ETH Zürich
Chair of Software Engineering

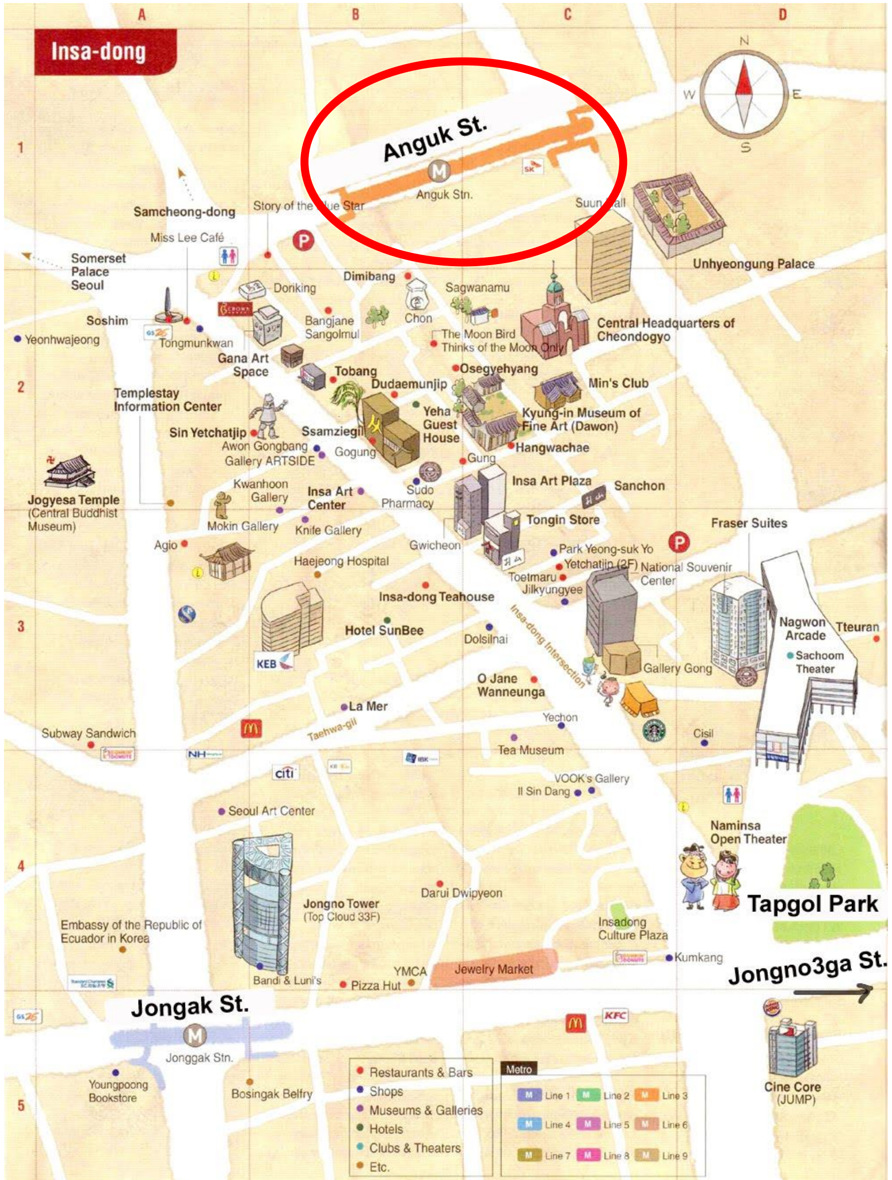
Robotics Programming Laboratory

Bertrand Meyer
Jiwon Shin

Lecture 9: Localization and mapping

This lecture is based on “Probabilistic Robotics” by Thrun, Burgard, and Fox (2005).

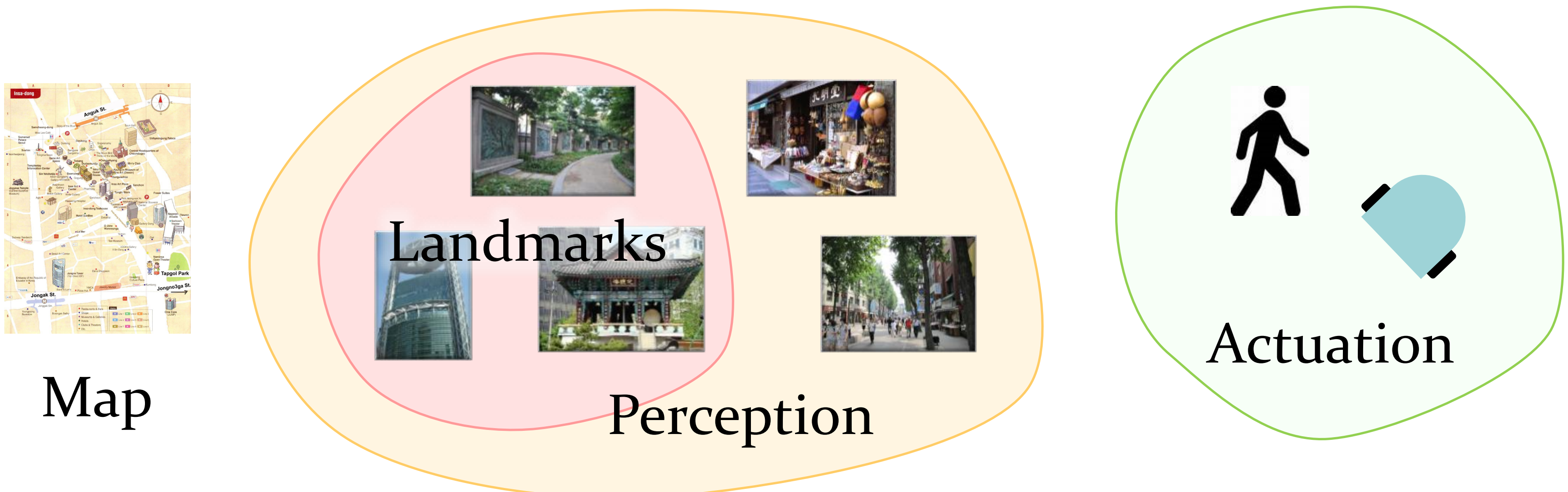
Where am I?



A
B
C
D
1
2
3
4
5
Insa-dong
N
W
E
S
Anguk St.
Anguk Stn.
Story of the Blue Star
Suun Hall
Unhyeongung Palace
Samcheong-dong
Miss Lee Café
Somerset Palace Seoul
Donking
Dimibang
Sagwanamu
Chon
Soshim
Yeonhwajeong
Tongmunkwan
Bangjane Sangolmul
The Moon Bird Thinks of the Moon Only
Central Headquarters of Cheondogyo
Gana Art Space
Tobang
Dudaemunjip
Osegyehyang
Min's Club
Templestay Information Center
Yeha Guest House
Kyung-in Museum of Fine Art (Dawon)
Hangwachae
Sin Yetchatjip
Ssamziegil
Awon Gongbang
Gallery ARTSIDE
Gogung
Gung
Jogyesa Temple
(Central Buddhist Museum)
Kwanhoon Gallery
Insa Art Center
Sudo Pharmacy
Insa Art Plaza
Sanchon
Mokin Gallery
Knife Gallery
Tongin Store
Fraser Suites
Agio
Gwicheon
Park Yeong-suk Yo
Yetchatjip (2F)
National Souvenir Center
Haejeong Hospital
Toetmaru
Jilkyungyee
Insa-dong Teahouse
Insa-dong Intersection
Nagwon Arcade
Tteuran
Sachoom Theater
Hotel SunBee
Dolsilnai
KEB
Gallery Gong
O Jane Wanneunga
La Mer
Yechon
Taehwa-gil
Subway Sandwich
Cisil
Tea Museum
citi
VOOK's Gallery
Il Sin Dang
Seoul Art Center
Naminsa Open Theater
Tapgol Park
Darui Dwipyeon
Jongno Tower
(Top Cloud 33F)
Embassy of the Republic of Ecuador in Korea
Insadong Culture Plaza
Kumkang
YMCA
Jewelry Market
Jongno3ga St.
Bandi & Luni's
Pizza Hut
Jongak St.
Jonggak Stn.
KFC
Youngpoong Bookstore
Bosingak Belfry
Restaurants & Bars
Shops
Museums & Galleries
Hotels
Clubs & Theaters
Etc.
Metro
Line 1
Line 2
Line 3
Line 4
Line 5
Line 6
Line 7
Line 8
Line 9
Cine Core
(JUMP)

Localization

Localization: process of locating an object in space



Dimensions of localization

Type of localization

- Local localization: initial pose is known.
- Global localization: initial pose is unknown.
- Kidnapped robot problem: the robot gets teleported to some location during the operation.

Environments

- Static: the robot is the only moving object.
- Dynamic: other objects change their configuration or location over time.

Approaches

- Passive: the localization module only observes the robot.
- Active: the localization module actively controls the robot to minimize the error and/or the cost of bad localization.

Number of robots

- Single-robot: all data are collected at a single robot platform.
- Multi-robot: communication between the robots can enhance their localization.

Probabilistic robotics

Uncertainty!

- Environment, sensor, actuation, model, algorithm
- Represent uncertainty using the calculus of probability theory

Probability theory

- X: random variable
  - Can take on discrete or continuous values
- $P(X = x)$, $P(x)$: probability of the random variable X taking on a value x
- Properties of $P(x)$
  - $P(X = x) >= 0$
  - $\sum_X P(X = x) = 1$ or $\int_X \; p(X = x) = 1$

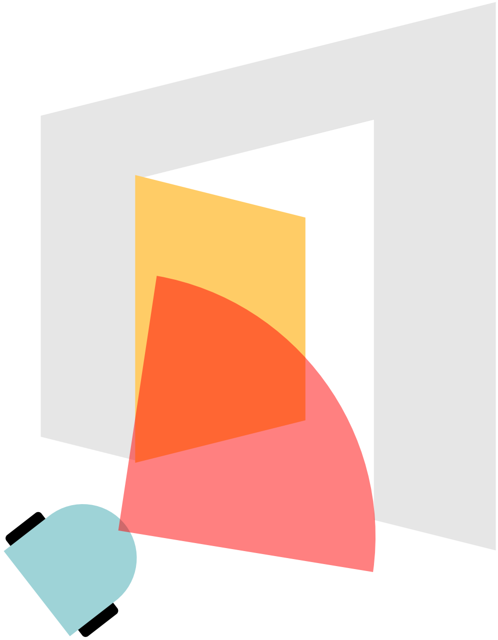
Probability

- $P(x,y)$: joint probability
  - $P(x,y) = P(x)\,P(y)$: X and Y are independent

- $P(x \mid y)$: conditional probability of x given y
  - $P(x \mid y) = p(x)$: X and Y are independent
  - $P(x,y \mid z) = P(x \mid z)\,P(y \mid z)$: conditional independence
  - $P(x \mid y) = P(x,y) \,/\, P(y)$
  - $P(x,y) \;\; = P(x \mid y)\,P(y) = P(y \mid x)\,P(x)$

- $P(x \mid y) = \dfrac{P(y \mid x)\,P(x)}{P(y)} = \dfrac{\text{likelihood} \cdot \text{prior}}{\text{evidence}}$: Bayes' rule
  - $P(y) = \sum_X P(x,y) = \sum_X P(y \mid x)\,P(x)$: Law of total probability

Bayes' rule



$$P(\text{door=open} \mid \text{sensor=far})$$

$$= \frac{P(\text{far} \mid \text{open})\, P(\text{open})}{P(\text{far})}$$

$$= \frac{P(\text{far} \mid \text{open})\, P(\text{open})}{P(\text{far} \mid \text{open})\, P(\text{open}) + P(\text{far} \mid \text{closed})\, P(\text{closed})}$$

Bayes' filter

$bel(x_t) = p(x_t \mid z_{1:t}, u_{1:t})$: belief on the robot's state x_t at time t

Compute robot's state: $bel(x_t)$

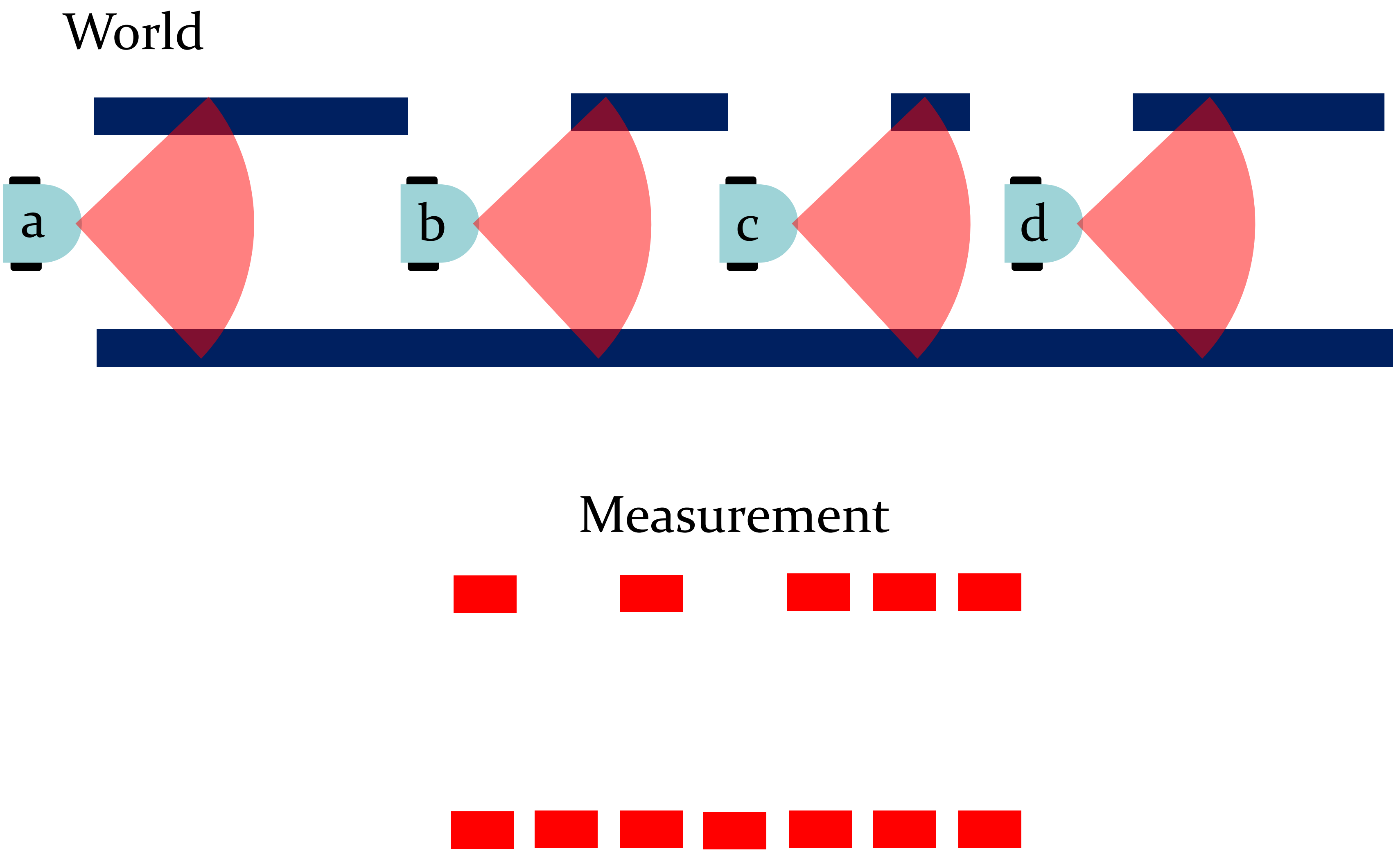
- Predict where the robot should be based on the control u_t

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

- Update the robot state using the measurement z_t

  $$bel(x_t) = \eta\, p(z_t \mid x_{t-1})\, bel^*(x_t)$$

Markov localization

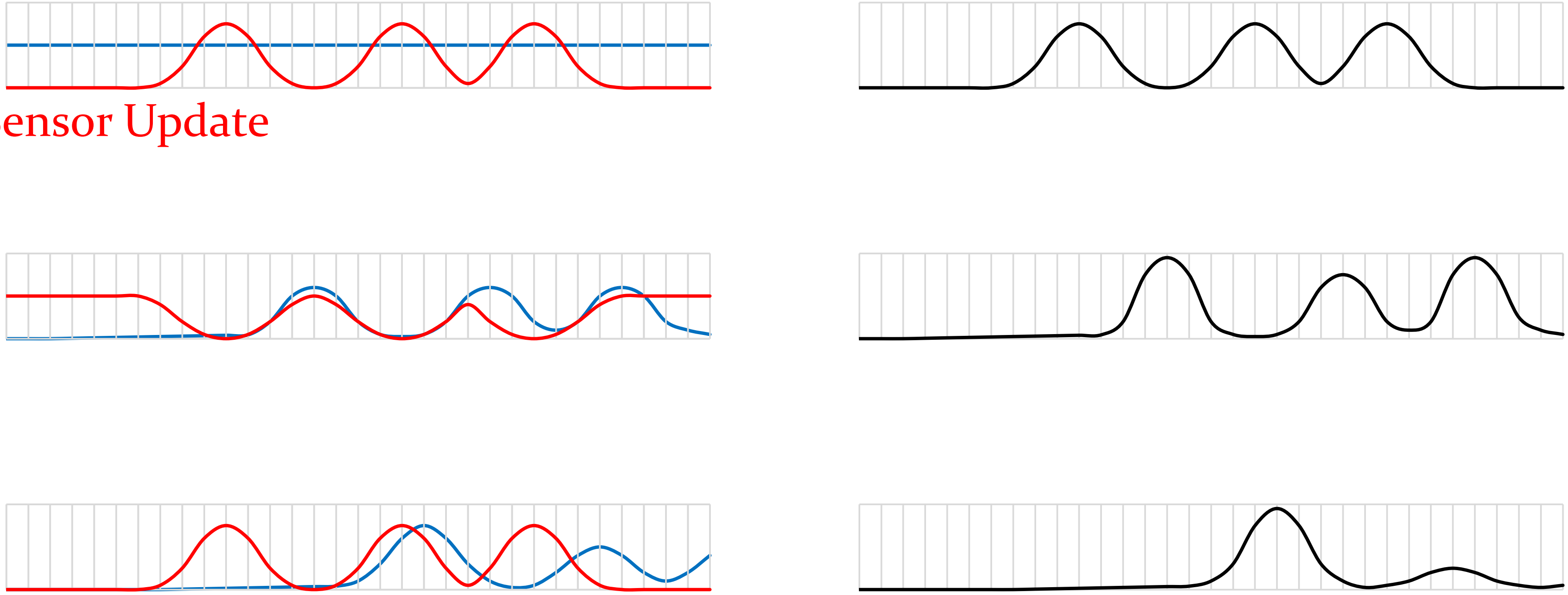


Markov localization

Belief

Motion Update

Sensor Update



Markov localization

Markov_localize ($bel(x_{t-1})$: BELIEF;
u_t: ROBOT_CONTROL;
z_t: SENSOR_MEASUREMENT;
m: MAP) : BELIEF

do

across $bel(x_t)$ **as** x_t **loop**

Motion Update $bel^*(x_t) := \int p(x_t \mid u_t, x_{t-1}, m)\, bel(x_{t-1})\, dx_{t-1}$

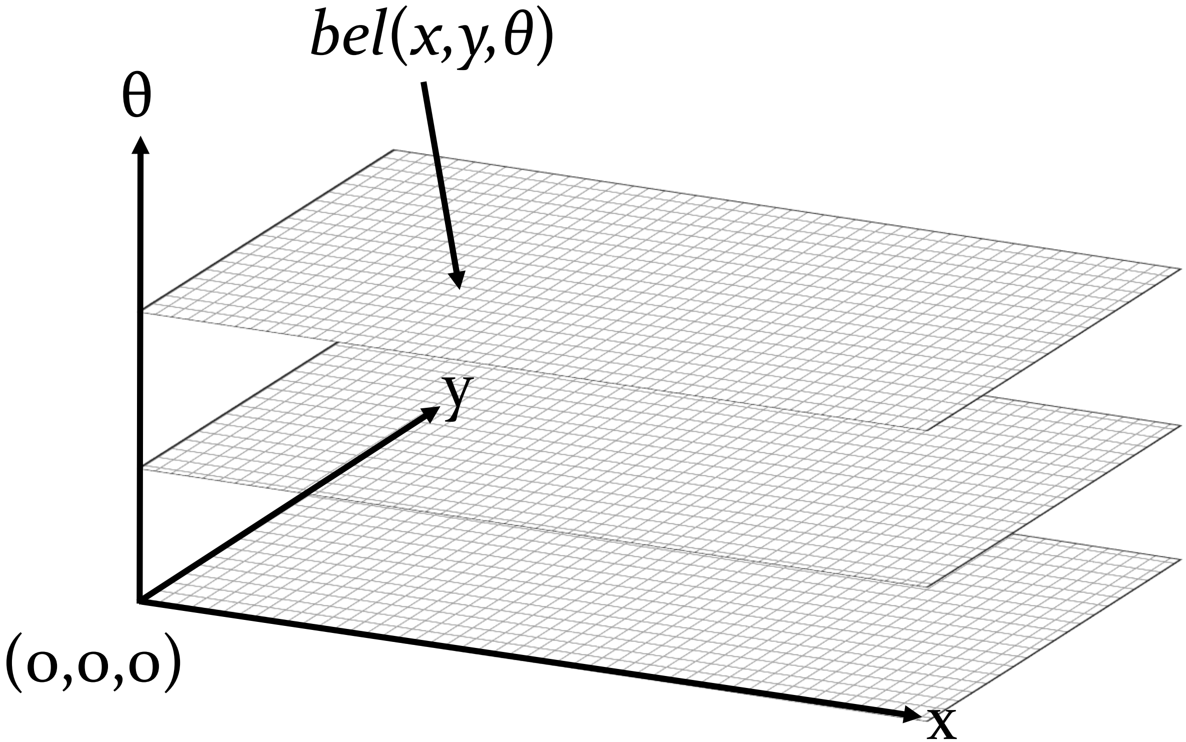
Sensor Update $bel(x_t) := \eta\, p(z_t \mid x_{t-1}, m)\, bel^*(x_t)$

end

Result := $bel^*(x_t)$

end

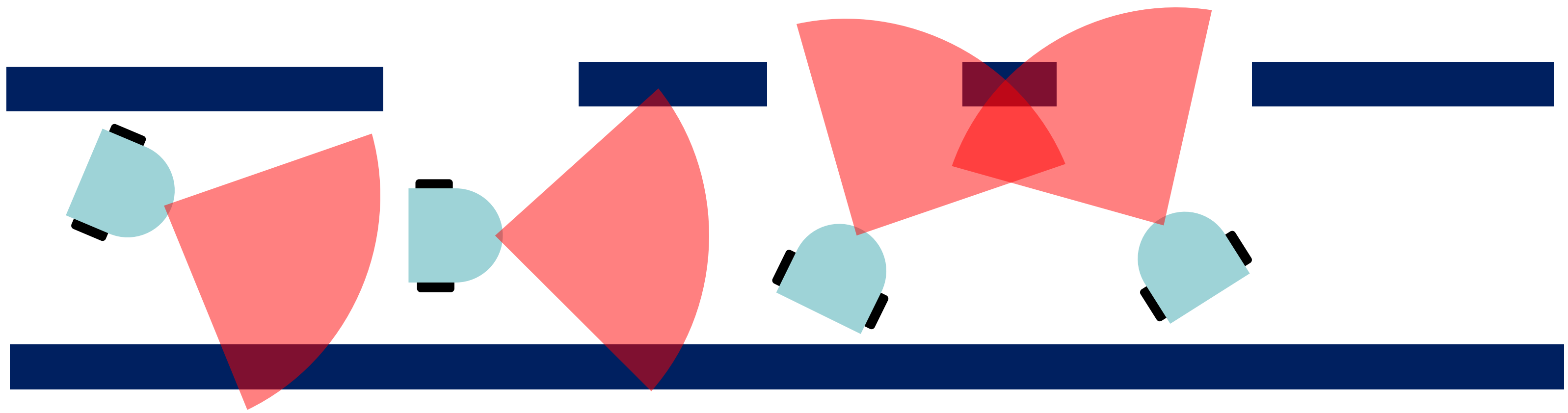
Representation of the robot states



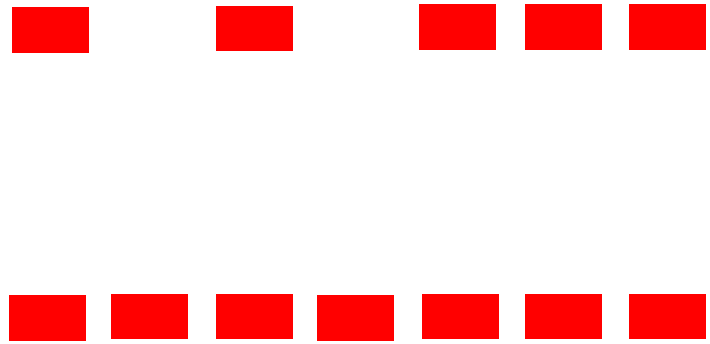
Markov localization

- Can be used for both local localization and global localization
  - If the initial pose (x^*_o) is known: point-mass distribution
    - $bel(x_o) = \begin{cases} 1 & \text{if } x_o = x*_o \\ 0 & \text{otherwise} \end{cases}$
  - If the initial pose (x^*_o) is known with uncertainty Σ: Gaussian distribution with mean at x^*_o and variance Σ
    - $bel(x_o) = \det(2\pi\Sigma)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(x_o - x*_o)^T \Sigma^{-1} (x_o - x*o)\right\}$
  - If the initial pose is unknown: uniform distribution
    - $bel(x_o) = \frac{1}{|X|}$
- Computationally expensive
  - Higher accuracy requires higher grid resolution

What if we keep track of multiple robot poses?



Measurement

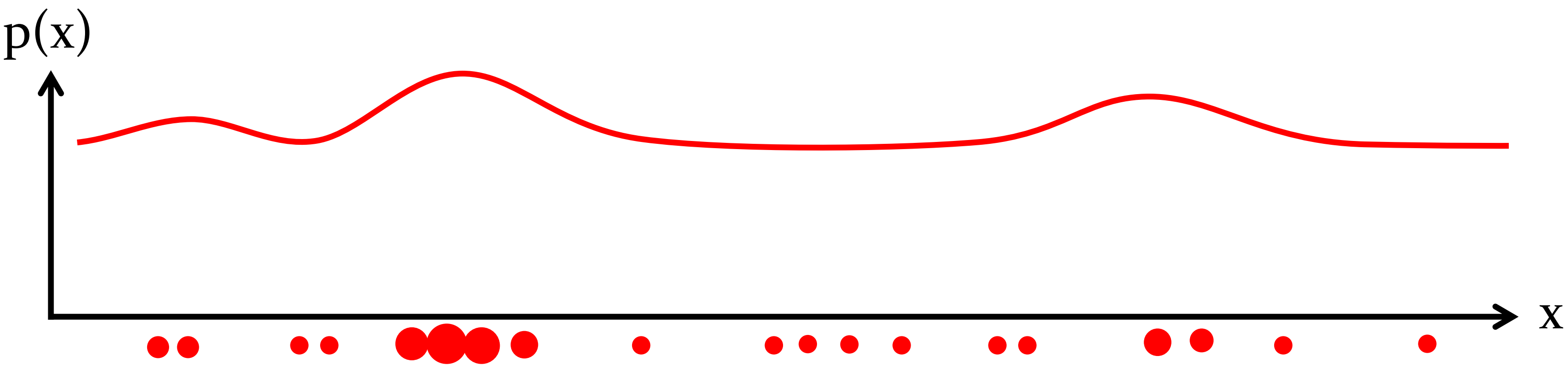


Particle filter

A sample-based Bayes filter

- Approximate the posterior bel(x_t) by a finite number of particles
- Each particle represents the probability of a particular state vector given all previous measurements
- The distribution of state vectors within the particle is representative of the probability distribution function for the state vector given all prior measurements

Importance sampling



Generate samples from a distribution

$$E_f[\ I(x \in A)\ ] = \int f(x)\ I(x \in A)\ dx$$
$$= \int f(x)/g(x)\ g(x)\ I(x \in A)\ dx$$
$$= E_g[\ w(x)\ I(x \in A)\ ]$$

f(x) : target distribution

g(x) : proposal distribution – $f(x) > 0 \rightarrow g(x) > 0$

Particle filter localization

particle_filter_localize (X_{t-1}: PARTICLES;
u_t: ROBOT_CONTROL;
z_t: SENSOR_MEASUREMENT;
m: MAP) : PARTICLES

do

across X_t **as** x_t **loop**

Motion Update x_t.pose := **motion_update**(x_{t-1}, u_t)

Sensor Update x_t.weight := **sensor_update**(x_t, z_t, m)

end

Result := **resample**(X_t)

end

Particle filter localization

- Global localization
  - Track the pose of a mobile robot without knowing the initial pose
- Can handle kidnapped robot problem with little modification
  - Insert some random samples at every iteration
  - Insert random samples proportional to the average likelihood of the particles
- Approximate
  - Accuracy depends the number of samples

Motion models

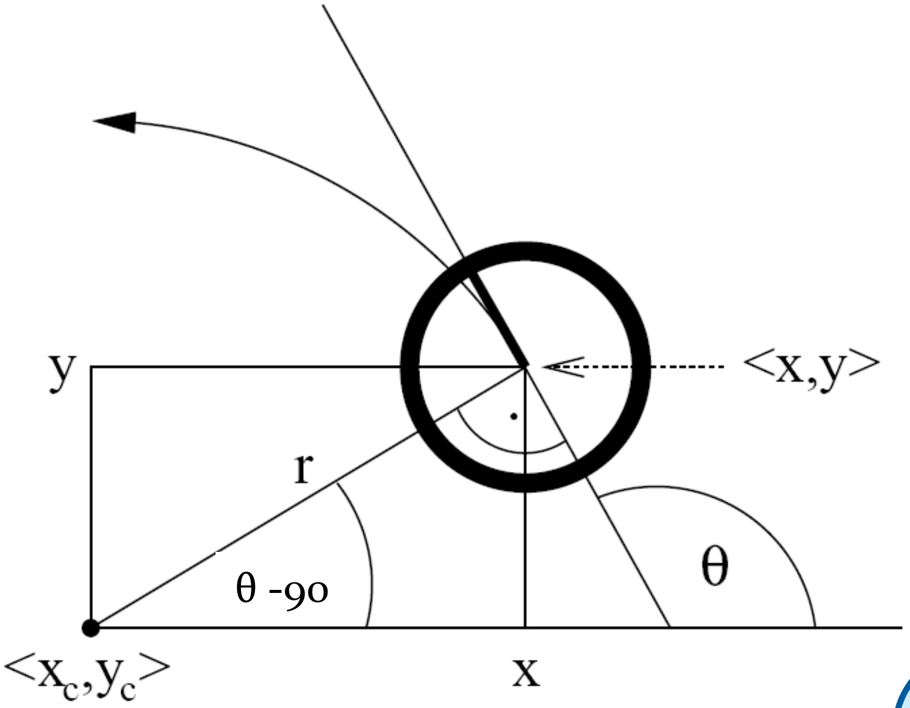
Velocity-based

- No wheel encoders are given.
- New pose is based on the velocities and time elapsed.

Odometry-based

- Systems are equipped with wheel encoders.

Velocity model



$$v = \omega * r \rightarrow r = \left|\frac{v}{\omega}\right|$$

$$x = x_c + \frac{v}{\omega} \sin\theta$$

$$y = y_c - \frac{v}{\omega} \cos\theta$$

$$x' = x_c + \frac{v}{\omega} \sin(\theta + \omega\,\Delta t)$$

$$y' = y_c - \frac{v}{\omega} \cos(\theta + \omega\,\Delta t)$$

$$\theta' = \theta + \omega\,\Delta t$$

$$\hat{v} = v + \varepsilon_{\alpha_1 v^2 + \alpha_2 \omega^2}$$

$$\hat{\omega} = \omega + \varepsilon_{\alpha_3 v^2 + \alpha_4 \omega^2}$$

$$\theta' = \theta + \hat{\omega}\,\Delta t + \varepsilon_{\alpha_5 v^2 + \alpha_6 \omega^2}\,\Delta t$$

Sampling from velocity motion model

sample_motion_model_velocity (x_{t-1} : ROBOT_POSE;
u_t: ROBOT_CONTROL) : ROBOT_POSE

do

$\hat{v} := v +$ **sample** $(\alpha_1 v^2 + \alpha_2 \omega^2)$

$\hat{\omega} := \omega +$ **sample** $(\alpha_3 v^2 + \alpha_4 \omega^2)$

$x' := x - \frac{\hat{v}}{\hat{\omega}} \sin(x.\theta) + \frac{\hat{v}}{\hat{\omega}} \sin(\theta + \hat{\omega}\,\Delta t)$

$y' := y + \frac{\hat{v}}{\hat{\omega}} \cos(x.\theta) - \frac{\hat{v}}{\hat{\omega}} \cos(\theta + \hat{\omega}\,\Delta t)$

$\theta' := \theta + \hat{\omega}\,\Delta t +$ **sample** $(\alpha_5 v^2 + \alpha_6 \omega^2)\,\Delta t$

Result $:= (x', y', \theta')^T$

end

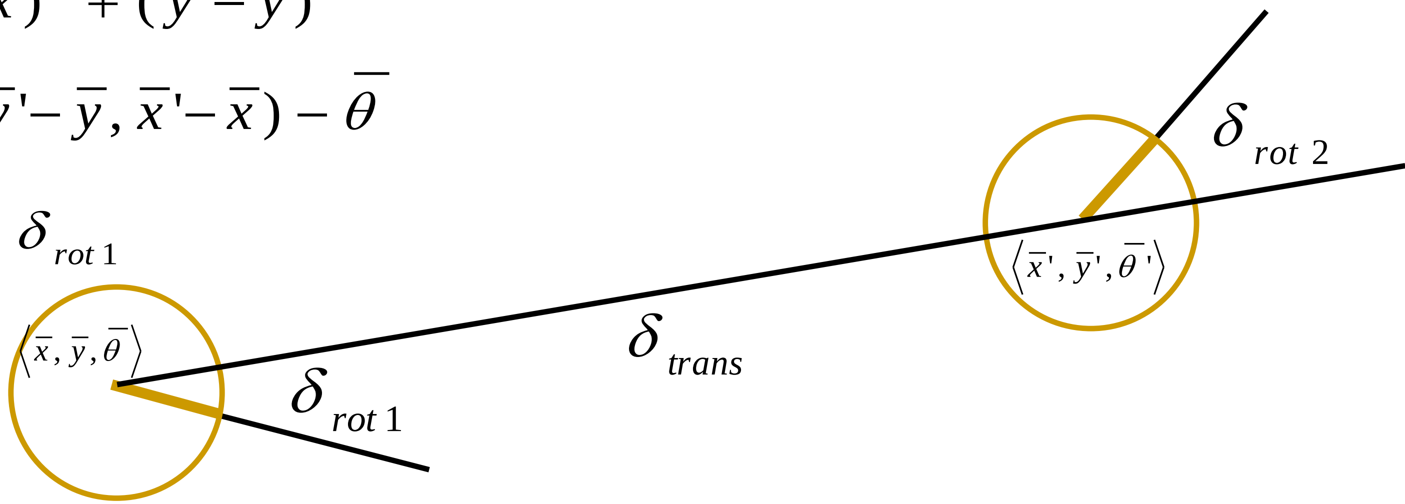
Odometry motion model

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

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

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

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



$$\hat{\delta}_{rot1} = \delta_{rot2} - \varepsilon_{\alpha_1 \delta^2{}_{rot2} + \alpha_2 \delta^2{}_{trans}}$$

$$\hat{\delta}_{trans} = \delta_{trans} - \varepsilon_{\alpha_3 \delta^2{}_{trans} + \alpha_4 \delta^2{}_{rot1} + \alpha_4 \delta^2{}_{rot2}}$$

$$\hat{\delta}_{rot2} = \delta_{rot2} - \varepsilon_{\alpha_1 \delta^2{}_{rot2} + \alpha_2 \delta^2{}_{trans}}$$

Sampling from odometry motion model

sample_motion_model_velocity (x_{t-1}: ROBOT_POSE;
u_t: ROBOT_CONTROL) : ROBOT_POSE

do

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

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

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

$\hat{\delta}_{rot1} := \delta_{rot1} - \textbf{sample}(\alpha_1 \delta^2_{rot1} + \alpha_2 \delta^2_{trans})$

$\hat{\delta}_{trans} := \delta_{trans} - \textbf{sample}(\alpha_3 \delta^2_{trans} + \alpha_4 \delta^2_{rot1} + \alpha_4 \delta^2_{rot2})$

$\hat{\delta}_{rot2} := \delta_{rot2} - \textbf{sample}(\alpha_1 \delta^2_{rot2} + \alpha_2 \delta^2_{trans})$

$x' := x + \hat{\delta}_{trans} \cos(\theta + \hat{\delta}_{rot1})$

$y' := y + \hat{\delta}_{trans} \sin(\theta + \hat{\delta}_{rot1})$

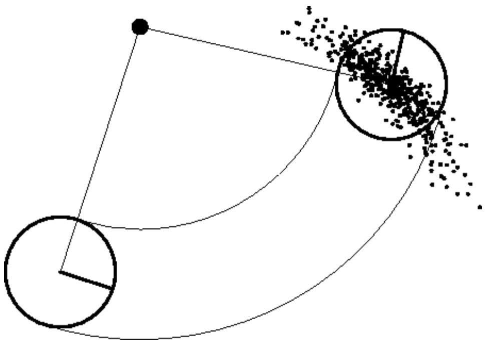
$\theta' := \theta + \hat{\delta}_{rot1} + \hat{\delta}_{rot2}$

Result := $(x', y', \theta')^T$

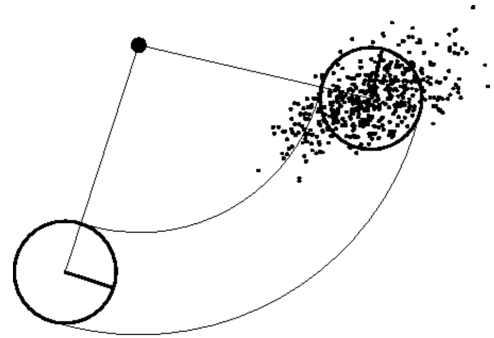
end

Effect of different noise parameter settings

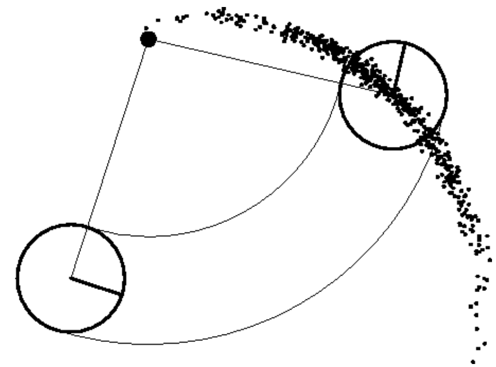
Velocity model



α_1 to α_6: moderate

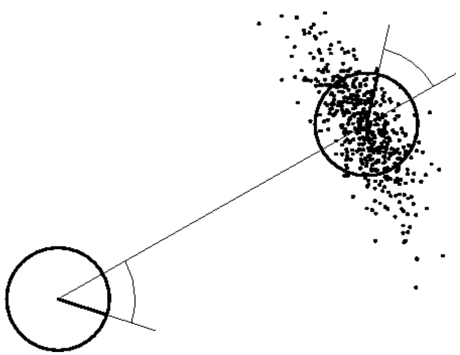


α_3 and α_4: small
α_1 and α_2: large

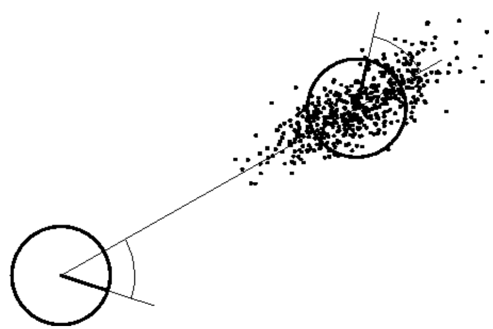


α_3 and α_4: large
α_1 and α_2: small

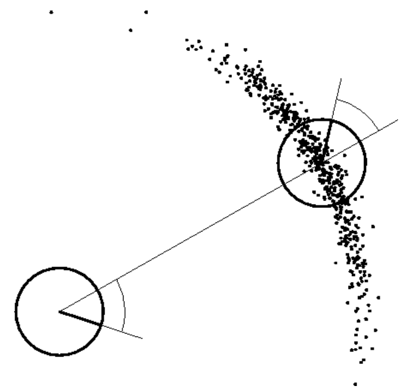
Odometry motion model



α_1 to α_4: moderate



α_1 and α_4: small
α_2 and α_3: large



α_1 and α_4: large
α_2 and α_3: small

Sensor models

Direct modeling of the sensor readings

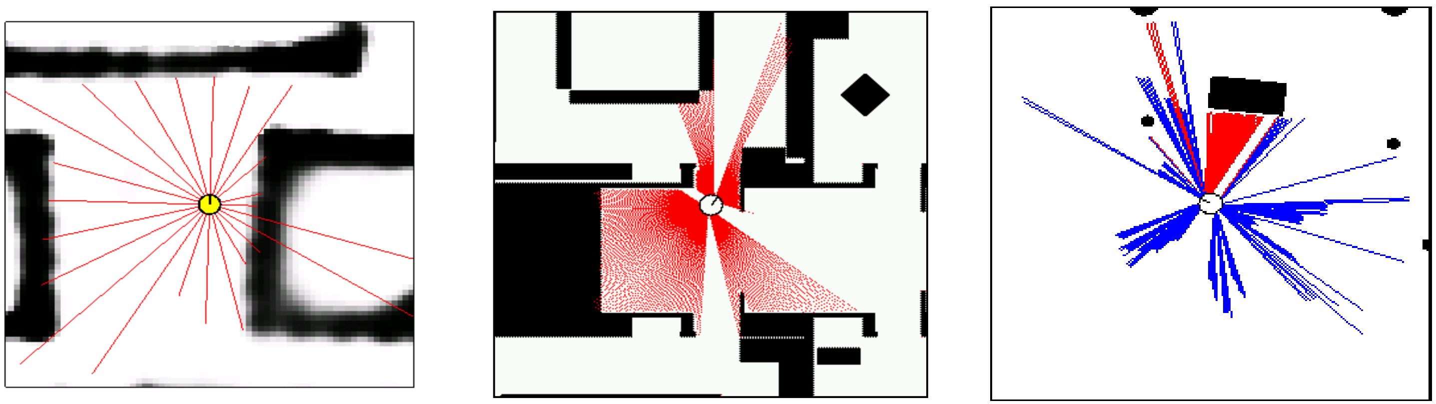
Feature-based models

Beam-based sensor model

Scan z consists of K measurements: $z = \{z_1, z_2, ..., z_K\}$

Individual measurements are independent given the robot position:

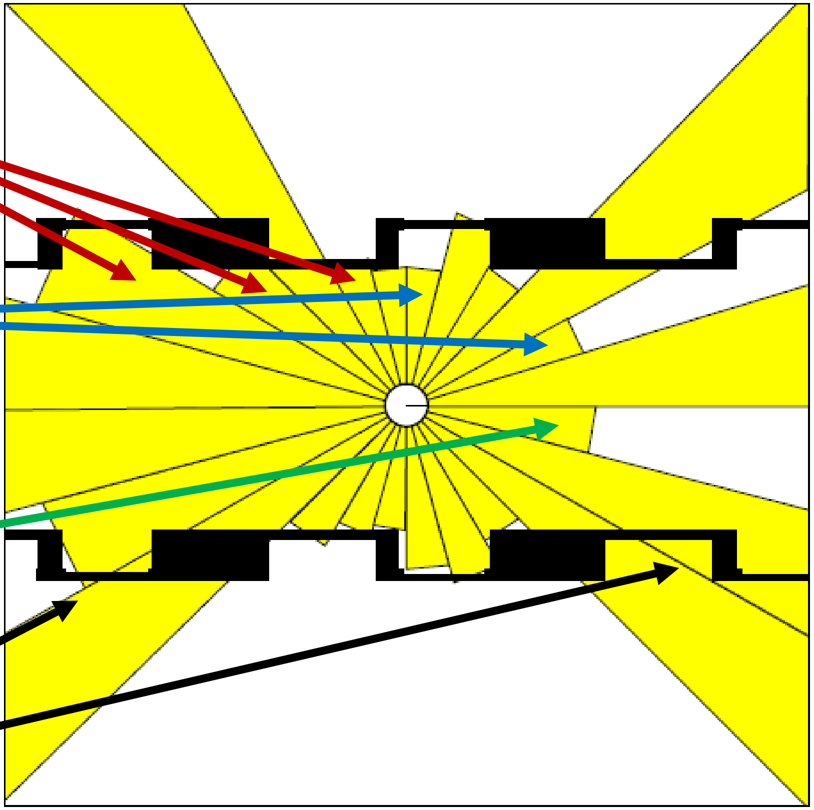
$$P(z \mid x, m) = \prod_{k=1}^{K} P(z_k \mid x, m)$$



Beam-based sensor model

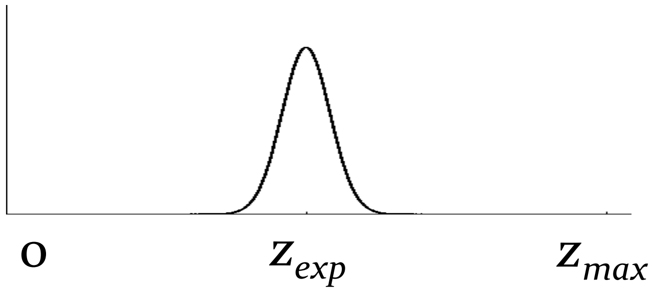
Typical measurement errors

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



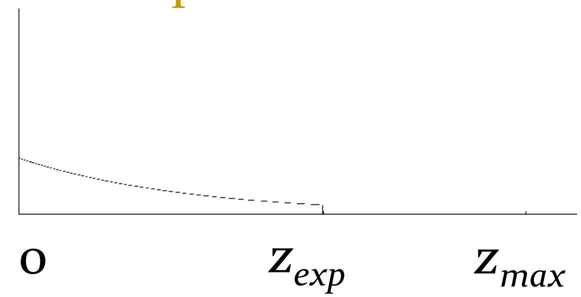
Beam-based sensor model

Measurement noise



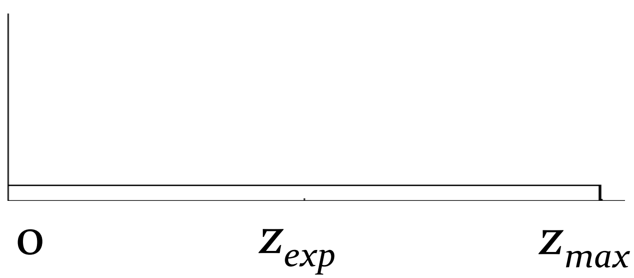
$$P_{hit}(z \mid x, m) = \eta \frac{1}{\sqrt{2\pi b}} e^{-\frac{1}{2}\frac{(z - z_{exp})^2}{b}}$$

Unexpected obstacles



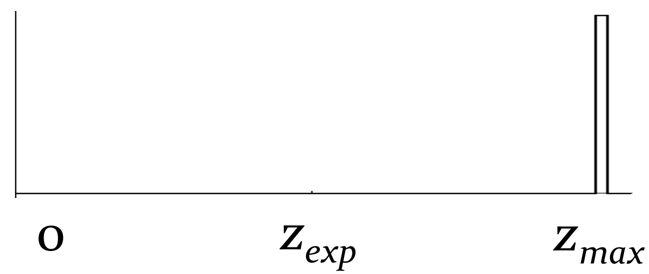
$$P_{unexp}(z \mid x, m) = \begin{cases} \eta \ \lambda \ e^{-\lambda z} & z < z_{exp} \\ 0 & otherwise \end{cases}$$

Random measurement



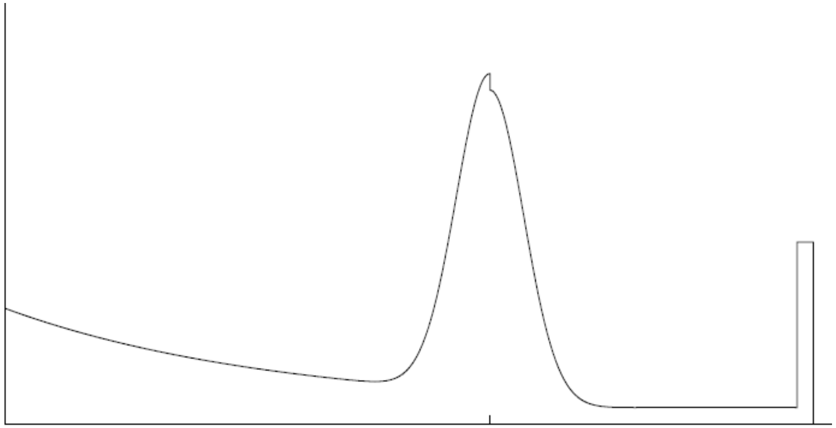
$$P_{rand}(z \mid x, m) = \eta \frac{1}{z_{\max}}$$

Max range



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

Beam-based sensor model



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

Beam-based sensor model

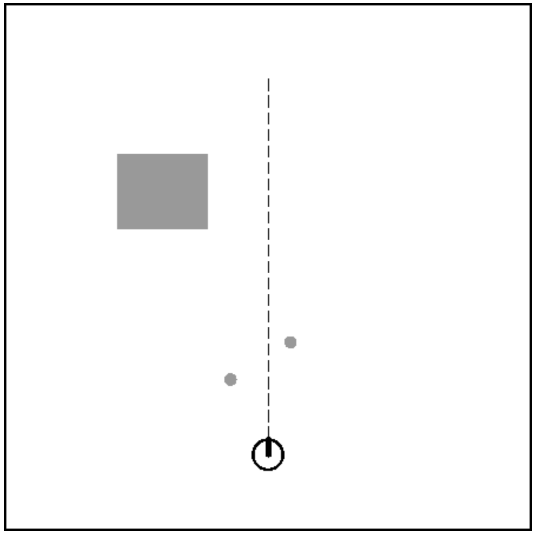
Advantages

- Closely linked to the geometry and physics of range finders

Disadvantages

- Lack of smoothness
- Computationally involved

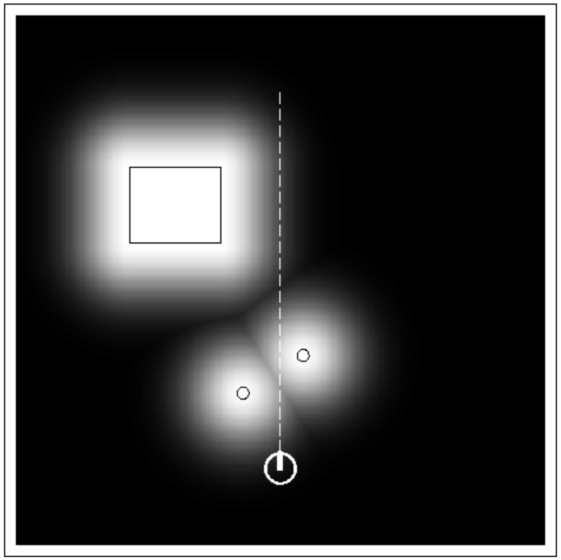
Likelihood fields



Map m

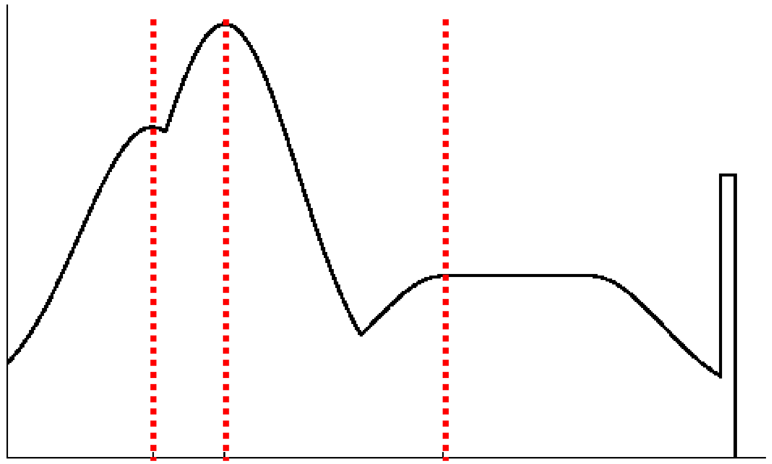
$$p(z^k_t \mid x_t, m) = z_{hit}\, p_{hit} + z_{rand}\, p_{rand} + z_{max}\, p_{max}$$

$z_{hit}, z_{rand}, z_{max}$: mixing weights



Likelihood field

$P(z|x,m)$



Likelihood fields

Project the end points of a sensor scan z_t into the map

- Measurement noise: Zero-centered Gaussian distribution
  - $p_{hit}(z_t^k \mid x_t, m) = \varepsilon_\sigma(\text{dist})$
  - dist: distance between the measurement and the nearest obstacle in the map m
- Failures: Point-mass distribution
  - $p_{max}(z_t^k \mid x_t, m) = \begin{cases} 1 & \text{if } z = z_{max} \\ 0 & \text{otherwise} \end{cases}$
- Unexplained random measurements: Uniform distribution
  - $p_{rand}(z_t^k \mid x_t, m) = \begin{cases} \frac{1}{z_{max}} & \text{if } 0 \le z_t^k \le z_{max} \\ 0 & \text{otherwise} \end{cases}$

Likelihood fields

```
likelihood_field_range_finder (x_t: ROBOT_POSE;
                                z_t: SENSOR_MEASUREMENT;
                                m: MAP ) : REAL_64
   do
        q := 1.0
        across z_t loop
             if z_t^k < z_max then
```

Measurement coordinate

$$x_{z_t^k} := x + x_{k,sens} \cos(\theta) - y_{k,sens} \sin(\theta) + z_t^k \cos(\theta + \theta_{k,sens})$$

$$y_{z_t^k} := y + y_{k,sens} \cos(\theta) + x_{k,sens} \sin(\theta) + z_t^k \sin(\theta + \theta_{k,sens})$$

$$d := m.compute\_distance\_to\_the\_nearest\_obstacle(x_{z_t^k}, y_{z_t^k})$$

$$q := q \cdot \left(z_{hit} \cdot \mathbf{prob}(d, \sigma_{hit}) + \frac{z_{rand}}{z_{max}} \right)$$

```
             end
        end
        Result := q
   end
```

Likelihood fields

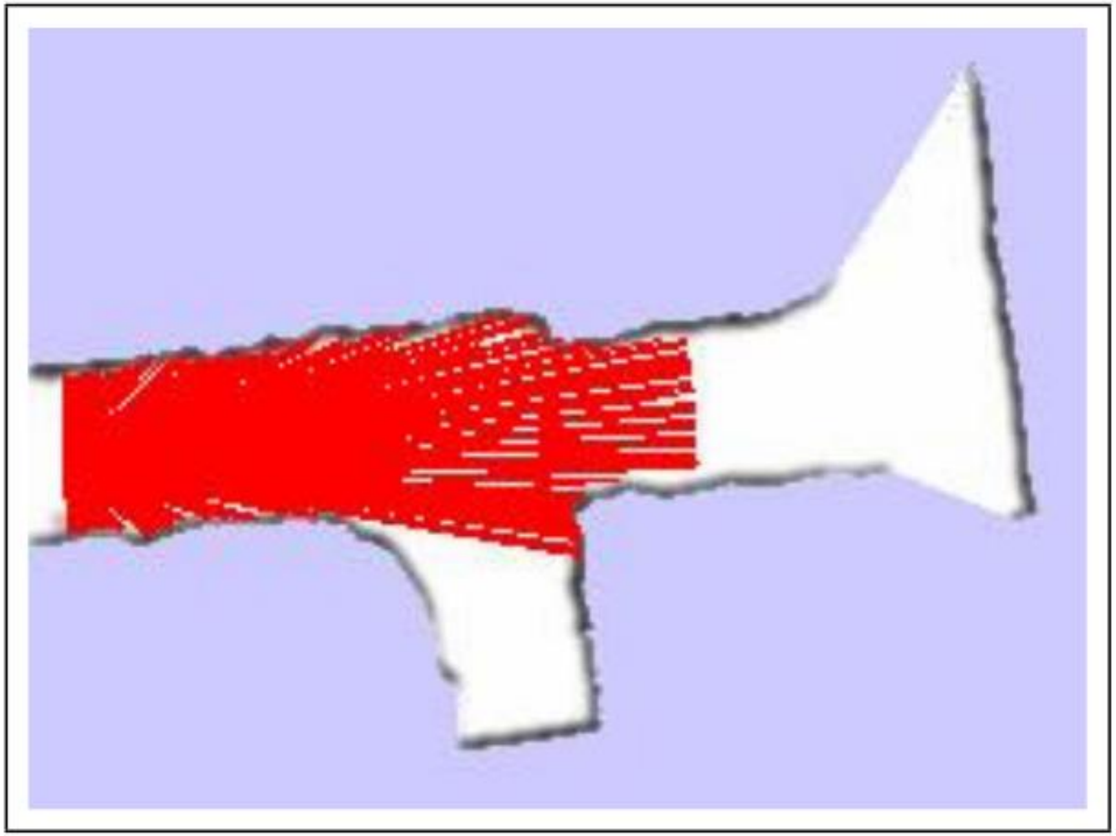
Advantages

- Smooth
  - Small changes in the robot's pose result in small changes of the resulting distribution
- Computationally more efficient than ray casting

Disadvantages

- No modeling of dynamic objects
- Sensors can see through the wall
  - Nearest neighbor cannot determine if a path is obstructed by an obstacle
- No map uncertainty considered
  - Can change occupancy to occupied, free, and unknown

Correlation-based measurement model



Correlation-based measurement model

Map matching

1. Compute a local map m_{robot} from the scans z_t in robot frame
2. Transform the local map m_{robot} to the global coordinate frame m_{local}
3. Compare the local map m_{local} and the map m

$$\rho = \frac{\Sigma_{x,y} (m_{x,y} - \bar{m}) \cdot (m_{x,y,local}(x_t) - \bar{m})}{\sqrt{\Sigma_{x,y} (m_{x,y} - \bar{m})^2 \; \Sigma_{x,y} (m_{x,y,local}(x_t) - \bar{m})^2}} \text{ : correlation}$$

$$\bar{m} = \frac{1}{2N} \Sigma_{x,y} (m_{x,y} + m_{x,y,local}) \text{ : average map value}$$

$$p(m_{local} \mid x_t, m) = \max \{ \rho, 0 \}$$

Correlation-based measurement model

Advantages

- Easy to compute
- Explicitly considers free-space

Disadvantages

- Does not yield smooth probability in pose x_t
    - May convolve the map m with a Gaussian kernel first
- Can incorporate inappropriate local map information
    - May contain areas beyond the maximum sensor range
- Does not include the noise characteristic of range sensors

Feature extraction

feature: compact representation of raw data

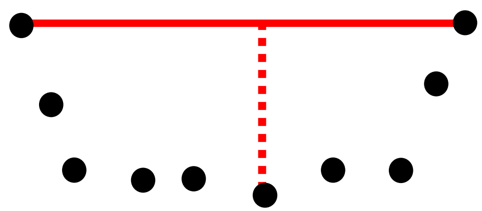
- Range scans: lines, corners, local minima in range scans, etc.
- Camera images: edges, corners, distinct patterns, etc.
- High level features in robotics: places

Advantages of using features

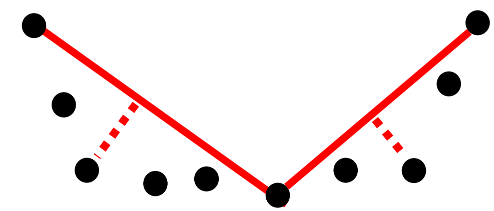
- Reduction of computational complexity
    - Increase in feature extraction
    - Decrease in feature matching

Feature extraction: split and merge

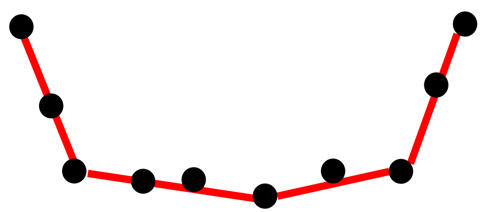
Split



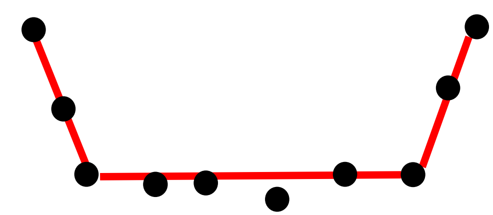
Split



Split



Merge



Feature extraction: split and merge

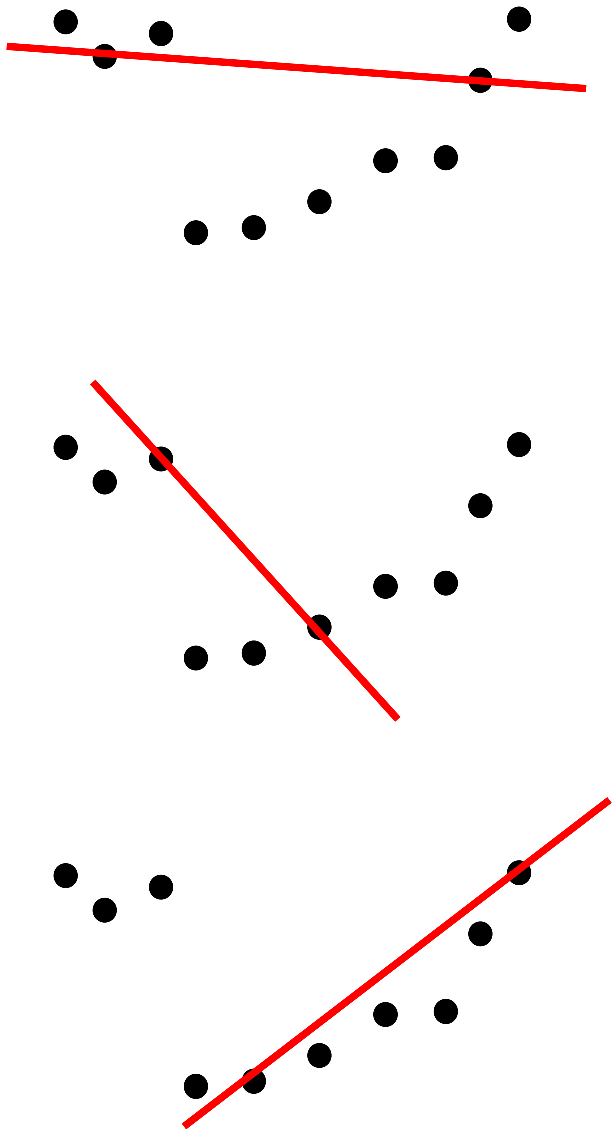
```
split( s: POINT_SET ) : LINE_SET -- sorted points
  do
      p_max := l.compute_farthest_point( s )
      if l.compute_distance( p_max ) > d_max then
            lines.add_set( split( s.split_set(1, p_max) ) )
            lines.add_set( split( s.split_set(p_max, l.size) ) )
      else
            lines.add( l )
      end
      Result := lines
  end
```

Feature extraction: split and merge

```
merge( lines: LINE_SET ) : LINE_SET
   do
         from  until not lines.is_next_pair_collinear loop
              l.merge_lines( lines.left_line , lines.right_line )
              if l.compute_distance( l.compute_farthest_point ) < dmax then
                  out_lines.add(l)
                  lines.mark_current_pair_as_used
              end
            lines.increment_next_pair
         end
         out_lines.add_set( lines.get_all_unmarked_lines )
         Result := out_lines
     end
```

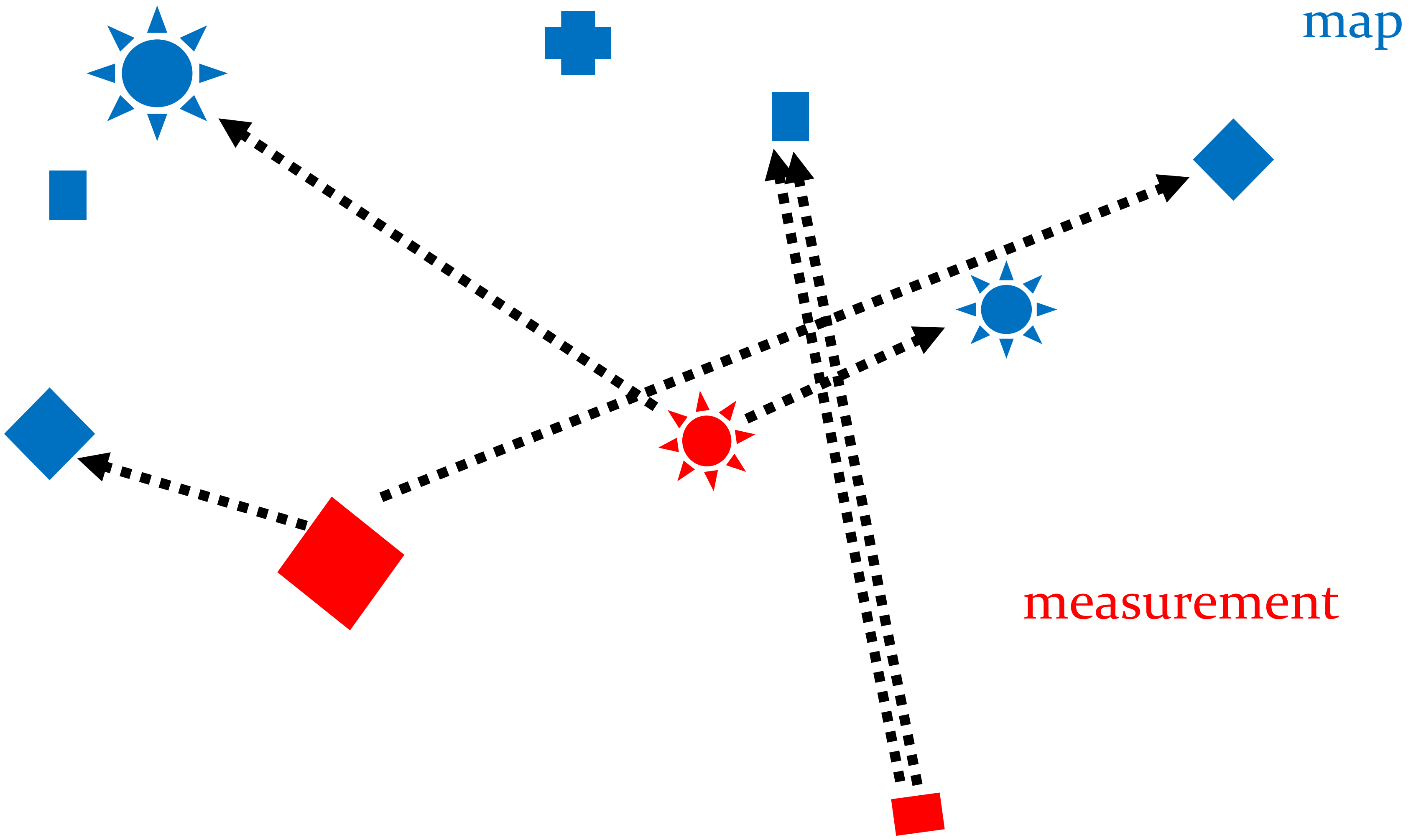
Feature extraction: RANSAC

```
RANSAC( s: POINT_SET ) :  LINE
    do
        from c := 1 until c > c_max loop
            l.set_line_from_two_random_points(s)
            if l.count_inliners > num then
                num := l.count_inliners
                line := l
            end
        end
        Result := line
    end
```



Fischler, M. and Bolles, R. 1981. “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography”. Communications of the ACM. 24(6).

Data association



Data association: nearest neighbor

Measurement: $F = \{f_1, ..., f_n\}$

Map features: $M = \{m_1, ..., m_l\}$

```
nearest_neighbor( F, M: ARRAY[FEATURE] ) : HYPOTHESIS
    do
        from i := 1 until i > n loop
            f_i := F.item(i)
            d_min := d_min.Max_value
            from j := 1 until j > l loop
                m_j := M.item(j)
                d_temp := Mahalanobis2(f_i, m_j)
                if d_temp < d_min then
                    d_min := d_temp
                    m_nearest := m_j
                end
            end
            if d_min < X²(d_i, α) then -- d_i = dim(z_i), α: desired confidence level
                H.add_pair(f_i, m_nearest)
            else
                H.add_pair(f_i, 0)
            end
        end
        Result := H
    end
```

Data association: joint compatibility

Measurement: $F = \{f_1, \ldots, f_n\}$

Map features: $M = \{m_1, \ldots, m_l\}$

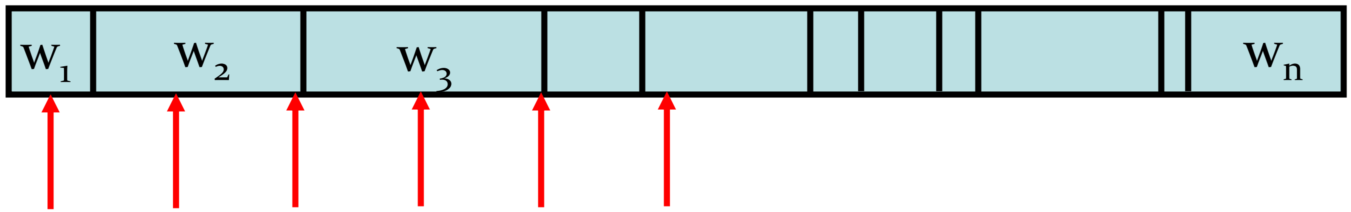
```
joint_compatibility(H: HYPOTHESIS; i: INTEGER_16; F, M: ARRAY[FEATURE] )
    do
            f_i := F.item(i)
            if i > l then
                if H.score > Best.score then
                        Best := H
              end
            else
              from  j := 1  until j > l loop
                        m_j := M.item(j)
                        if is_compatible(f_i, m_j) and H.is_joint_compatible(f_i, m_j) then
                            joint_compatibility(H.add_pair(f_i, m_j) , i+1, F, M)
                        end
              end
                if H.score + n – i >= Best.score then -- can do better?
                        joint_campatibility(H.add_pair(f_i, 0), i+1, F, M)
                end
            end
    end
```

Neira, J. Tardos, J.D. 2001. "Data association in stochastic mapping using the joint compatibility test", Robotics and Automation, IEEE Transactions on 17 (6): 890–897.

Resampling

Roulette wheel sampling

Stochastic universal sampling



distance between two samples = total weight / number of samples

starting sample: random number in [0, distance between samples)

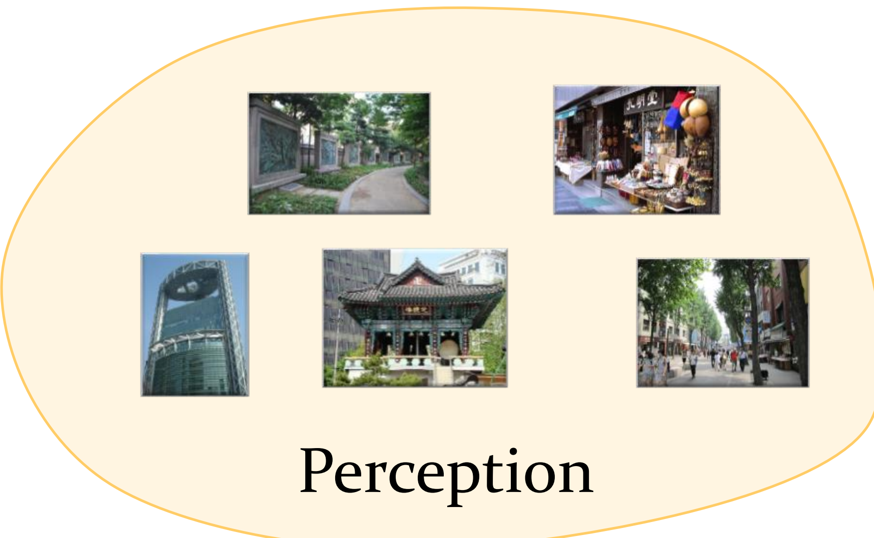
Mapping

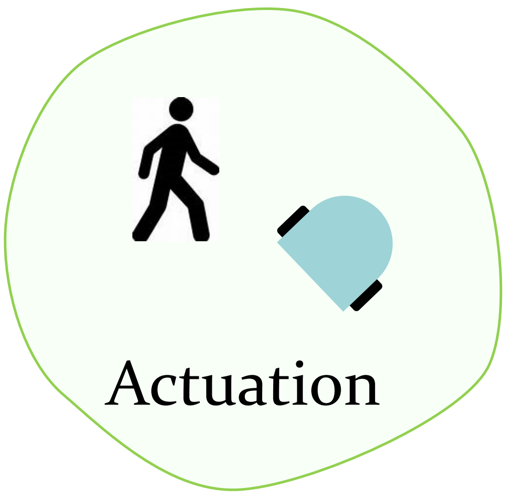
Map: a list of objects and their locations in an environment

- Given N objects in an environment

$$m = \{ m_1, \dots , m_N \}$$

Mapping: the process of creating a map





Types of Maps

Lacation-based map

- $m = \{ m_1, \ldots, m_N \}$ contains N locations
- Volumetric representation
  - A label for any location in the world
  - Knowledge of presence and absence of objects

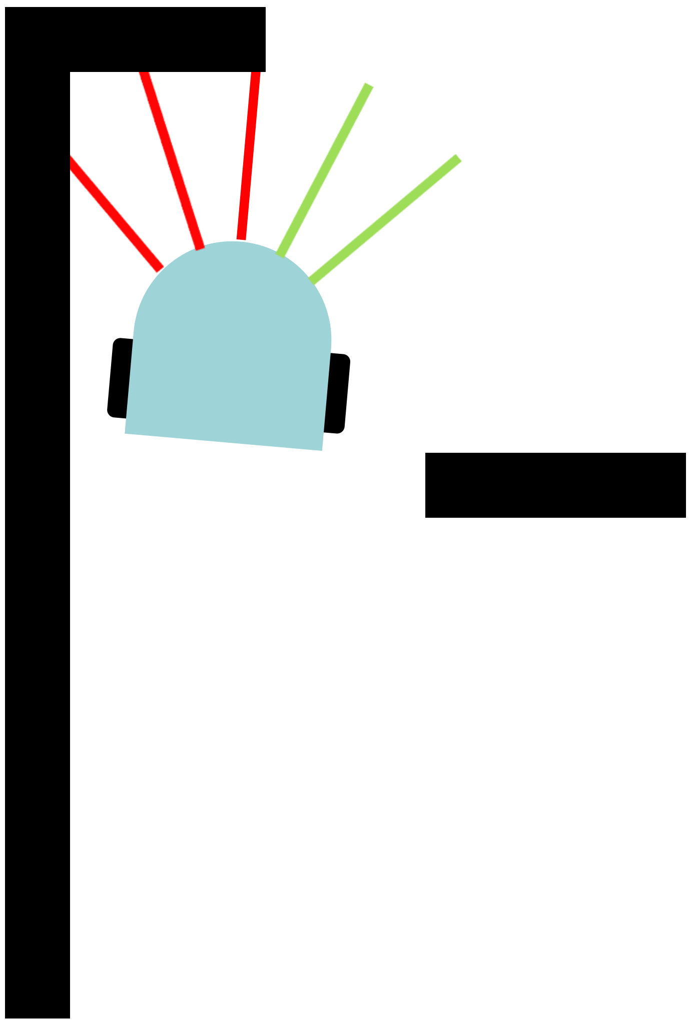
Feature-based map

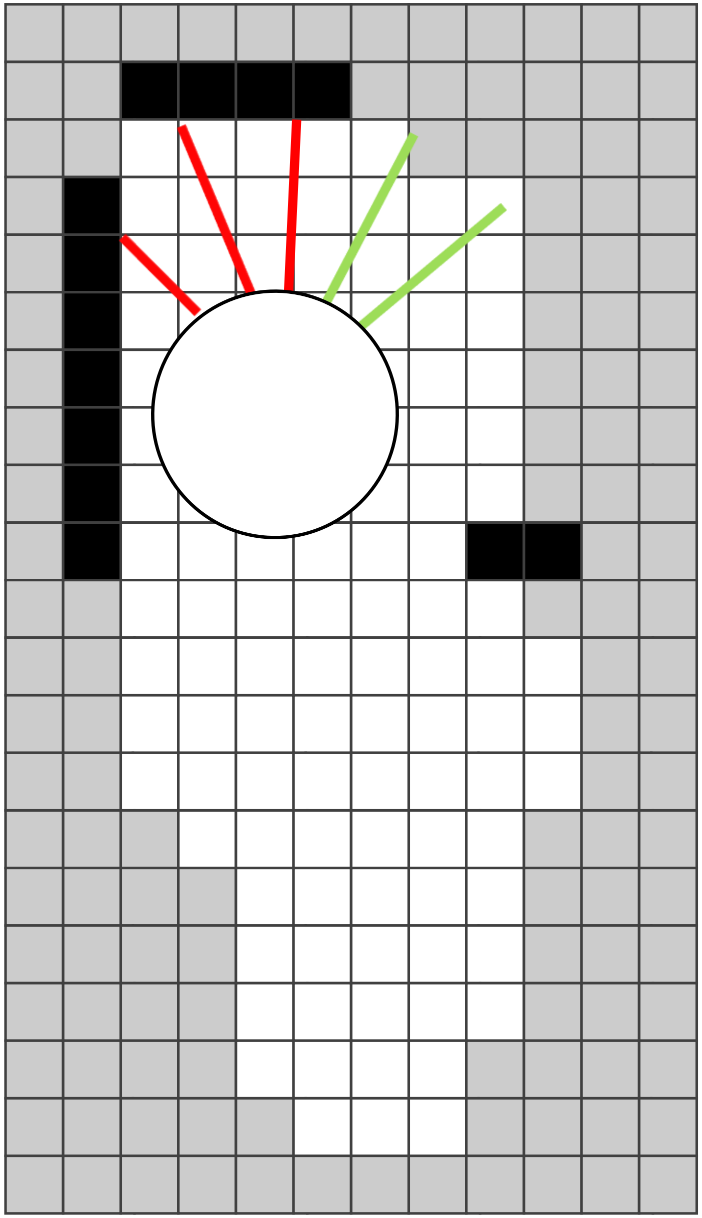
- $m = \{ m_1, \ldots, m_N \}$ contains N features
- Sparse representation
  - A label for each object location
  - Easier to adjust the position of an object

Occupancy grid map

- Location-based map
- An environment as a collection of grid cells
- Each grid cell with a probability value that the cell is occupied
  - Each grid cell is independent!
- Easy to combine different sensor scans and different sensor modalities
- No assumption about type of features

Occupancy grid mapping





Occupancy grid cells

$\mathbf{m}_i$: the grid cell with index i

z_t: the measurement at time t

x_t: the robot's pose (x, y, θ) at time t

$p(\mathbf{m}_i \mid z_t, x_t)$: probability of occupancy

$$\frac{p(\mathbf{m}_i \mid z_t, xt)}{p(\neg\mathbf{m}_i \mid z_t, x_t)} = \frac{p(\mathbf{m}_i \mid z_t, x_t)}{1 - p(\mathbf{m}_i \mid z_t, x_t)} \text{ : odds of occupancy}$$

$$l_{t,i} = \log \frac{p(\mathbf{m}_i \mid z_t, xt)}{1 - p(\mathbf{m}_i \mid z_t, xt)} \text{ : log odds of occupancy}$$

$$p(\mathbf{m}_i \mid z_t, x_t) = 1 - \frac{1}{1 + \exp(lt_{,i})}$$

Bayes' law using log odds

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

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

$$o(A|B) = \frac{p(A|B)}{p(\neg A|B)} = \frac{p(B|A)\, p(A)}{p(B|\neg A)\, p(\neg A)} = \lambda(B|A)\, o(A)$$

$$\log(\, o(A|B)\,) = \log(\, \lambda(B|A)\,) + \log(\, o(A)\,)$$

- Ranges between -∞ and ∞
- Avoids truncation problem around probabilities near 0 and 1

Occupancy grid mapping

occupancy_grid_mapping (x_t : ROBOT_POSE;
z_t : SENSOR_MEASUREMENT;
$\{l_{t-1,i}\}$: OCCUPANCY) : OCCUPANCY

do

  across m as m$_i$ **loop**

    if m$_i$.is_in_perceptial_field(z) **then**

      $l_{t,i} := l_{t-1,i}$ + **inverse_sensor_model** (x_t, z_t, $\mathbf{m}_i$) – l_o

    else

      $l_{t,i} := l_{t-1,i}$

    end

  end

  Result := $\{l_{t,i}\}$

end

$$l_{t,i} := \log \frac{p(\mathbf{m}_i \mid x_{i:t}, zi_{:t})}{1 - p(\mathbf{m}_i \mid x_{i:t}, zi_{:t})}$$

$$l_o := \log \frac{p(\mathbf{m}_i = 1)}{p(\mathbf{m}_i = 0)} := \log \frac{p(\mathbf{m}_i)}{1 - p(\mathbf{m}_i)}$$

Occupancy grid mapping

inverse_range_sensor_model (x_t : ROBOT_POSE;
z_t : SENSOR_MEASUREMENT;
$\mathbf{m}_i$: GRID_CELL) : REAL_64

α: thickness of the obstacle
β: opening angle of the beam
z_{max}: max range of the beam

```
do
    x_i := m_i,x
    y_i := m_i,y
    r := √( (x_i - x)² + (y_i - y)² )                          grid range
    ϕ := atan2(y_i – y, x_i – x) – θ                            grid angle
    k := argmin_j | ϕ – θ_j,sens |                              beam index
    if r > min( z_max, z_t^k + α/2 ) or | ϕ – θ_j,sens | > β/2 then
            Result := l_o        grid out of range or behind an obstacle
    elseif z_t^k < z_max and | r - z_t^k | < α/2 then
            Result := l_occ                              grid in the obstacle
    else -- r <= z_t^k
            Result := l_free                                grid unaccupied
    end
end
```

But what about drift?

Localization

- If we have a map, we can localize

Mapping

- If we know the robot's pose, we can map

Do both!

- Estimate a map
- Localize itself relative to the map

Simultaneous Localization and Mapping (SLAM)

Simultaneous Localization and Mapping

Localization: $p(x \mid m, z, u)$

Mapping: $p(m \mid x, z)$

SLAM: $p(x, m \mid z, u)$

- The map depends on the robot's pose during the measurement
- If the pose is known, mapping is easy

Rao-Blackwellization

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

SLAM posterior = robot path posterior * mapping with known poses

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

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

$x_{1:t}$: the robot's poses (x, y, θ)

m: the map

$z_{1:t}$: the measurements

$u_{0:t-1}$: the controls

Murphy, K. 1999. Bayesian map learning in dynamic environments. In NIPS '99 (Neural Info. Proc. Systems)

Rao-Blackwellized particle filter SLAM

Use a particle filter to represent potential trajectories of the robot

- Every particle carries its own map
- The probability of survival of a particle is proportional to the likelihood of the measurement with respect to the particle's own map

Problem: big map * large number of particles!

Improve pose estimate

- Use scan matching to compute locally consistent pose correction
- Smaller error -> fewer particles necessary