



## Intelligent Robotics Systems

Armin Biess



## **Robot motion and perception**

## News from the robotics world

clip

## **Robot motion**

## Motion models

- **state-transition pdf or motion model:**

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

is the pdf for the transition from state  $x_{t-1}$  to state  $x_t$  performing action  $u_t$

- **prediction step** in Bayes filter algorithm

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

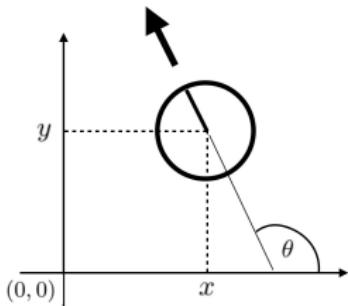
- **next:** derive motion models for specific robots

## Kinematic motion models

- ignore dynamics
- In practice, one often finds two types of kinematic motion models:
  - odometry-based**
  - velocity-based**
- Odometry-based models for systems that are equipped with wheel encoders
- Velocity-based when no wheel encoders are available

## Kinematic motion models

- kinematic motion model of a rigid robot: six DOF (three Cartesian coordinates and three Euler angles)
- mobile robot in a planar environment: three DOF  
**robot pose = location**  $(x, y)$  and **orientation**  $\theta$



$$\text{state: } \boldsymbol{x} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$$

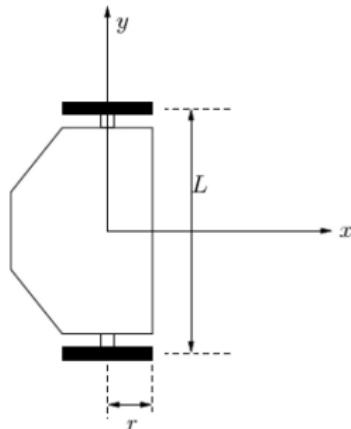
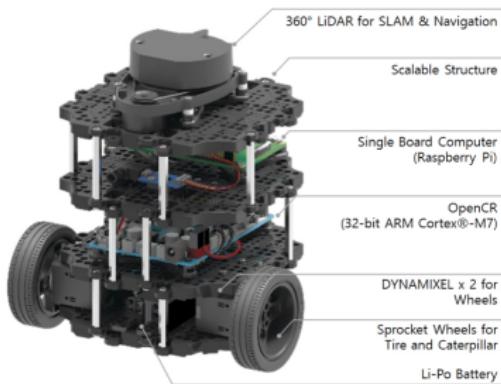
remark: orientation is called **bearing** or **heading direction**

$$\theta = 0 : \quad \text{robot moves in } x\text{-direction}$$

$$\theta = \pi/2 : \quad \text{robot moves in } y\text{-direction}$$

# Velocity-based motion model: Differential drive

TurtleBot<sub>3</sub> Burger



## Differential drive: motion model

motion model:

$$\dot{x} = \frac{r}{2}(u_l + u_r) \cos \theta$$

$u = (u_l, u_r)$ : action  
= angular wheel velocities (in rad/s)

$$\dot{y} = \frac{r}{2}(u_l + u_r) \sin \theta$$

$L$ : distance between the two wheels

$r$ : radius of wheel

$$\dot{\theta} = \frac{r}{L}(u_l - u_r)$$

## Differential drive: motion model

motion model:

it is sometimes preferable to transform the action space:

$$(u_l, u_r) \Rightarrow (u_{tr}, u_{rot})$$

$$\begin{aligned} u_{tr} &= (u_l + u_r)/2 \\ u_{rot} &= (u_l - u_r) \end{aligned}$$

$u_{tr}$  : angular velocity inducing translation

$u_{rot}$  : angular velocity inducing rotation

$$\begin{aligned} \dot{x} &= u_{tr} \cos \theta \\ \dot{y} &= u_{tr} \sin \theta \\ \dot{\theta} &= \frac{r}{L} u_{rot} \end{aligned}$$

## Differential drive: modes of operation

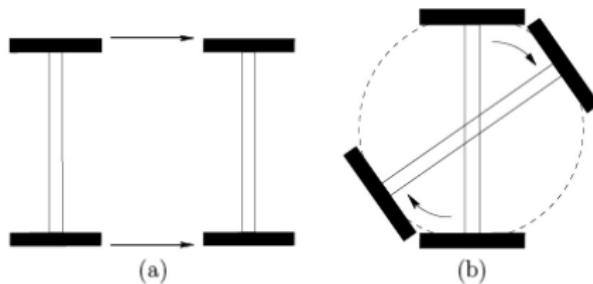


Figure 13.3: (a) Pure translation occurs when both wheels move at the same angular velocity; (b) pure rotation occurs when the wheels move at opposite velocities.

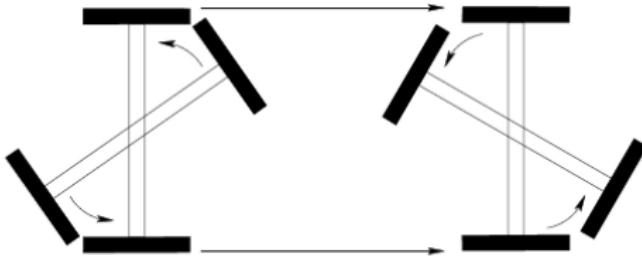


Figure 13.4: The shortest path traversed by the center of the axle is simply the line segment that connects the initial and goal positions in the plane. Rotations appear to be cost-free.

## Differential drive: discretized motion model

- robot moves from  $(x, y, \theta)$  to  $(x', y', \theta')$  in time interval  $\Delta t$
- motion command  $u = (u_l, u_r)$

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} \frac{r}{2}(u_l + u_r) \cos \theta \Delta t \\ \frac{r}{2}(u_l + u_r) \sin \theta \Delta t \\ \frac{r}{L}(u_l - u_r) \Delta t \end{pmatrix}$$

## Differential drive: real motion model

- real motion is noisy
- actual velocities  $(\hat{u}_l, \hat{u}_r)^T$  differ from the commanded ones  $(u_l, u_r)^T$ :

$$\begin{pmatrix} \hat{u}_l \\ \hat{u}_r \end{pmatrix} = \begin{pmatrix} u_l \\ u_r \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \end{pmatrix},$$

where  $\epsilon_1$  and  $\epsilon_2$  are Gaussian random variables with zero mean:

$$\epsilon_1 \sim \mathcal{N}(0, \alpha_1|u_l| + \alpha_2|u_r|)$$

$$\epsilon_2 \sim \mathcal{N}(0, \alpha_3|u_l| + \alpha_4|u_r|)$$

and  $\alpha_i \geq 0$  ( $i = 1, \dots, 4$ ) are robot-specific error parameters.

- real motion model

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} \frac{r}{2}(\hat{u}_l + \hat{u}_r) \cos \theta \Delta t \\ \frac{r}{2}(\hat{u}_l + \hat{u}_r) \sin \theta \Delta t \\ \frac{r}{L}(\hat{u}_l - \hat{u}_r) \Delta t \end{pmatrix}$$

## Differential drive: sampling algorithm

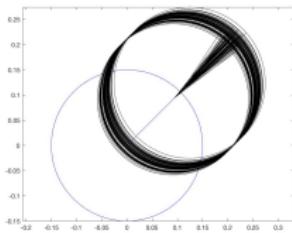
- sample poses  $(x', y', \theta')$  from a pose  $(x, y, \theta)$  and control  $u = (u_l, u_r)$

- 1: **Algorithm sample differential drive**  $(u_t, x_{t-1})$   
 $(u_l, u_r) = u_t, (x, y, \theta) = x_{t-1}$
- 2:  $\hat{u}_l = u_l + \text{sample}(0, \alpha_1|u_l| + \alpha_2|u_r|)$
- 3:  $\hat{u}_r = u_r + \text{sample}(0, \alpha_3|u_l| + \alpha_4|u_r|)$
- 4:  $x' = x + \frac{r}{2}(\hat{u}_l + \hat{u}_r) \cos \theta \Delta t$
- 5:  $y' = y + \frac{r}{2}(\hat{u}_l + \hat{u}_r) \sin \theta \Delta t$
- 6:  $\theta' = \theta + \frac{r}{L}(\hat{u}_l - \hat{u}_r) \Delta t$
- 7: **return**  $x_t = (x', y', \theta')$

## Sampling algorithm - velocity motion model

robots:  $N = 100, L = 0.3, r = 0.05$

initial pose:  $x_0 = (0, 0, \pi/4)$



motor commands:

$$u_l = 3$$

$$u_r = 3$$

noise parameters:

$$\alpha_1 = 0.05$$

$$\alpha_2 = 0.05$$

$$\alpha_3 = 0.05$$

$$\alpha_4 = 0.05$$

motor commands:

$$u_l = 3$$

$$u_r = -3$$

noise parameters:

$$\alpha_1 = 0.05$$

$$\alpha_2 = 0.05$$

$$\alpha_3 = 0.05$$

$$\alpha_4 = 0.05$$

motor commands:

$$u_l = 3$$

$$u_r = -3$$

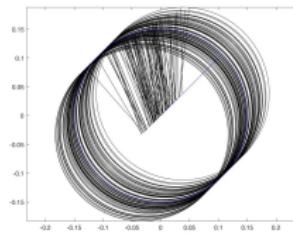
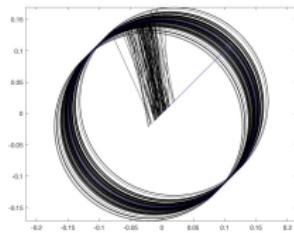
noise parameters:

$$\alpha_1 = 0.05$$

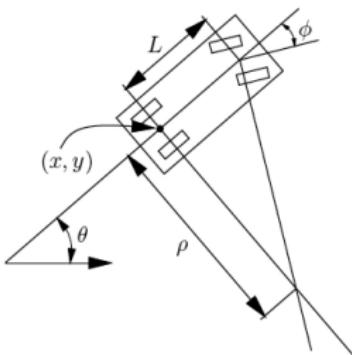
$$\alpha_2 = 0.05$$

$$\alpha_3 = 0.15$$

$$\alpha_4 = 0.15$$



## Velocity-based motion model: Simple car model - version 1



derivation:

motion model:

$$\dot{x} = v \cos \theta$$

$$\dot{y} = v \sin \theta$$

$$\dot{\theta} = \frac{v}{L} \tan \phi$$

$$ds = \rho d\theta$$

$$\rho = L / \tan \phi$$

$\Rightarrow$

$$\dot{\theta} = \frac{ds}{L} \tan \phi$$

action:  $u = (v, \phi)$

$L$ : length of car

## Velocity-based motion model: Simple car model

motion model - version 2:

$$\begin{aligned}\dot{x} &= v \cos \theta \\ \dot{y} &= v \sin \theta \\ \dot{\theta} &= \frac{v}{L} \tan \phi \\ \dot{v} &= a\end{aligned}$$

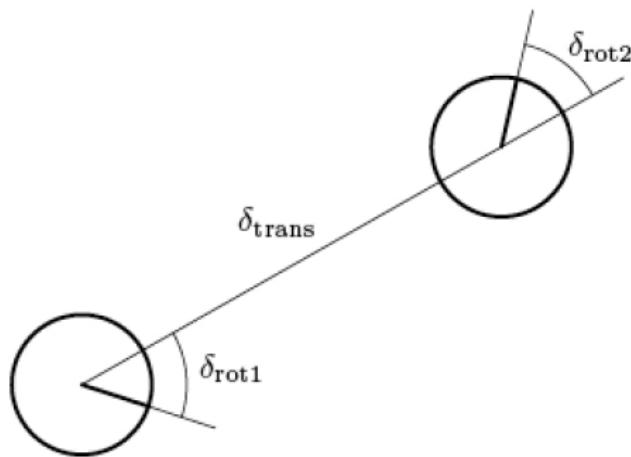
action:  $u = (a, \phi)$

motion model - version 3:

$$\begin{aligned}\dot{x} &= v \cos \theta \\ \dot{y} &= v \sin \theta \\ \dot{\theta} &= \frac{v}{L} \tan \phi \\ \dot{v} &= a \\ \dot{\phi} &= \omega\end{aligned}$$

action:  $u = (a, \omega)$

## Odometry motion model



The robot motion is modeled by a rotation  $\delta_{rot1}$ , a translation  $\delta_{trans}$  and another rotation  $\delta_{rot2}$ .

**Homework:** PR, p.132-139

## Summary

from deterministic to probabilistic motion models:

- ① identify (kinematic) state and actions
- ② derive deterministic (kinematics) motion model from geometry and physics
- ③ discretize motion model
- ④ add noise to action variables (often Gaussian white noise)

## Motion and maps

- robot motion in **free space**:  $p(x_t|u_t, x_{t-1})$
- robot motion in **an environment** described by a **map**  $m$ :  
 $p(x_t|u_t, x_{t-1}, m)$

If  $m$  carries information relevant to pose estimation then:

$$p(x_t|u_t, x_{t-1}) \neq p(x_t|u_t, x_{t-1}, m)$$

Approximation:  $p(x_t|u_t, x_{t-1}, m) = \eta p(x_t|u_t, x_{t-1}) \cdot p(x_t|m)$

Proof:

$$\begin{aligned} p(x_t|u_t, x_{t-1}, m) &\stackrel{\text{Bayes}}{=} \eta \cdot p(m|x_t, u_t, x_{t-1}) \cdot p(x_t|u_t, x_{t-1}) \\ &\stackrel{\text{Markov}}{=} \eta \cdot p(m|x_t) \cdot p(x_t|u_t, x_{t-1}) \\ &\stackrel{\text{Bayes}}{=} \eta \cdot \frac{p(x_t|m)p(m)}{p(x_t)} \cdot p(x_t|u_t, x_{t-1}) \\ &= \eta \cdot \frac{p(x_t|m) \cdot p(x_t|u_t, x_{t-1})}{p(x_t)} \\ &= \eta \cdot p(x_t|m) \cdot p(x_t|u_t, x_{t-1}) \end{aligned}$$

## Motion and maps

```
1: Algorithm motion_model_with_map( $x_t, u_t, x_{t-1}, m$ ):  
2:     return  $p(x_t | u_t, x_{t-1}) \cdot p(x_t | m)$ 
```

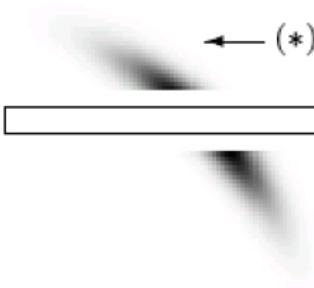
```
1: Algorithm sample_motion_model_with_map( $u_t, x_{t-1}, m$ ):  
2:     do  
3:          $x_t = \text{sample\_motion\_model}(u_t, x_{t-1})$   
4:          $\pi = p(x_t | m)$   
5:         until  $\pi > 0$   
5:     return  $\langle x_t, \pi \rangle$ 
```

## Motion and maps

(a)  $p(x_t | u_t, x_{t-1})$



(b)  $p(x_t | u_t, x_{t-1}, m)$



## **Robot perception**

## Measurement models

- **Measurement models or sensor models:**

$$p(z_t|x_t)$$

is the pdf for observing  $z_t$  in state  $x_t$

- **measurement step (correction)** in Bayes filter algorithm

$$bel(x_t) = \int p(z_t|x_t) \bar{bel}(x_{t-1}) dx_{t-1}$$

- **next:** derive measurement models for specific sensors

## Measurement and maps

- measurement model in **free space**:

$$p(z_t|x_t)$$

- measurements in an **environment** described by a map:

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

# Maps

- a **map** is a list of objects in the environment along with their properties (locations, signatures, etc)

$$m = \{m_1, m_2, \dots, m_n, \dots m_N\}$$

two types of maps:

- ① **location-based** maps
- ② **feature-based** maps

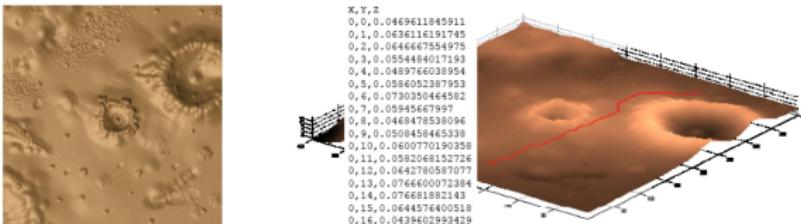
## Location-based maps

- in **location-based** maps the element  $m_n$  specifies the location of the element. In planar map one writes  $m_{x,y}$ .
- location-based maps are **volumetric**, since they give a label to every point in the environment
- volumetric maps contain information not only about the object in the map, but also about the absence of objects (free space)

## Location-based maps - examples

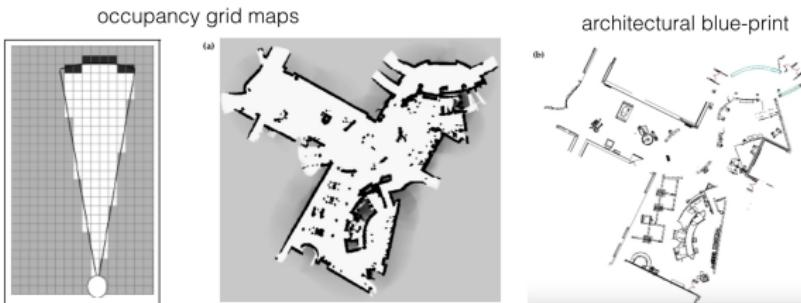
- **digital elevation map (DEM)**

$$m_{x,y} \Rightarrow (x, y, z)$$



- **occupancy grid map**

$$m_{x,y} \Rightarrow (x, y, o), \text{ where } o = 0 \text{ free and } o = 1 \text{ occupied}$$



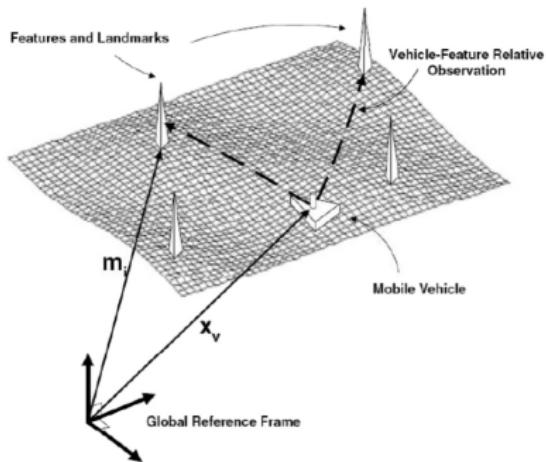
## Feature-based maps

- features are distinctive elements of geometric primitives of the environment
- in feature-based maps the element  $m_k$  contains the property (or properties) of the feature and its location on the map:

$$m_n = \left( \underbrace{m_{j,x}, m_{j,y}}_{\text{location of feature } j}, \underbrace{s_j}_{\text{property of feature } j} \right)$$

- feature-based maps only specify the shape of the environment at specific locations (not volumetric).
- features-based maps are popular in robotics. Features can be extracted from measurements  $z_t$  using vision at no costs. This can be formalized with a feature extractor function  $f(z_t)$
- reduction of computational complexity: highdimensional measurement space  $\Rightarrow$  lowdimensional feature space
- features often corresponds to distinct objects or places in the environment: **landmarks**

## Feature-based maps - examples

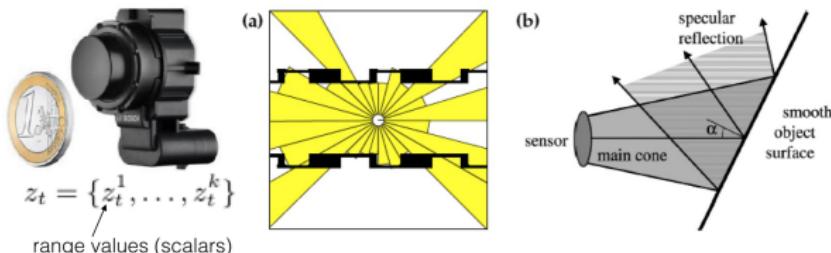


six landmarks

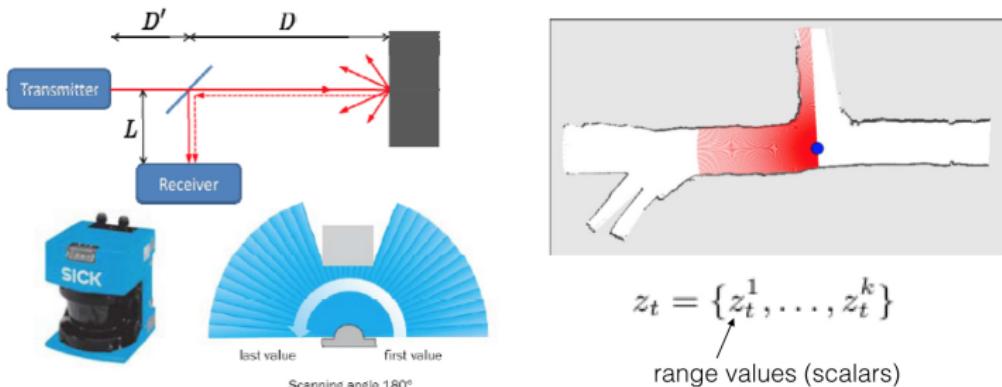
## Sensors - Range scanners

most mobile robots have range scanners

- **sonar range scanners** - range is measured along a cone



- **laser range scanners** - range is measured along a beam



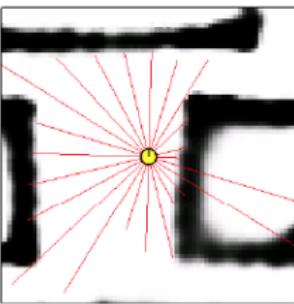
## Measurement model

- scan  $z$  consists of  $K$  measurements

$$z_t = \{z_t^1, \dots, z_t^K\}$$

- individual measurements are independent given the robot position  $x_t$  and map  $m$

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



$z_t^k$  is an individual measurement (e.g., one range value)  
 $m$  is a map

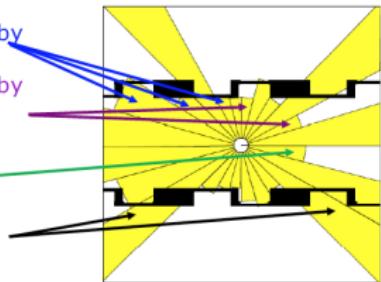
## Beam models of range finders

- measurement model  $p(z_t^k | x_t, m)$  of range finders

four types of measurement errors:

- ① small measurement noise
- ② errors due to unexpected objects
- ③ errors due to failures to detect objects
- ④ random unexplained noise

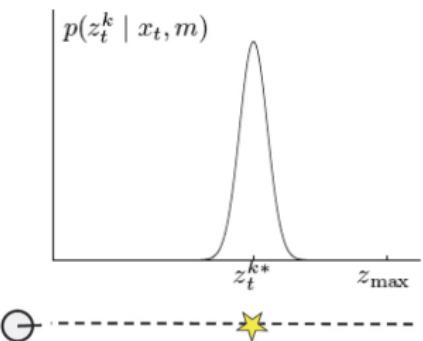
1. Beams reflected by obstacles
2. Beams reflected by persons / caused by crosstalk
3. Random measurements
4. Maximum range measurements



# Beam models of range finders - $p(z_t^k | x_t, m)$

## 1. measurement noise

(a) Gaussian distribution  $p_{\text{hit}}$

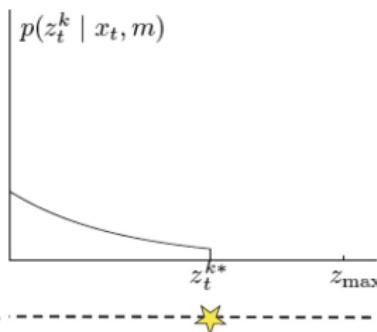


$$p_{\text{hit}}(z_t^k | x_t, m) = \begin{cases} \eta \mathcal{N}(z_t^k; z_t^{k*}, \sigma_{\text{hit}}^2) & \text{if } 0 \leq z_t^k \leq z_{\max} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathcal{N}(z_t^k; z_t^{k*}, \sigma_{\text{hit}}^2) = \frac{1}{\sqrt{2\pi\sigma_{\text{hit}}^2}} e^{-\frac{1}{2}\frac{(z_t^k - z_t^{k*})^2}{\sigma_{\text{hit}}^2}}$$

## 2. unexpected obstacles

(b) Exponential distribution  $p_{\text{short}}$



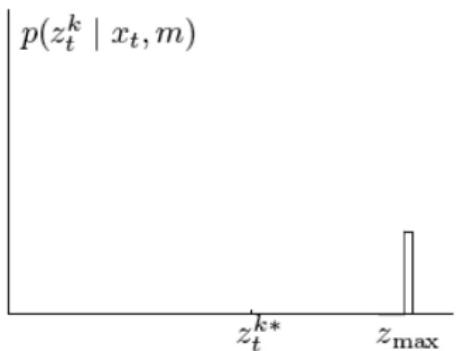
$$p_{\text{short}}(z_t^k | x_t, m) = \begin{cases} \eta \lambda_{\text{short}} e^{-\lambda_{\text{short}} z_t^k} & \text{if } 0 \leq z_t^k \leq z_t^{k*} \\ 0 & \text{otherwise} \end{cases}$$

$z_t^k$ : true range of the object extracted from a map (location or feature-based map)

## Beam models of range finders - $p(z_t^k | x_t, m)$

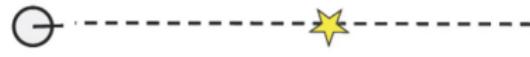
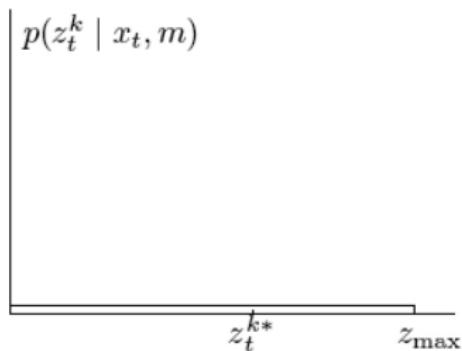
### 3. failures

(c) Uniform distribution  $p_{\max}$



### 4. random measurements

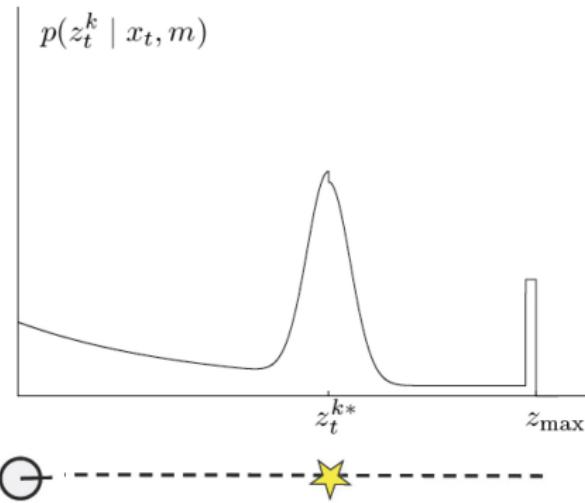
(d) Uniform distribution  $p_{\text{rand}}$



$$p_{\max}(z_t^k | x_t, m) = I(z = z_{\max}) = \begin{cases} 1 & \text{if } z = z_{\max} \\ 0 & \text{otherwise} \end{cases}$$

$$p_{\text{rand}}(z_t^k | x_t, m) = \begin{cases} \frac{1}{z_{\max}} & \text{if } 0 \leq z_t^k < z_{\max} \\ 0 & \text{otherwise} \end{cases}$$

## Beam models of range finders - $p(z_t^k | x_t, m)$

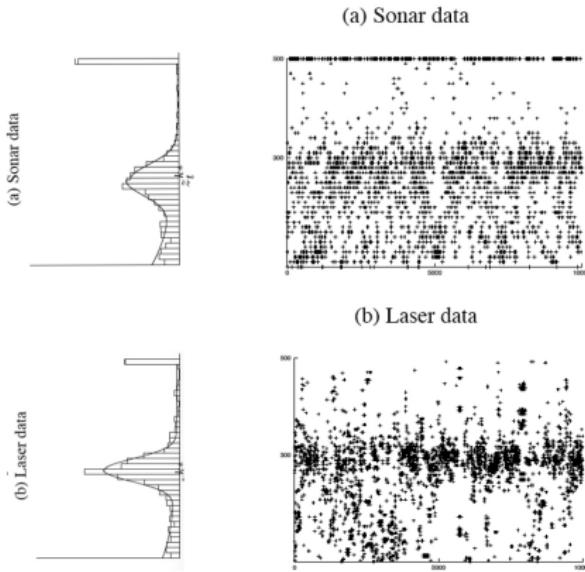


final measurement pdf = mixture of the four distributions

$$p(z_t^k | x_t, m) = \begin{pmatrix} z_{\text{hit}} \\ z_{\text{short}} \\ z_{\max} \\ z_{\text{rand}} \end{pmatrix}^T \cdot \begin{pmatrix} p_{\text{hit}}(z_t^k | x_t, m) \\ p_{\text{short}}(z_t^k | x_t, m) \\ p_{\max}(z_t^k | x_t, m) \\ p_{\text{rand}}(z_t^k | x_t, m) \end{pmatrix}$$

## Beam models - raw measurement data and fitted pdf

measured distance for expected distance of  $z_t^{k*} = 300$  cm



Optimization techniques are used to choose the intrinsic model parameters (e.g., EM algorithm)

## Beam measurement model

```
1: Algorithm beam_range_finder_model( $z_t, x_t, m$ ):
2:      $q = 1$ 
3:     for  $k = 1$  to  $K$  do
4:         compute  $z_t^{k*}$  for the measurement  $z_t^k$  using ray casting
5:          $p = z_{\text{hit}} \cdot p_{\text{hit}}(z_t^k | x_t, m) + z_{\text{short}} \cdot p_{\text{short}}(z_t^k | x_t, m)$ 
6:          $+ z_{\text{max}} \cdot p_{\text{max}}(z_t^k | x_t, m) + z_{\text{rand}} \cdot p_{\text{rand}}(z_t^k | x_t, m)$ 
7:          $q = q \cdot p$ 
8:     return  $q$ 
```

**Input:** map  $m$ , robot pose  $x_t = (x, y, \theta)$ , complete range scan  
 $z_t = \{z_t^1, z_t^2, \dots, z_t^K\}$

**Output:** numerical value of measurement pdf  $q = p(z_t | x_t, m)$

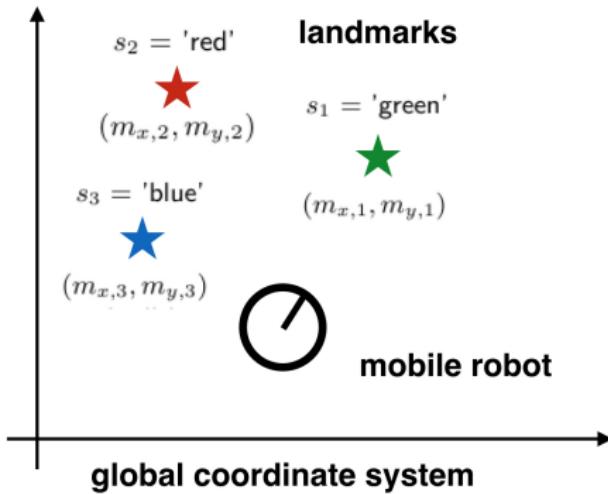
## Feature-based measurement models

Given:

- ① a **map** with a list of features (landmarks)

$$m = (m_1, m_2, \dots, m_n, \dots, m_N)$$

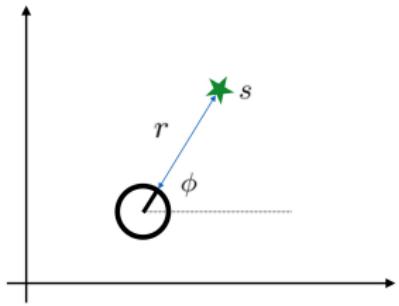
with  $m_n = (\underbrace{m_{j,x}, m_{j,y}}_{\text{location of feature } j}, \underbrace{s_j}_{\text{property of feature } j})$



## Feature-based measurement models

- ② a sensor can measure the **range** and **bearing** of the landmarks with respect to the local robot coordinate frame
- ③ the feature extractor  $f(z_t)$  can provide one or more **signatures** (height, color, binary code, etc)

measurements:



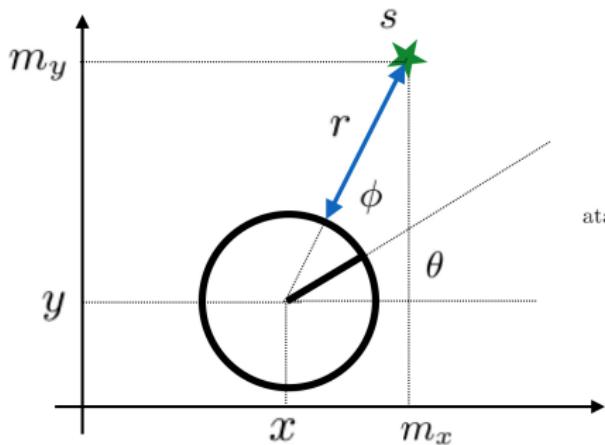
$$f(z_t) = \{f_t^1, f_t^2, \dots\} = \left\{ \begin{pmatrix} r_t^1 \\ \phi_t^1 \\ s_t^1 \end{pmatrix}, \begin{pmatrix} r_t^2 \\ \phi_t^2 \\ s_t^2 \end{pmatrix}, \dots \right\}$$

measurement model:

$$p(f(z_t)|x_t, m) = \prod_i p(r_t^i, \phi_t^i, s_t^i | x_t, m)$$

next: derive the measurements from the given pose  $x_t$  and map  $m$  ...

## Feature-based measurement models



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

*i-th* feature at time  $t$  corresponds to *j-th* landmark in the map:

$$\begin{pmatrix} r_t^i \\ \phi_t^i \\ s_t^i \end{pmatrix} = \begin{pmatrix} \sqrt{(m_{j,x} - x)^2 + (m_{j,y} - y)^2} \\ \text{atan2}(m_{j,y} - y, m_{j,x} - x) - \theta \\ s_j \end{pmatrix} + \begin{pmatrix} \epsilon_{\sigma_r^2} \\ \epsilon_{\sigma_\phi^2} \\ \epsilon_{\sigma_s^2} \end{pmatrix} \quad \text{robot pose : } \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$$

$$\epsilon \sim \mathcal{N}$$

## Feature-based measurement models

- Key problem: data association. Who is what?
- We assume that a **correspondence variable** exists, able to uniquely identifying the landmark:

$$c_t^i \in \{1, 2, \dots, N + 1\}, \quad N : \text{total number of landmarks}$$

if  $c_t^i = j \leq N$  then the  $i$ -th feature corresponds to the  $j$ -th landmark

if  $c_t^i = N + 1$  then the  $i$ th feature does not corresponds to any feature in the map  $m$

## Feature-based measurement models

Now we can compute the feature-based measurement pdf:

$$p(f_t^i | c_t^i, x_t, m)$$

```
1:   Algorithm landmark_model_known_correspondence( $f_t^i, c_t^i, x_t, m$ ):  
2:      $j = c_t^i$   
3:      $\hat{r} = \sqrt{(m_{j,x} - x)^2 + (m_{j,y} - y)^2}$   
4:      $\hat{\phi} = \text{atan2}(m_{j,y} - y, m_{j,x} - x) - \theta$   
5:      $q = \text{prob}(r_t^i - \hat{r}, \sigma_r^2) \cdot \text{prob}(\phi_t^i - \hat{\phi}, \sigma_\phi^2) \cdot \text{prob}(s_t^i - s_j, \sigma_s^2)$   
6:     return  $q$ 
```

**Input:** map  $m$ , pose  $x_t = (x, y, \theta)$ , feature measurement  $f_t^i$ , correspondence variable  $c_t^i$

**Output:** numerical value of measurement pdf  $q = p(f_t^i | c_t^i, x_t, m)$

## MC Localization revisited

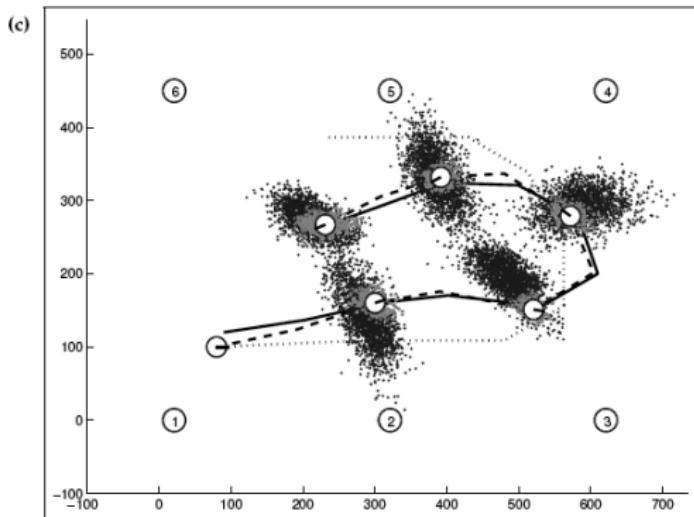
MCL algorithm for landmark-based localization

```
1:   Algorithm MCL( $\mathcal{X}_{t-1}, u_t, z_t, m$ ):
2:      $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$ 
3:     for  $m = 1$  to  $M$  do
4:        $x_t^{[m]} = \text{sample\_motion\_model}(u_t, x_{t-1}^{[m]})$ 
5:        $w_t^{[m]} = \text{measurement\_model}(z_t, x_t^{[m]}, m)$ 
6:        $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$ 
7:     endfor
8:     for  $m = 1$  to  $M$  do
9:       draw  $i$  with probability  $\propto w_t^{[i]}$ 
10:      add  $x_t^{[i]}$  to  $\mathcal{X}_t$ 
11:    endfor
12:    return  $\mathcal{X}_t$ 
```

The algorithm **landmark\_model\_known\_correspondance** provides the likelihood model used in line 5 to weigh the predicted samples.

## MC Localization revisited - example

MCL algorithm for landmark-based localization



six landmarks:  $1, 2, \dots, 6$

... : path from commanded control

—: true trajectory

—: mean path estimate by MCL algorithm

## MC Localization revisited

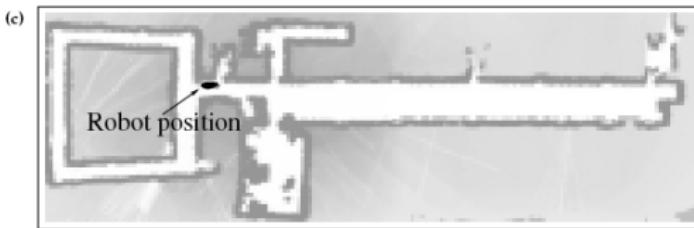
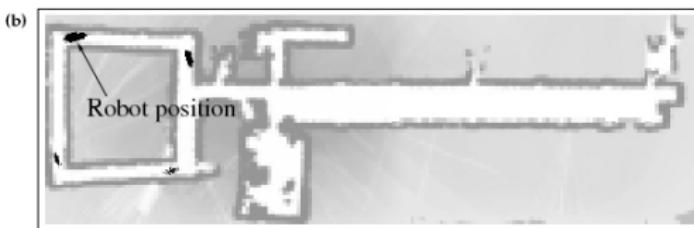
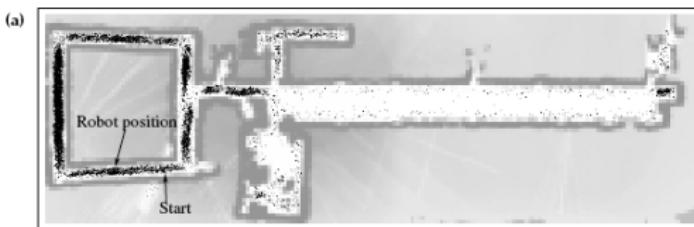
MCL algorithm using an array of sonar range finders

```
1:   Algorithm MCL( $\mathcal{X}_{t-1}, u_t, z_t, m$ ):  
2:      $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$   
3:     for  $m = 1$  to  $M$  do  
4:        $x_t^{[m]} = \text{sample\_motion\_model}(u_t, x_{t-1}^{[m]})$   
5:        $w_t^{[m]} = \text{measurement\_model}(z_t, x_t^{[m]}, m)$   
6:        $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$   
7:     endfor  
8:     for  $m = 1$  to  $M$  do  
9:       draw  $i$  with probability  $\propto w_t^{[i]}$   
10:      add  $x_t^{[i]}$  to  $\mathcal{X}_t$   
11:    endfor  
12:    return  $\mathcal{X}_t$ 
```

The algorithm **beam\_range\_finder\_model** provides the likelihood model used in line 5 to weigh the predicted samples.

## MC Localization revisited - example

MCL algorithm using an array of sonar range finders



## Acknowledgements - additional resources

- Lecture Notes: *Robot Mapping* by Cyrill Stachniss
- Lecture Notes: *Probabilistic Fundamentals in Robotics* by Basilio Bona