Part II

Probabilistic Models

Probabilistic Modeling for Mobile Robotics

Mobile Robotics

Mobile robots are those capable of moving in their surrounding.

Personal mobility	Autonomous navigation	Intelligent agriculture

0:00 / 1:17

State-space representation

The robot's location around the environment is modeled as a time series with index t representing a time step

1. Robot pose $\mathbf{x}_t = [x_t,\ y_t,\ \theta_t]$

   Robot's locations in 2D space

2. Robot Perception $\mathbf{z}_t$

   Vector with all sensor measurements

3. Action $\mathbf{u}_t$

   Actions that cause a change in other states

Modeling Robot's Pose

Considering a starting pose x_0,
It is assumed that the robot first takes an action u_1, which leads the robot to a new pose x_1, where a new observation z_1 is made

This process (action->new pose->new measure) is repeated indefinitely.

Modeling Robot's Pose

Then, for an arbitrary point t in the time series

Our belief on the robot's pose, conditioned on all previous observations and actions, is

$$p(x_t | x_{0:t-1}, z_{1:t-1}, u_{1:t})$$

Our belief for new measurements, conditioned on all previous observations is

$$p(z_t | x_{0:t}, z_{1:t-1}, u_{1:t})$$

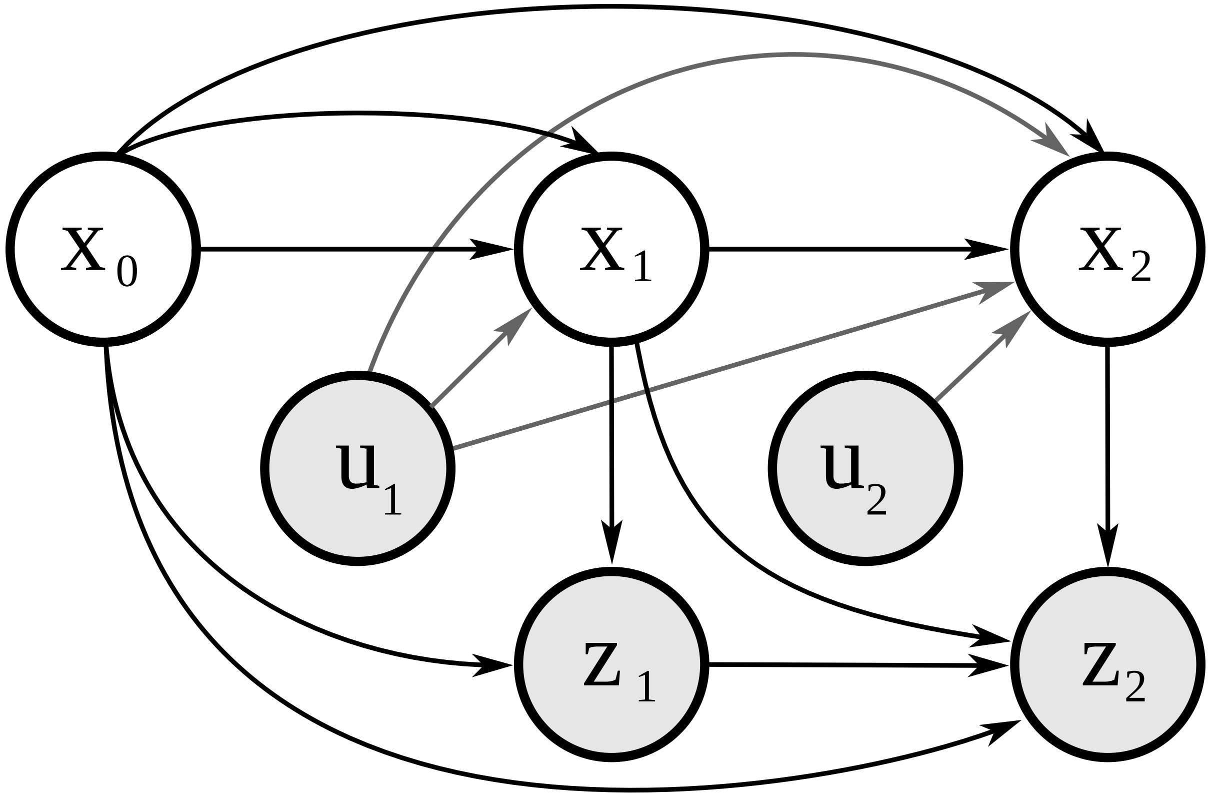
Bayes Network

To represent the conditional dependencies of a set of rv, we can use a Bayes Network.

A Bayes network is a **probabilistic graphical model** that represents conditional dependencies as a directed acyclic graph

- Nodes represent rv
- Edges represent conditional dependencies
  If two nodes are connected, they are conditionally dependent

Bayes Network for Robot Pose



Markov Assumption

Under this assumption, to calculate the posterior for any rv in time slice t, it suffices to know the rvs in that slice and its immediate prior state $t - 1$

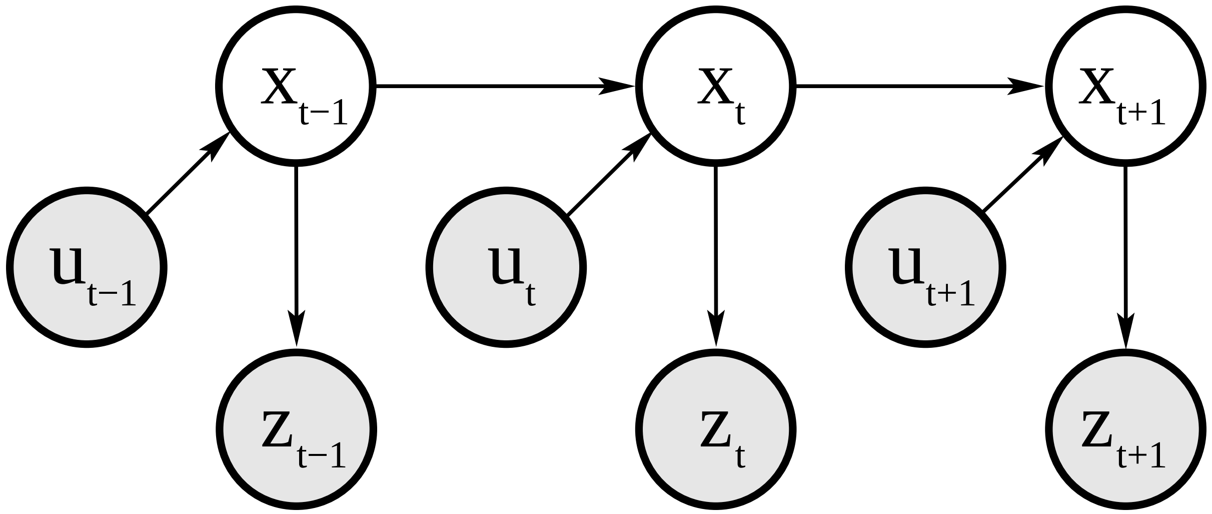
That is, **it assumes independence** between rvs in the current time slice, and all previous ones, except for their immediate previous state

Dynamic Bayes Network

(a.k.a Hidden Markov Model, Two-Timeslice Bayesian Network)

Is a Bayes Network commonly used when modeling time series that use the Markov Assumption

Dynamic Bayes Network for Robot Pose



Dynamic Bayes Network for Robot Pose

For robot localization, we want to know the probability of x at time t given all previous information

$$p(x_t | x_{0:t-1}, z_{1:t}, u_{1:t})$$

Using Bayes' Rule

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

Dynamic Bayes Network for Robot Pose

Under the Markov assumption, we have that

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

which is called the **state transition probability** and shows how the robot's pose changes in time due to input actions.

And, we also have that

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

which is called the **measurement probability**

Dynamic Bayes Network for Robot Pose

Therefore, we have

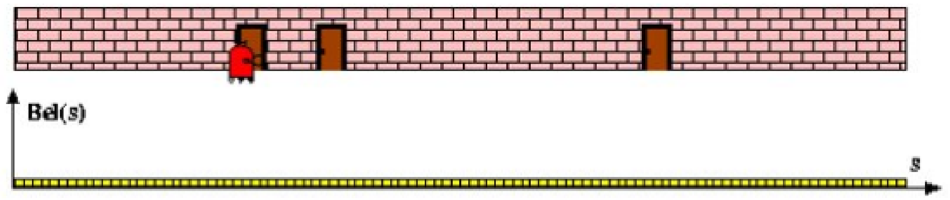
$$p(x_t|x_{0:t-1}, z_{1:t}, u_{1:t}) = p(z_t|x_t)p(x_t|x_{t-1}, u_t)$$

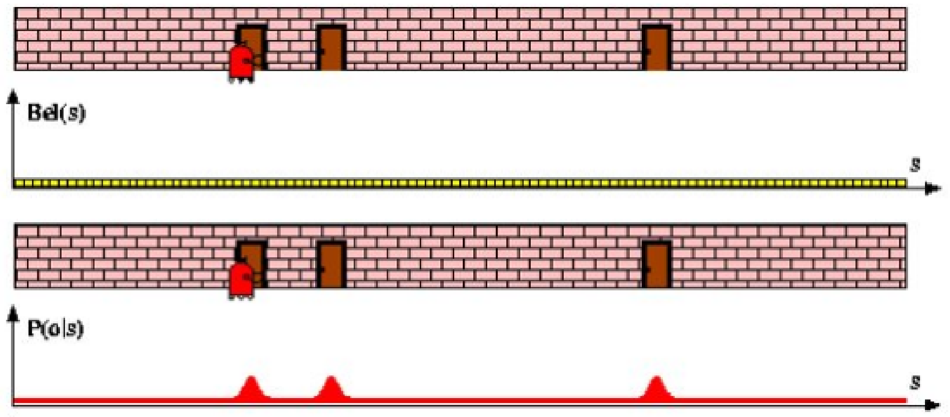
`Note` **From this point onwards, most of our probabilistic modeling for mobile robotics will be centered on either (or both) of these equations.**

※ State transition probability $p(x_t|x_{t-1}, u_t)$
※ Measurement probability $p(z_t|x_t)$

`Supplementary reading`
Probabilistic Robotics - Chapter 2 "Recursive State Estimation"

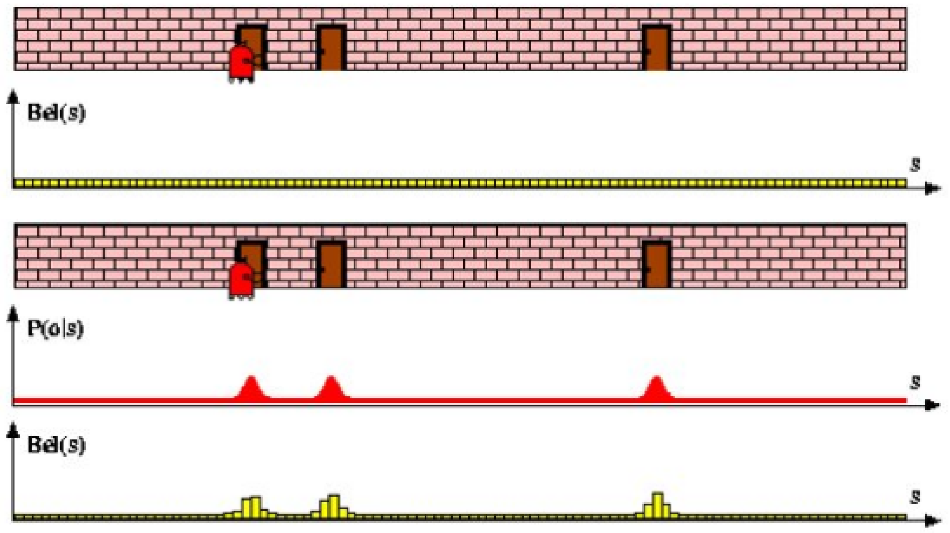
Hallway localization example*



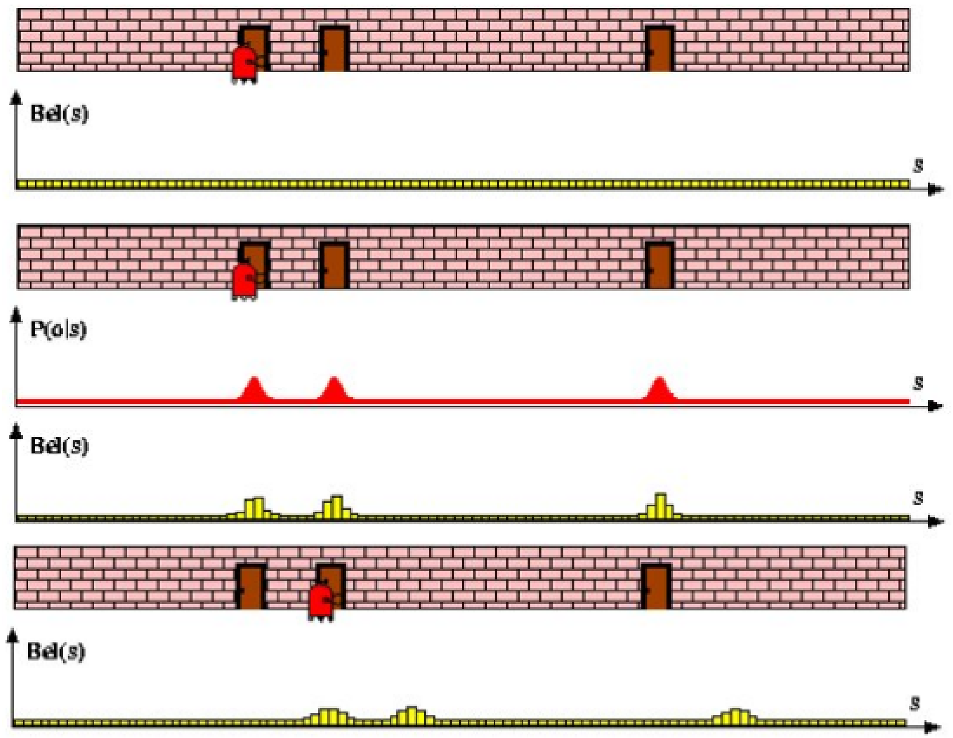


Hallway localization example*

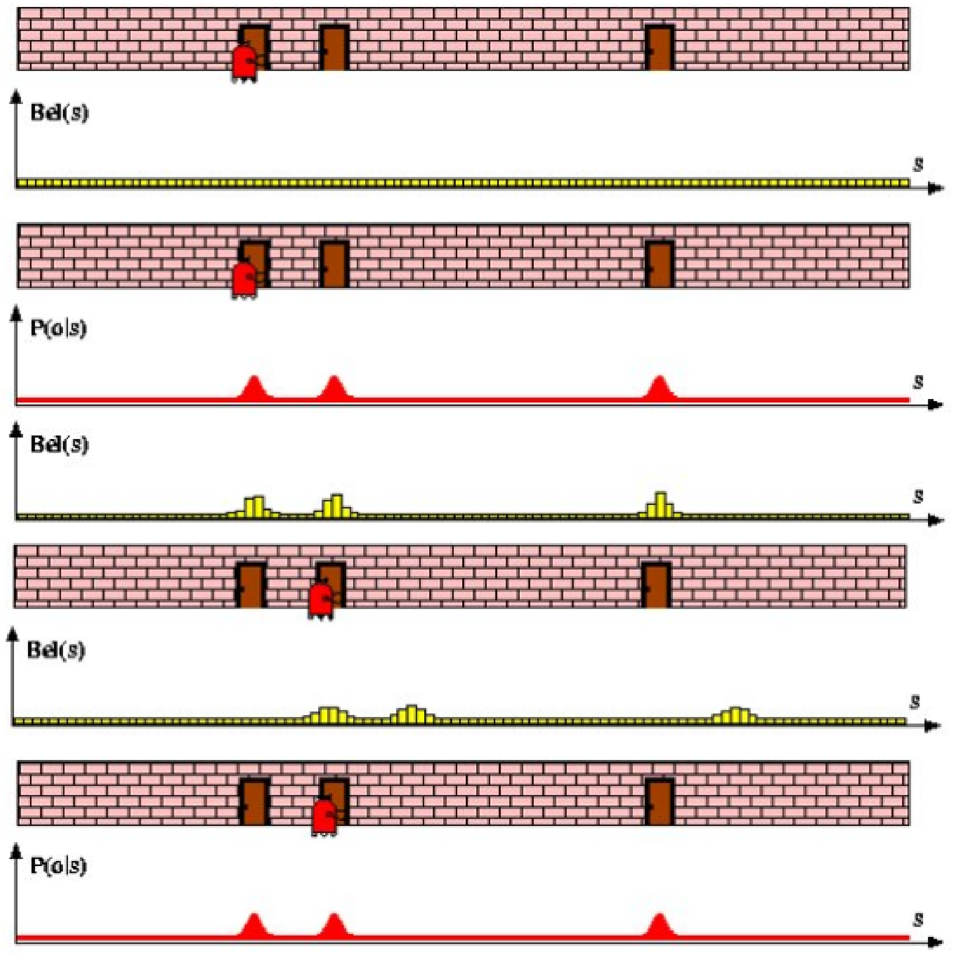
Hallway localization example*



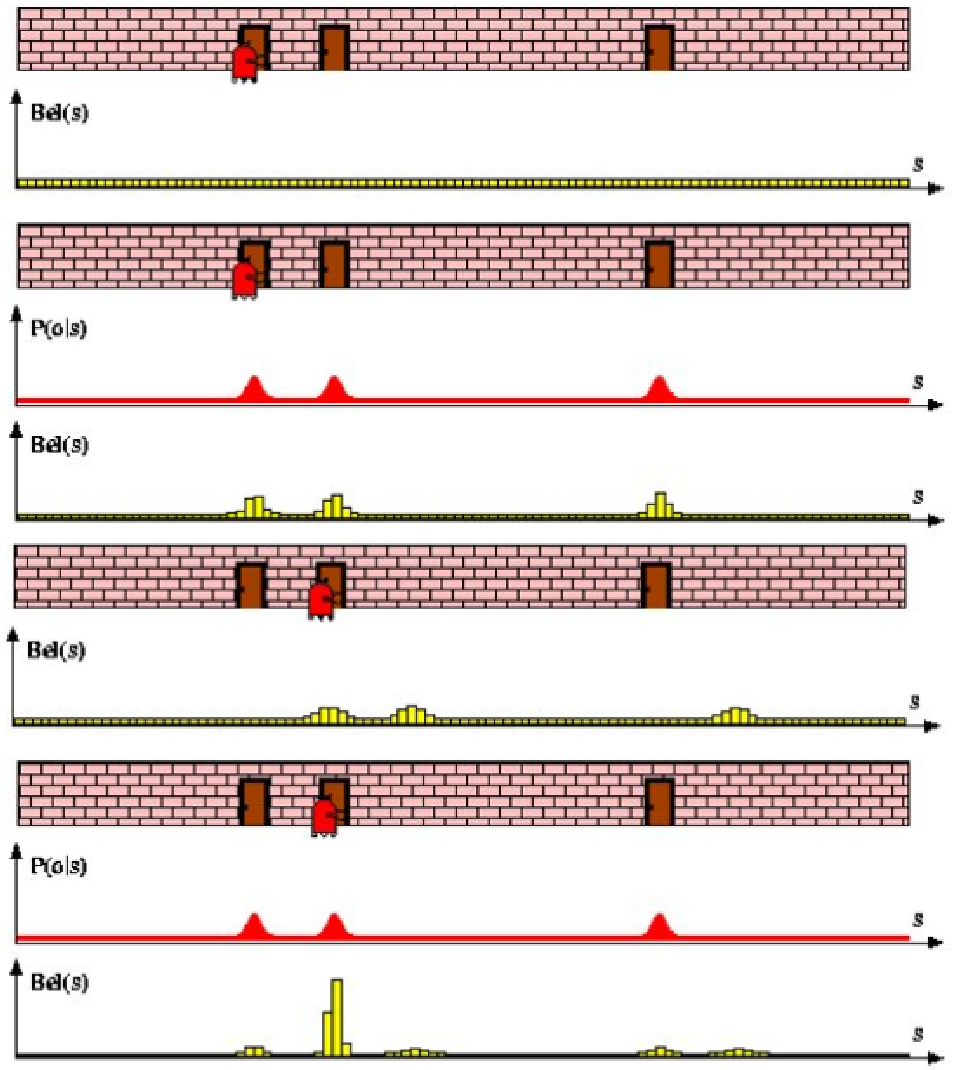
Hallway localization example*



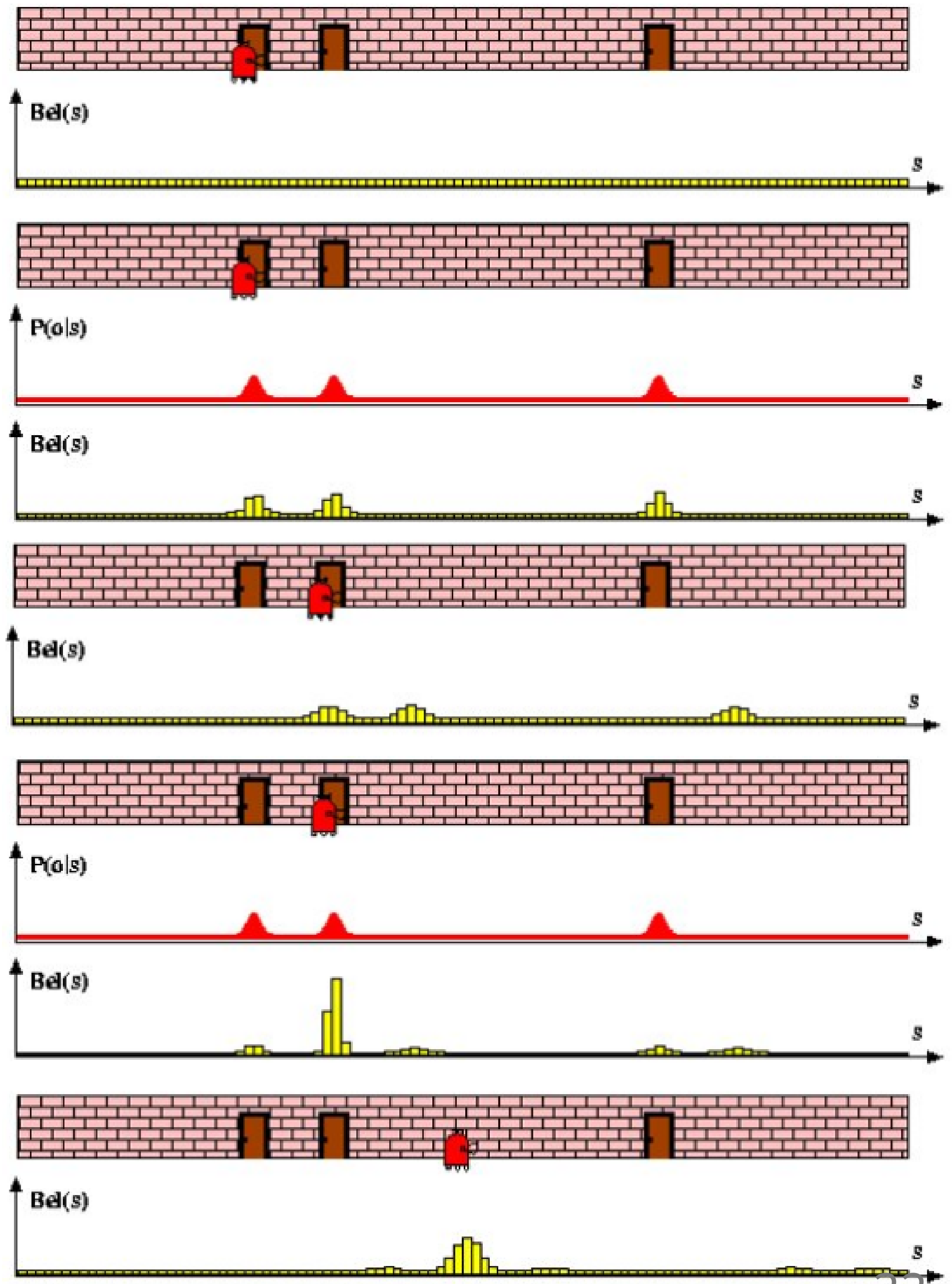
Hallway localization example*



Hallway localization example*



Hallway localization example*

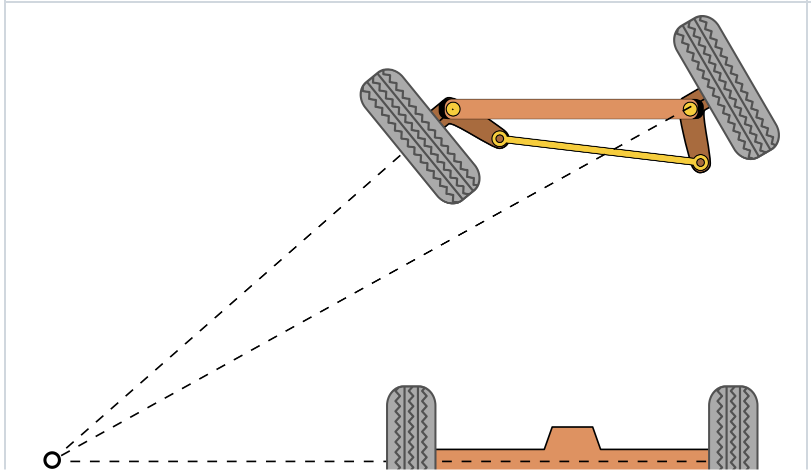
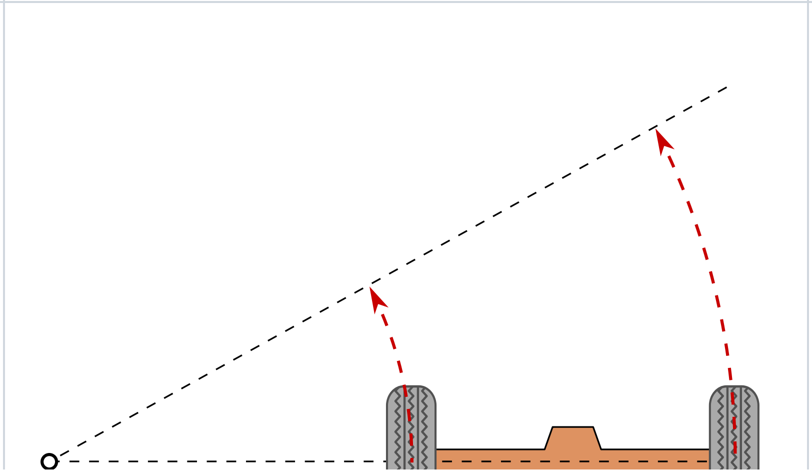
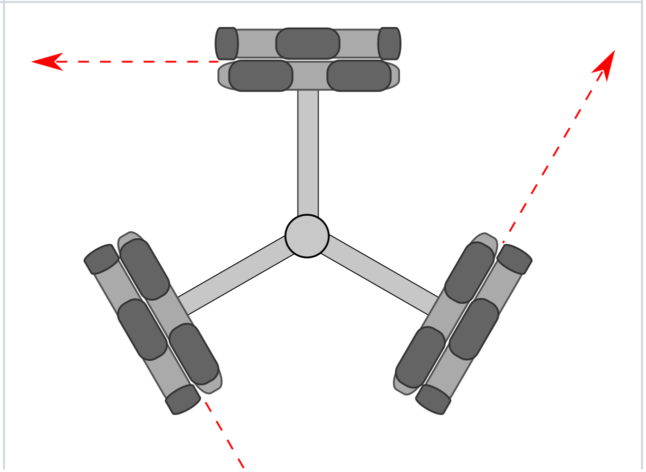


Robot Motion

State transition probability

Wheeled Locomotion

Popular steering mechanisms

Ackermann	Differential	Omnidirectional
		

`Note` We will use the differential steering mechanism for our motion models onwards

Coordinate Systems

Considering robots with planar motion, $[x,\ y,\ \theta]$,

We use two reference frames

1. Global reference frame (Fixed on the map)
   ${}^{G}x$ - robot coordinates in the global frame

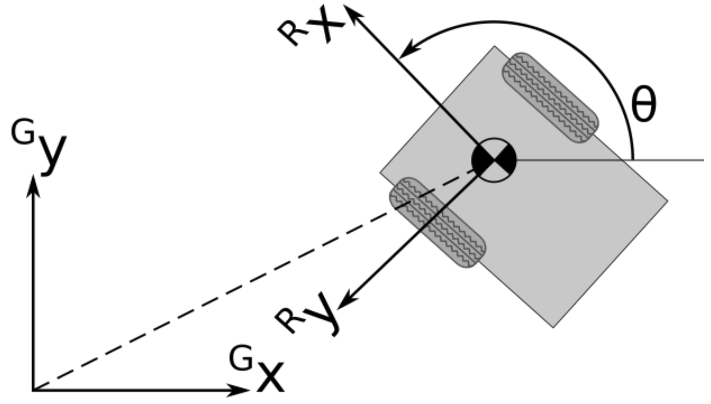
   We will follow the East-North-Up (ENU) Convention

2. Robot local reference frame (Fixed on the robot)
   ${}^{R}x$ - robot coordinates in the robot frame

   We will follow the X-Forward, Z-Up convention

`Note 1` if not specifically stated, $x = {}^{G}x$

`Note 2` This is the same notation ROS uses



Reminder: Transformation

We can transform a vector on one reference to another using the Rotation Matrix ${}^{G}R_{R}$

$$
{}^{G}x = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} {}^{R}x
$$

If you need a refresh on linear transformation watch the series of videos on Khan academy on Linear Transformations, specifically Rotations on R2

Velocity Motion Model

Considers control of the robot through rotational ω and translational v velocity

$$u_t = \begin{bmatrix} v_t \\ \omega_t \end{bmatrix}$$

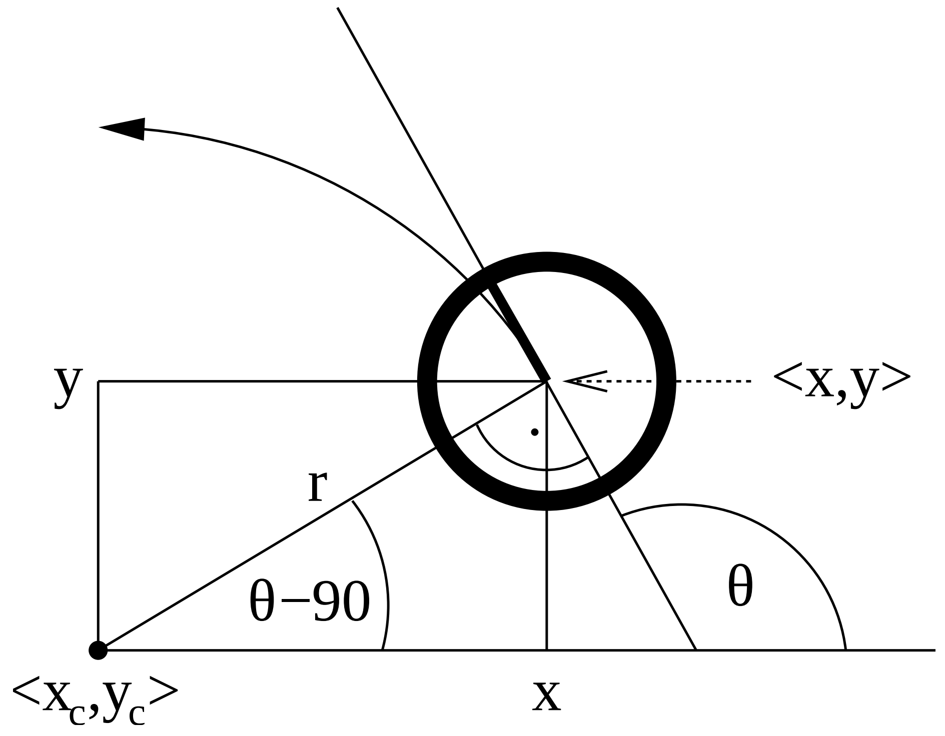
Centre of Rotation

For any robot, if we have a constant velocity applied to the robot $u = (v,\ \omega)$

It can be easily shown that the robot would move (ideally) on a circle of radius $r = |v/\omega|$ as v has to be equal to $r\omega$

Furthermore, we can also show that the centre of this circle on robot coordinates would be on

$$^{R}x_c = \begin{bmatrix} 0 \\ \frac{v}{\omega} \end{bmatrix}$$

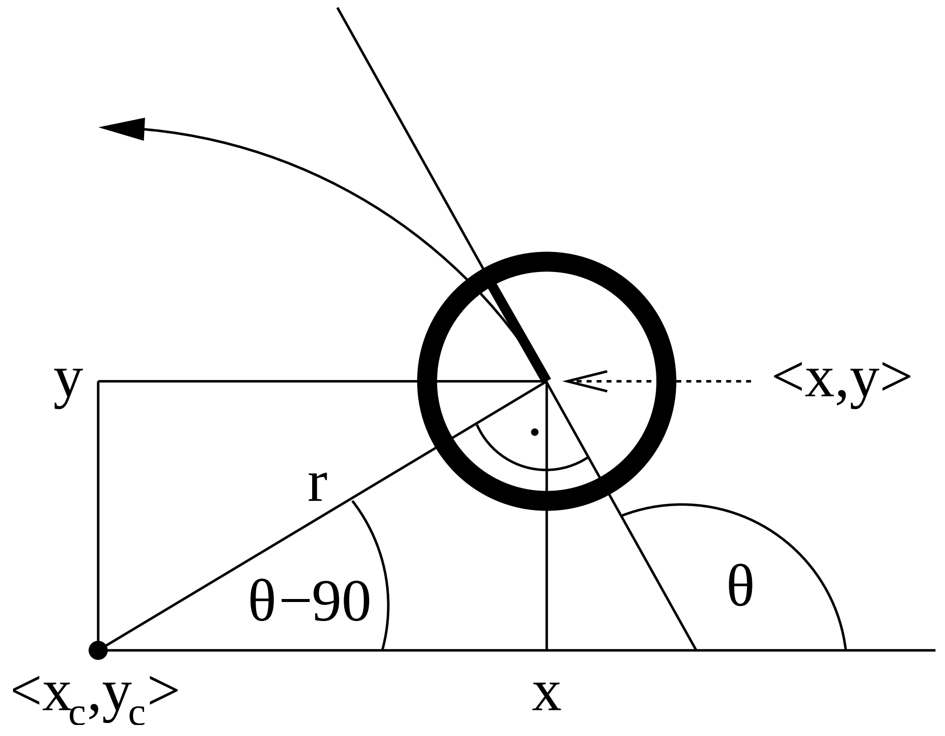


Centre of Rotation

Consider the initial robot's pose as $x = (x,\, y,\, \theta)$, and transforming to global coordinates, we have

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

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



Exact Motion

Considering the constant velocity to the robot $u = (v,\, \omega)$ was applied for some time Δt, the robot would move a distance $w\Delta t$ around the arc formed by the circle with radius r.

Making its final location $x' = (x',\, y',\, \theta')$

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} -\frac{v}{\omega}\sin\theta + \frac{v}{\omega}\sin(\theta + \omega\Delta t) \\ \frac{v}{\omega}\cos\theta - \frac{v}{\omega}\cos(\theta + \omega\Delta t) \\ \omega\Delta t \end{bmatrix}$$

Noise

Now consider independent noise on the command velocities, so

$$\begin{bmatrix} \hat{v} \\ \hat{\omega} \end{bmatrix} = \begin{bmatrix} v \\ \omega \end{bmatrix} + \begin{bmatrix} \varepsilon_{\alpha_1 v^2 + \alpha_2 \omega^2} \\ \varepsilon_{\alpha_3 v^2 + \alpha_4 \omega^2} \end{bmatrix}$$

where ε_{b^2} has a Gaussian distribution with zero-mean and variance b^2

Hence the true linear and rotational velocity equal the command sent (observed) plus some additive white noise (zero-mean)

White noise variance is assumed to be proportional to the linear and rotational velocity commands sent, with non-negative proportional constants (α_1, α_2), and (α_3, α_4) respectively

Noise

The given model assumes an exact circular trajectory, which degenerates the three-dimensional pose $(x,\ y,\ \theta)$ into a two-dimensional manifold $(u,\ v)$.

For better generalization, it is useful to separate robot motion into:

1. circular motion previously described, and then
2. an additional rotation which is assumed to have been generated by an independent rotational velocity with additive white noise

$$\hat{\gamma} = \varepsilon_{\alpha_5 v^2 + \alpha_6 \omega^2}$$

Velocity Motion Model

Replacing on our exact model, we get

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} -\frac{\hat{v}}{\hat{\omega}} \sin\theta + \frac{\hat{v}}{\hat{\omega}} \sin(\theta + \hat{\omega}\Delta t) \\ \frac{\hat{v}}{\hat{\omega}} \cos\theta - \frac{\hat{v}}{\hat{\omega}} \cos(\theta + \hat{\omega}\Delta t) \\ \hat{\omega}\Delta t + \hat{\gamma}\Delta t \end{bmatrix}$$

State transition probability

To compute the state transition probability $p(x'|x,u)$, we do not use our described velocity model to estimate x' given x, u and Δt,

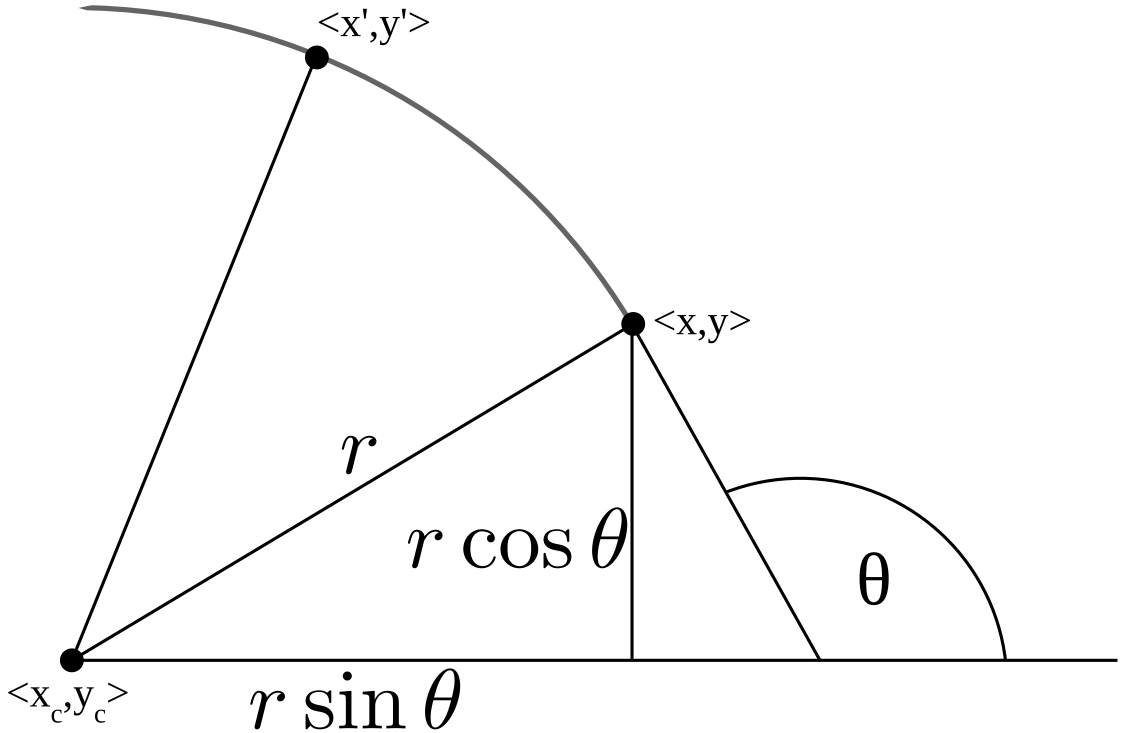
Instead, we do the inverse and compute the noisy controls $\hat{u}$ which would make x transition to x' given Δt.

- The reason is that evaluating the probability of $\hat{u}$ simply requires evaluating three Gaussians while computing the probability of $\hat{x}'$ would require complex evaluations of ratio distributions and wrapped normals

Let's first ignore θ' and compute the required real linear and angular velocities to take $(x,\, y,\, \theta)$ to $(x',\, y')$ in Δt

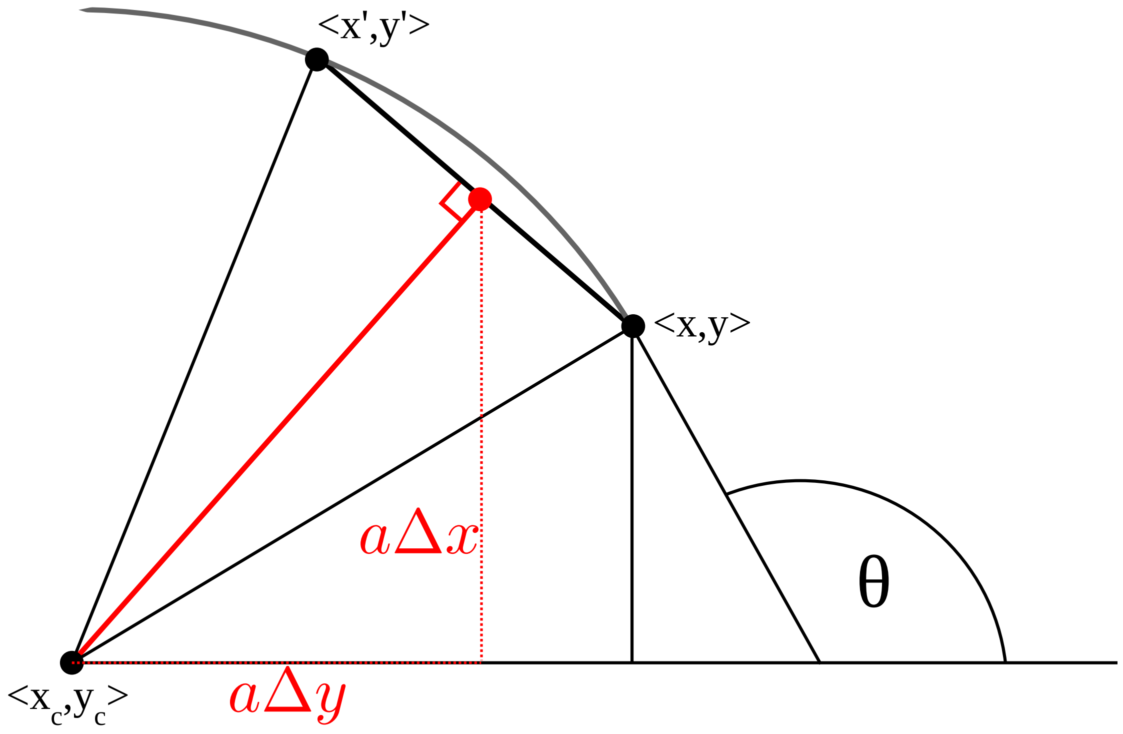
1. From our center of rotation definition we have

$$\begin{bmatrix} x_c \\ y_c \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} -r \sin\theta \\ r \cos\theta \end{bmatrix}$$



2. The center of rotation has to lie on a ray from the halfway point between x and x' that is orthogonal to the line between these coordinates.

$$\begin{bmatrix} x_c \\ y_c \end{bmatrix} = \begin{bmatrix} \frac{x+x'}{2} + a(y - y') \\ \frac{y+y'}{2} + a(x' - x) \end{bmatrix}$$



Solving this set of linear equations gives us

$$a = \frac{1}{2}\frac{(x - x')\cos\theta + (y - y')\sin\theta}{(y - y')\cos\theta - (x - x')\sin\theta}$$

So

$$\begin{aligned}
x_c &= \frac{x + x'}{2} + \frac{1}{2}\frac{(x - x')\cos\theta + (y - y')\sin\theta}{(y - y')\cos\theta - (x - x')\sin\theta}(y - y') \\
y_c &= \frac{y + y'}{2} + \frac{1}{2}\frac{(x - x')\cos\theta + (y - y')\sin\theta}{(y - y')\cos\theta - (x - x')\sin\theta}(x' - x) \\
r &= \sqrt{(x - x_c)^2 + (y - y_c)^2}
\end{aligned}$$

Now we can compute the change in heading direction $\Delta\theta = \hat{\omega}\Delta t$

$$\hat{\omega} = \frac{\Delta\theta}{\Delta t} = \frac{\mathrm{atan2}(y' - y_c,\, x' - x_c) - \mathrm{atan2}(y - y_c,\, x - x_c)}{\Delta t}$$

and v

$$v = r\hat{\omega}$$

Finally, we compute the rotation velocity $\hat{\gamma}$ required for the final rotation to become θ'

$$\hat{\gamma}\Delta t = (\theta' - \theta) - \hat{\omega}\Delta t, \quad \hat{\gamma} = \frac{(\theta' - \theta)}{\Delta t} - \hat{\omega}$$

State transition probability

As we assumed independent noise, we can compute the state transition probability as

$$p(x'|x,u) = \mathcal{N}(\hat{v} \,|\, v, \sigma_v)\, \mathcal{N}(\hat{\omega} \,|\, \omega, \sigma_\omega)\, \mathcal{N}(\hat{\gamma} \,|\, 0, \sigma_\gamma)$$

with

$$\sigma_v = \sqrt{\alpha_1 v^2 + \alpha_2 \omega^2}$$
$$\sigma_\omega = \sqrt{\alpha_3 v^2 + \alpha_4 \omega^2}$$
$$\sigma_\gamma = \sqrt{\alpha_5 v^2 + \alpha_6 \omega^2}$$

Odometry Motion Model

Considers odometry information (change in position) from sensors such as encoders

$$u_t = \begin{bmatrix} \bar{x}' \\ \bar{x} \end{bmatrix}$$

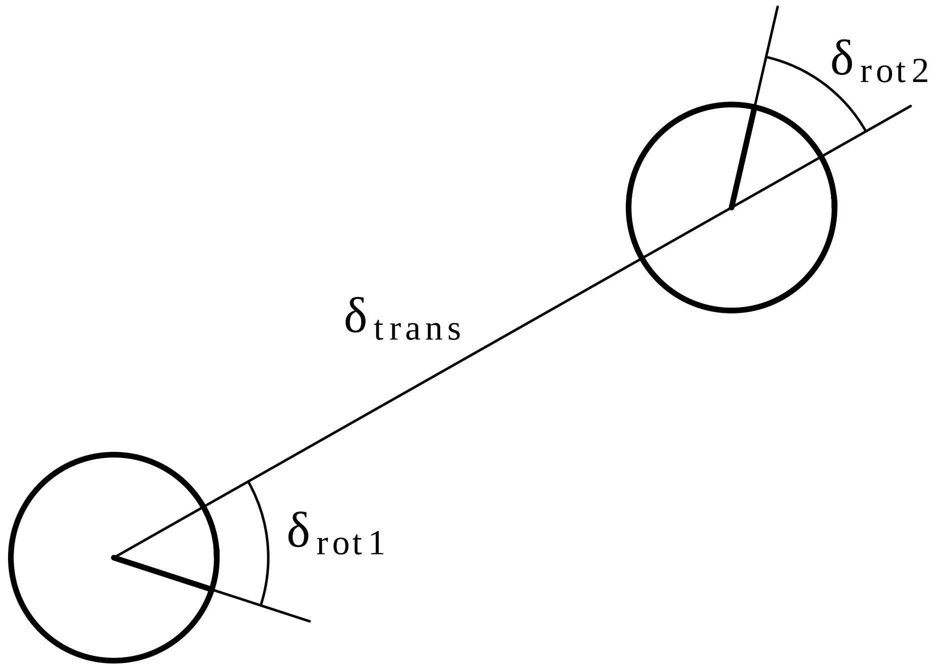
`Note` Odometry information $\bar{x}$ is measured in a robot's coordinate frame, whose transformation to the map frame is not explicitly known

`Note` While odometry is a sensor observation rather than a *control* input, this model is widely used as odometry information is well suited for motion modeling and tends to yield more precise results than velocity

Odometry Motion Model

In the odometry motion model, the motion encoded in the odometry information u is transformed into a sequence of three steps:

1. pure rotation δ_{rot1}
2. pure linear motion (translation) δ_{trans}
3. pure rotation δ_{rot2}



Motions

The initial rotation δ_{rot1} is the necessary rotation for the robot's heading angle and the direction of movement from $\bar{x}$ to $\bar{x}'$ to be the same

$$\bar{\delta}_{rot1} = \mathrm{atan2}(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta}$$

Linear motion δ_{trans} is the straight motion from $\bar{x}$ to $\bar{x}'$

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

The second rotation $\bar{\delta}_{rot2}$ is the necessary rotation so the robot's final heading angle is the same as $\bar{\theta}'$

$$\bar{\delta}_{rot2} = \bar{\theta}' - \bar{\theta} - \bar{\delta}_{rot1}$$

Noise

The model considers independent noise for each motion, so

$$\begin{aligned}
\hat{\delta}_{rot1} &= \bar{\delta}_{rot1} + \varepsilon_{\alpha_1 \delta^2_{rot1} + \alpha_2 \delta^2_{trans}} \\
\hat{\delta}_{trans} &= \bar{\delta}_{trans} + \varepsilon_{\alpha_3 \delta^2_{trans} + \alpha_4 \delta^2_{rot1} + \alpha_4 \delta^2_{rot2}} \\
\hat{\delta}_{rot2} &= \bar{\delta}_{rot2} + \varepsilon_{\alpha_1 \delta^2_{rot2} + \alpha_2 \delta^2_{trans}}
\end{aligned}$$

where ε_{b^2} has a Gaussian distribution with zero-mean and variance b^2 and δ_{rot1}, δ_{trans}, and δ_{rot2} are the motions from x to x'

Odometry Motion Model

Considering noise we get

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \hat{\delta}_{trans} \cos(\theta + \hat{\delta}_{rot1}) \\ \hat{\delta}_{trans} \sin(\theta + \hat{\delta}_{rot1}) \\ \hat{\delta}_{rot1} + \hat{\delta}_{rot2} \end{bmatrix}$$

State transition probability

Similar as for the velocity motion model, to compute the state transition probability $p(x'|x, u)$, we do not compute x' but rather compare the difference in the three motions when using $u = (\bar{x},\ \bar{x}')$ and when using $(x,\ x')$

Analogous to $u = (\bar{x},\ \bar{x}')$ we compute the motions for $(x,\ x')$ as

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

State transition probability

As we assumed independent noise, we can compute the state transition probability as

$$p(x'|x,u) = \mathcal{N}(\bar{\delta}_{rot1} \,|\, \delta_{rot1}, \sigma_{rot1})\, \mathcal{N}(\bar{\delta}_{trans} \,|\, \delta_{trans}, \sigma_{trans})\, \mathcal{N}(\bar{\delta}_{rot2} \,|\, \delta_{rot2}, \sigma_{rot2})$$

with

$$\sigma_{rot1} = \sqrt{\alpha_1 \delta_{rot1}^2 + \alpha_2 \delta_{trans}^2}$$
$$\sigma_{trans} = \sqrt{\alpha_3 \delta_{trans}^2 + \alpha_4 \delta_{rot1}^2 + \alpha_4 \delta_{rot2}^2}$$
$$\sigma_{rot2} = \sqrt{\alpha_1 \delta_{rot2}^2 + \alpha_2 \delta_{trans}^2}$$

Sampling Motion Models

Sampling Motion Models

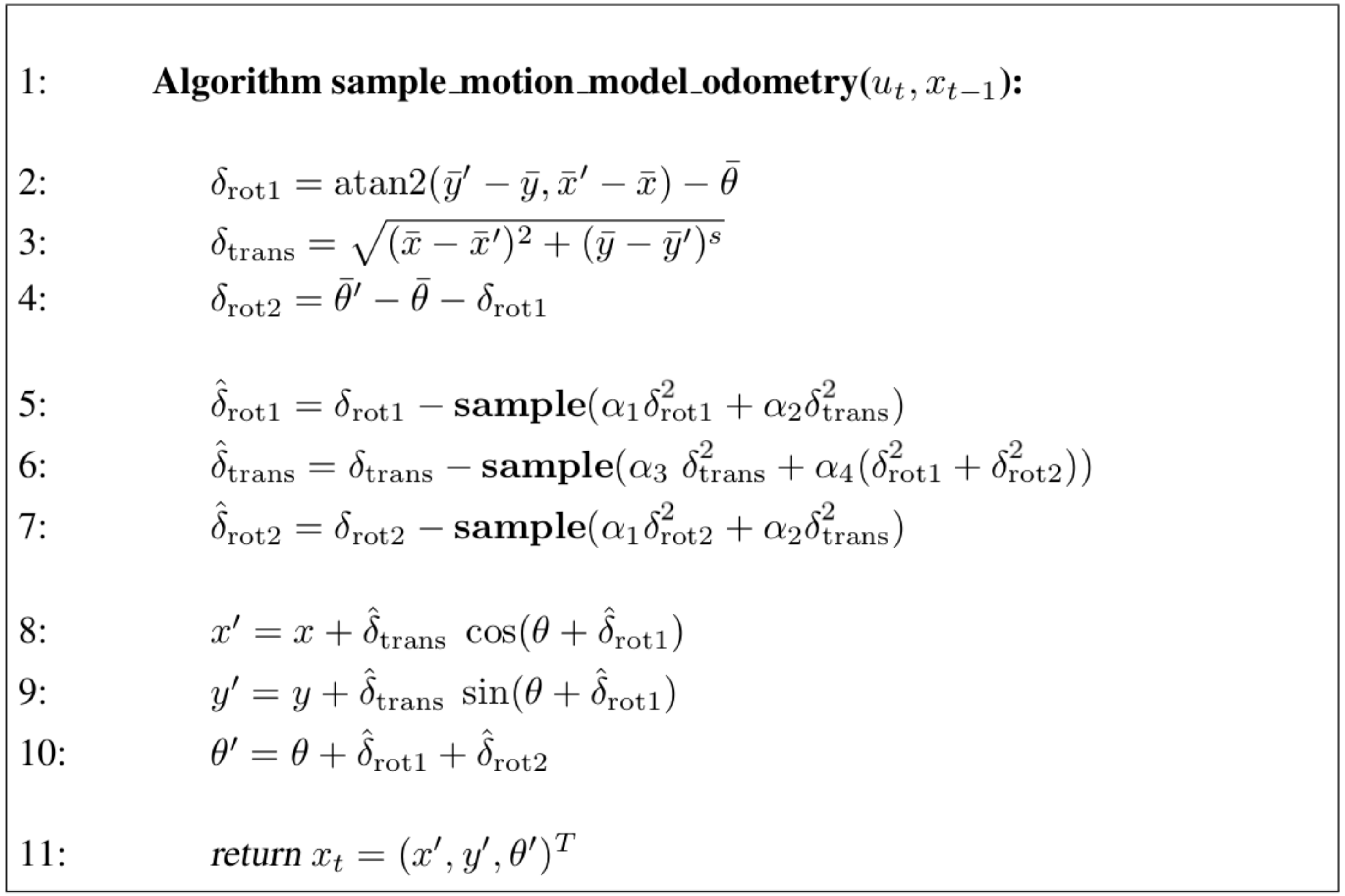
Other than computing state transition probabilities, motion models are often used to sample poses x' from $(x,\, u)$

Sampling poses can be used to understand the distribution of x' given different noise parameters α

Moreover, is fundamental for sample(particle) based localization algorithms that we will study later

For the developed models, as all noise considered is additive white noise over the control states, we only have to use the forward motion model with the noisy control states.

Odometry Motion Model



```
1:  **Algorithm sample_motion_model_odometry**($u_t, x_{t-1}$):

2:      $\delta_{\text{rot1}} = \text{atan2}(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta}$
3:      $\delta_{\text{trans}} = \sqrt{(\bar{x} - \bar{x}')^2 + (\bar{y} - \bar{y}')^s}$
4:      $\delta_{\text{rot2}} = \bar{\theta}' - \bar{\theta} - \delta_{\text{rot1}}$

5:      $\hat{\delta}_{\text{rot1}} = \delta_{\text{rot1}} - \textbf{sample}(\alpha_1 \delta^2_{\text{rot1}} + \alpha_2 \delta^2_{\text{trans}})$
6:      $\hat{\delta}_{\text{trans}} = \delta_{\text{trans}} - \textbf{sample}(\alpha_3\ \delta^2_{\text{trans}} + \alpha_4 (\delta^2_{\text{rot1}} + \delta^2_{\text{rot2}}))$
7:      $\hat{\delta}_{\text{rot2}} = \delta_{\text{rot2}} - \textbf{sample}(\alpha_1 \delta^2_{\text{rot2}} + \alpha_2 \delta^2_{\text{trans}})$

8:      $x' = x + \hat{\delta}_{\text{trans}}\ \cos(\theta + \hat{\delta}_{\text{rot1}})$
9:      $y' = y + \hat{\delta}_{\text{trans}}\ \sin(\theta + \hat{\delta}_{\text{rot1}})$
10:     $\theta' = \theta + \hat{\delta}_{\text{rot1}} + \hat{\delta}_{\text{rot2}}$

11:     *return* $x_t = (x', y', \theta')^T$
```

$\textbf{sample}(b^2)$ gets a value from a zero-mean Gaussian distribution with $\sigma = b$