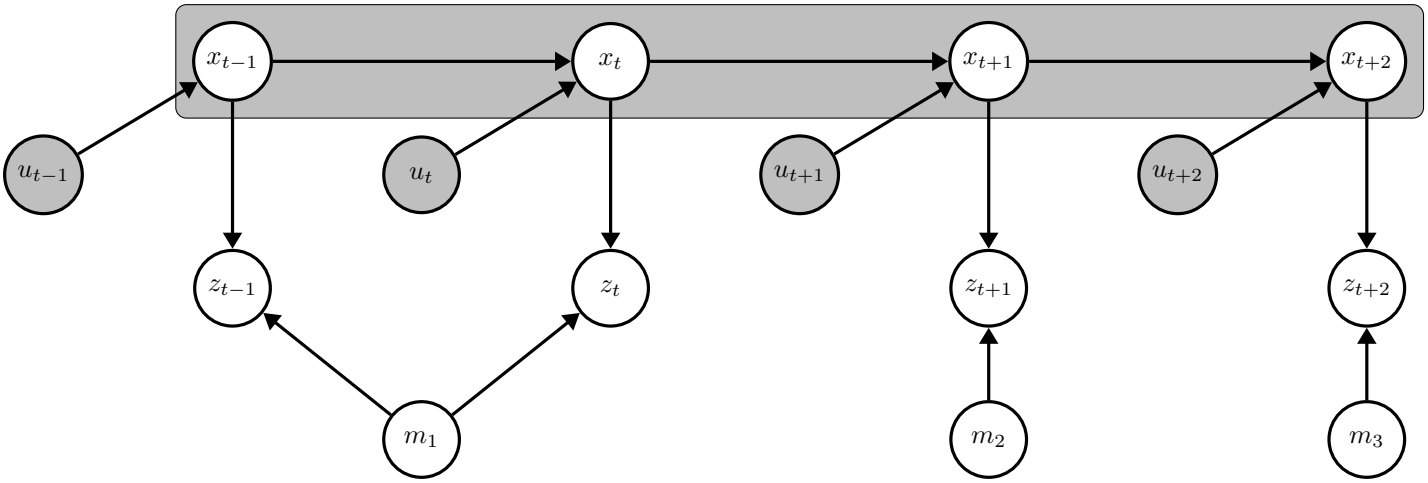
$$p(x_{1:t}, l_{1:m} | z_{1:t}, u_{0:t-1}) = p(x_{1:t} | z_{1:t}, u_{0:t-1}) \cdot p(l_{1:m} | x_{1:t}, z_{1:t})$$

Rao-Blackwellization:

- Factorization to exploit dependencies between variables: $p(a, b) = p(a) \cdot p(b|a)$
- If $p(b|a)$ can be computed in close form, represent only $p(a)$ with samples and compute $p(b|a)$ for every sample
- It comes from the Rao-Blackwell theorem

Revisit the Graphical Model:

The state are known, and the landmarks are conditionally independent given the poses, meaning they are disconnected given the robot's path.



This means we can update the factored posterior such that:

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

Given that the second term can be computed efficiently, particle filtering becomes possible.

FastSLAM:

- Rao-Blackwellized particle filtering based on landmarks
- Each landmark is presented by a 2×2 EKF
- Each particle therefore has to maintain M EKFs

$$\begin{aligned} &\text{Particle \# 1} : \{x, y, \theta, l_1, \ldots, l_M\} \\ &\text{Particle \# 2} : \{x, y, \theta, l_1, \ldots, l_M\} \\ &\qquad\vdots \\ &\text{Particle \# n} : \{x, y, \theta, l_1, \ldots, l_M\} \end{aligned}$$

FastSLAM Complexity:

- Naive:
  - Update robot particles based on the control: $\mathcal{O}(N)$
  - Incorporate an observation into the Kalman filters (given the data association): $\mathcal{O}(N)$
  - Resampl particle set with N the number of particles and M the number of map features: $\frac{\mathcal{O}(NM)}{\mathcal{O}(NM)}$
- Sharing of map features between particles:
  - Update robot particles based on the control: $\mathcal{O}(N)$
  - Incorporate an observation into the Kalman filters (given the data association): $\mathcal{O}(N \log(M))$
  - Resampl particle set with N the number of particles and M the number of map features: $\frac{\mathcal{O}(N \log(M))}{\mathcal{O}(N \log(M))}$

Data Association Problem:

- Sometimes it is hard to decided which observation belong to which landmark
- A robust SLAm solution must consider possible data associations
- Potential data associations depend also on the pose of the robot

Multi-Hypothesis Data Association:

- Data association is done on a per-particle basis
- Robot pose error is factored out of data association decisions

Per-Particle Data Association:

- Two options for per-particle data association
  - Pick the most likely match
  - Pick association randomly with probability proportional to the observation likelihood
- if the data association probability is too low, generate a new landmark

FastSLAM Characteristics:

- FastSLAM shows high robustness to motion errors when compared to EKF
- Factors the SLAM posterior into low-dimensional estimation problems - Scales to problems with over 1 million features
- Factors robot pose uncertainty out of the data association problem - Robust to significant ambiguity in data association and allows data association decisions to be delayed until unambiguous evidence is collected
- Advantages compared to the classical EKF approach (especially with non-linearities)
- Complexity $\mathcal{O}(N \log(M))$

Grid.based SLAM:

- Approach when no pre-defined landmarks are available
- Use the ideas of FastSLAM to build grid maps
- As with landmarks, the map depends on the poses of the robot during data acquisition
- If the poses are known, grid-based mapping is easy (*mapping with known poses*)

Mapping with Rao-Blackwellized Particle Filters:

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

Problems:

- Each map is quite big in case of grid maps
- Each particle maintains its own map; therefore, one needs to keep the number of particles small
- Possible solution is to compute better proposal distributions and improve the pose estimate before applying the particle filter
- When it comes to re-sampling, at each step, it limits the *memory* of our filter.
- Possible solution is to use selective re-sampling

Number of Effective Particles:

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

- Assuming normalized particle weights that sum up to 1:

$$\sum_{i=1}^{n} (w_t^{(i)}) = 1 \Longrightarrow n_{eff} \in [1, n]$$

- Empirical measure of how well the goal distribution is approximated by samples drawn from the proposal
- It describes *the variance of the particle weights*
- It is maximal for equal weights. In this case, the distribution is close to the proposal

Resampling with n_{eff}:

- if our approximation is close to the proposal, no resampling is needed
- We only re-sample when n_{eff} drops below a given threshold, typically $\frac{n}{2}$

FastSLAM with Grid Maps: Characteristics:

- The ideas of FastSLAM can also be applied in the context of grid maps
- Utilizing accurate sensor observation leads to good proposals and highly efficient filters
- It is similar to scan-matching on a per-particle base
- The number of necessary particles and re-sampling steps can be seriously reduced
- Improved versions of grid-based FastSLAM can handle large environments in *real time*