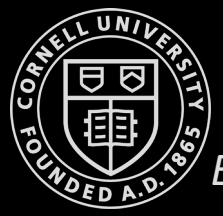
ECE 4960

Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

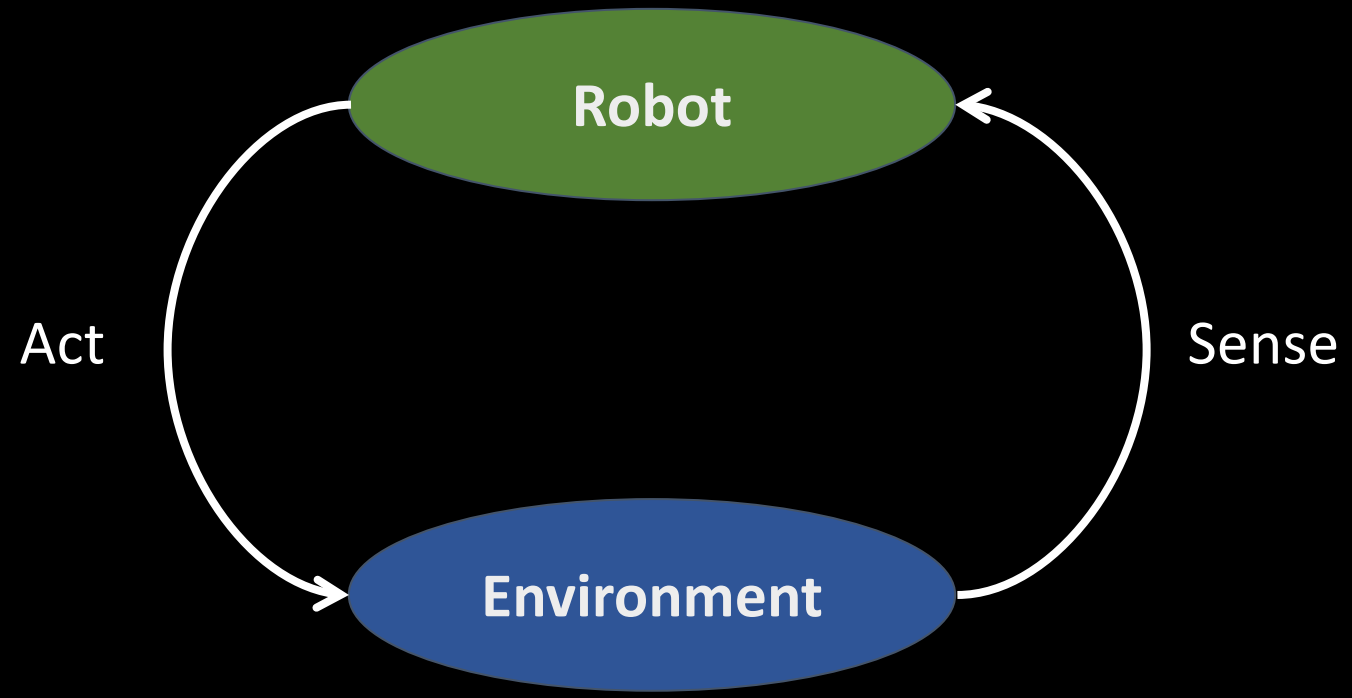
Vivek Thangavelu
vs353@cornell.edu

Fast Robots

Bayes Filter2
and
Motion models



Robot-Environment Interaction



- Two fundamental types of interaction between a robot and its environment:
  - Sensor Measurements/Observations
  - Control Actions

Markov Assumption

The Markov assumption postulates that past and future data are independent if one knows the current state

- State generative model
  - $p(x_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t}) = p(x_t \mid x_{t-1}, u_t)$
- Measurement generative model
  - $p(z_t \mid x_{0:t}, z_{1:t-1}, u_{1:t}) = p(z_t \mid x_t)$

Andrey Markov (1856–1922) was a Russian mathematician best known for his work on stochastic processes

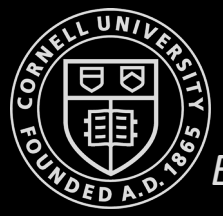
Robot Belief

- A robot maintains an internal representation of itself and the environment
  - "Posterior conditional probability distributions"
- The belief of a robot is the posterior distribution over the state of the environment, given all past sensor measurements and all past controls
- Belief over a state variable x_t is denoted by $bel(x_t)$ which is an abbreviation for

$$bel(x_t) = p(x_t | z_{1:t}, u_{1:t})$$

- Occasionally it prove useful in our probabilistic algorithms to define a (prior) belief before incorporating the latest measurement z_t

$$\overline{bel}(x_t) = p(x_t | z_{1:t-1}, u_{1:t})$$



Robot-Environment Model

+

Markov Assumption

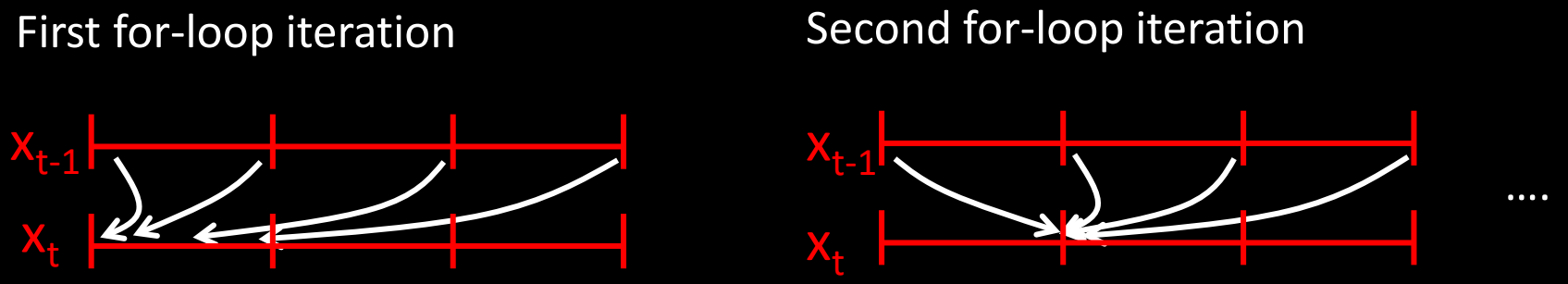
+

Bayes Theorem

=

Bayes Filter

Bayes Filter



1. **Algorithm Bayes_Filter** $(bel(x_{t-1}), u_t, z_t)$:
2. for all x_t do
3. $\overline{bel}(x_t) = \sum_{x_{t-1}} p(x_t|u_t, x_{t-1})\ bel(x_{t-1})$ (Prediction step)
4. $bel(x_t) = \eta\, p(z_t|x_t)\ \overline{bel}(x_t)$ (Update step)
5. endfor
6. return $bel(x_t)$

Transition probability /action model

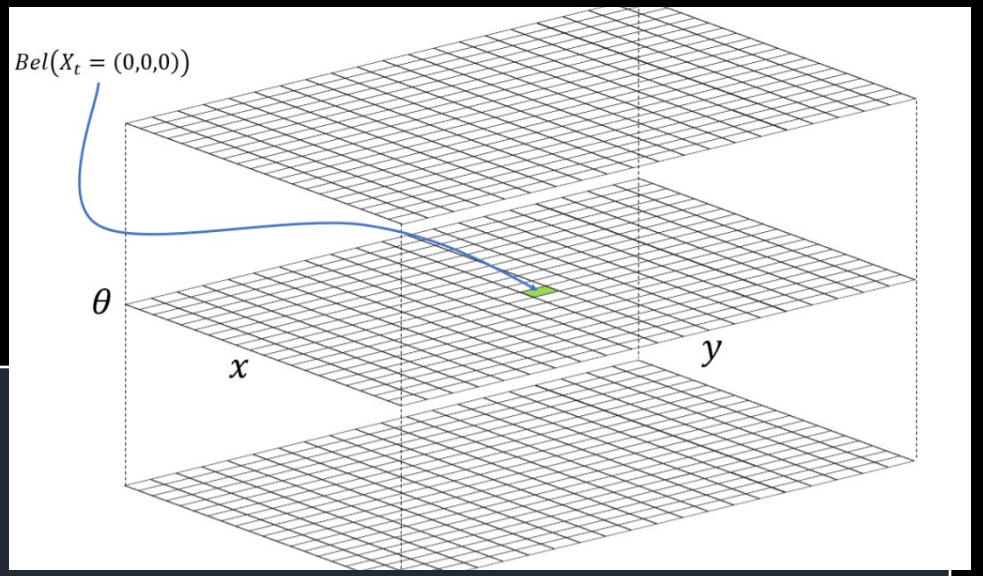
prior

Measurement Probability / Sensor Model

Normalization constant

Bayes Filter

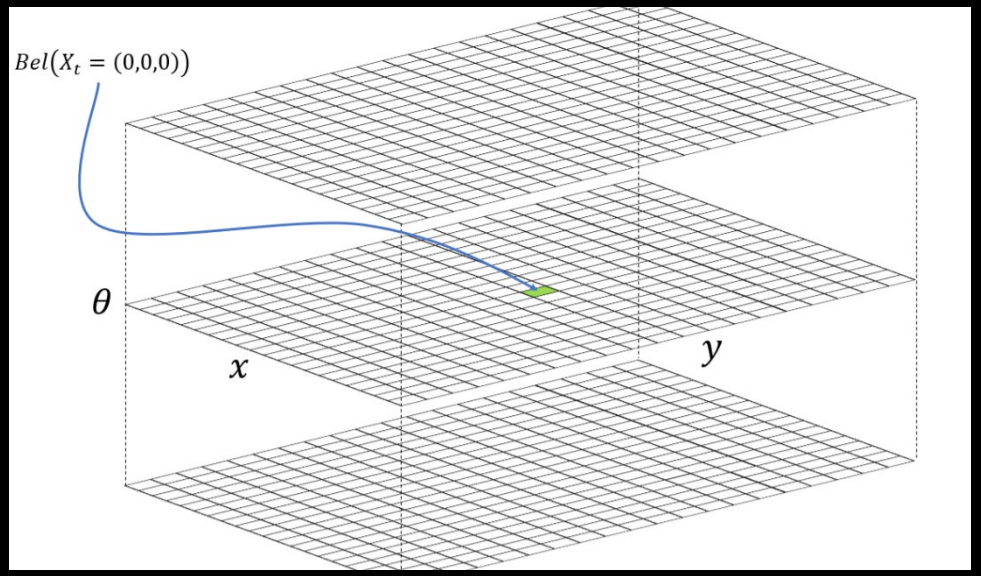
This is a lot of computation!



1. **Algorithm Bayes_Filter** $(bel(x_{t-1}), u_t, z_t)$:
2. for all x_t do
3. $\overline{bel}(x_t) = \sum_{x_{t-1}} p(x_t|u_t, x_{t-1})\ bel(x_{t-1})$ (Prediction step)
4. $bel(x_t) = \eta\, p(z_t|x_t)\ \overline{bel}(x_t)$ (Update step)
5. endfor
6. return $bel(x_t)$

Violations of Markov Assumption

This is a lot of computation!



- Typical violations of the Markov assumption:
  - Unmodeled dynamics in the environment not included in x_t
  - Inaccuracies in the probabilistic models $p(z_t \mid x_t)$ and $p(x_t \mid u_t, x_{t-1})$
  - Approximation errors when representing belief functions
- Incomplete state representations are often preferable to reduce computational complexity of the Bayes filter algorithm
- *In practice Bayes filters have been found to be surprisingly robust to such violations*

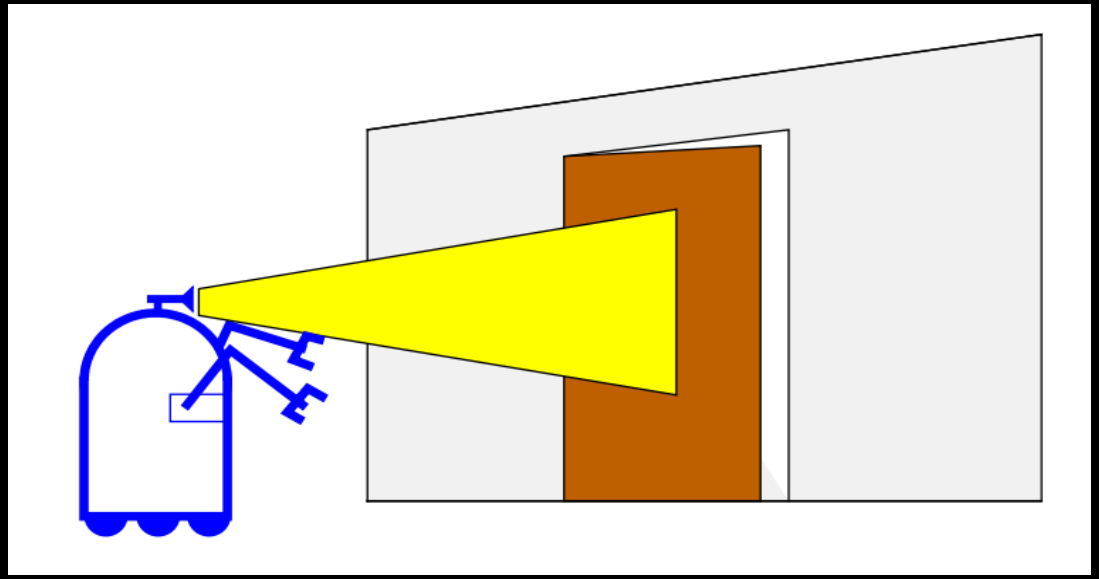
Bayes Filter

Example 1

Bayes Filter - Example 1

- A robot can "*observe*" a door through its sensor and can interact with it by "*pushing*"
- The door may be in one of two states
  - *open* or *closed*
- At any given time, the robot can either
  - *push* or *do_nothing*
- The sensors and the actuators on the robot are noisy

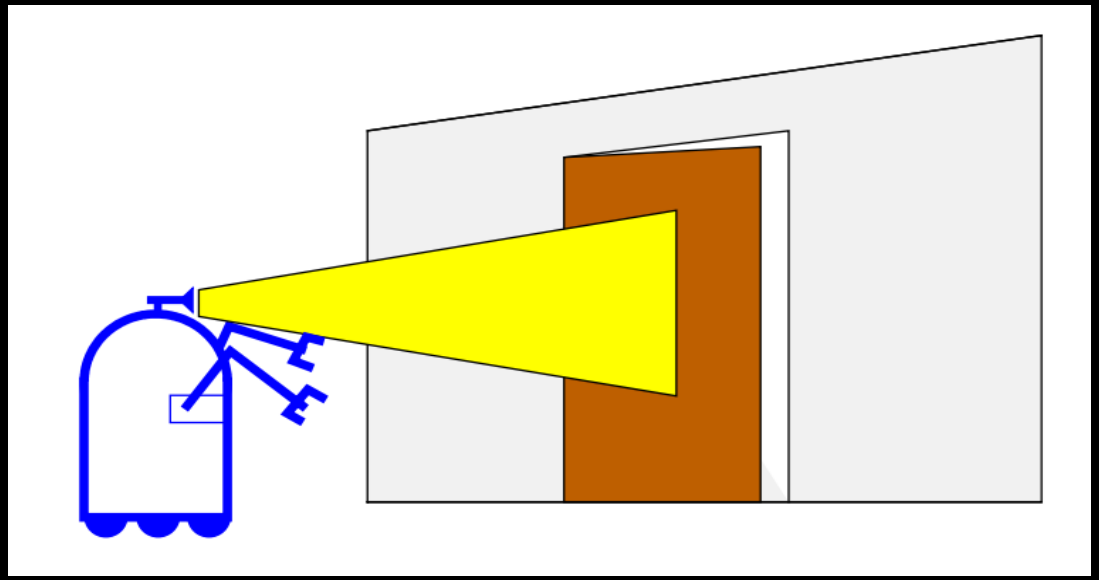
- The probability that the robot can sense an *open* door is 0.6
- The probability that the robot can sense a *closed* door is 0.8
- After a *push* action, probability that a door is *open* if it was previously open is 1
- After a *push* action, probability that a door is *open* if it was previously closed is 0.8
- If the robot *does nothing*, the door continues to be in the previous state



Bayes Filter - Example 1

- Measurement model
  - $p(Z_t = closed \mid X_t = is\_closed) = 0.8$
  - $p(Z_t = open \mid X_t = is\_closed) = 0.2$
  - $p(Z_t = closed \mid X_t = is\_open) = 0.4$
  - $p(Z_t = open \mid X_t = is\_open) = 0.6$

- The probability that the robot can sense an *open* door is 0.6
- The probability that the robot can sense a *closed* door is 0.8
- After a *push* action, probability that a door is *open* if it was previously open is 1
- After a *push* action, probability that a door is *open* if it was previously closed is 0.8
- If the robot *does nothing*, the door continues to be in the previous state



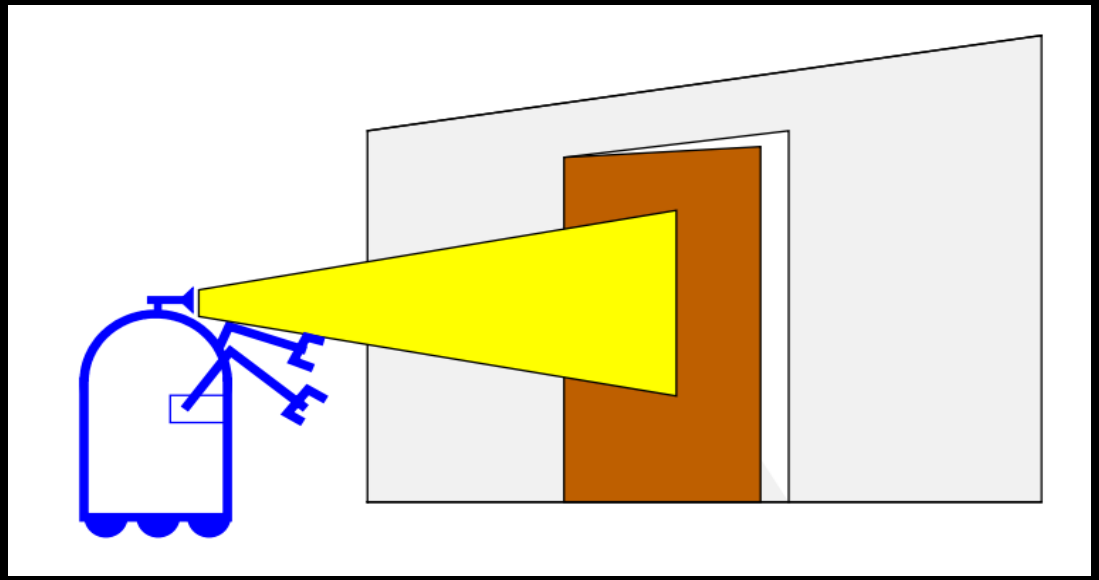
Bayes Filter - Example 1

Action model

- $p(X_t = is\_closed | U_t = do\_nothing, X_{t-1} = is\_closed) = 1$
- $p(X_t = is\_open | U_t = do\_nothing, X_{t-1} = is\_closed) = 0$
- $p(X_t = is\_closed | U_t = do\_nothing, X_{t-1} = is\_open) = 0$
- $p(X_t = is\_open | U_t = do\_nothing, X_{t-1} = is\_open) = 1$

- $p(X_t = is\_closed | U_t = push, X_{t-1} = is\_closed) = 0.2$
- $p(X_t = is\_open | U_t = push, X_{t-1} = is\_closed) = 0.8$
- $p(X_t = is\_closed | U_t = push, X_{t-1} = is\_open) = 0$
- $p(X_t = is\_open | U_t = push, X_{t-1} = is\_open) = 1$

- The probability that the robot can sense an *open* door is 0.6
- The probability that the robot can sense a *closed* door is 0.8
- After a *push* action, probability that a door is *open* if it was previously open is 1
- After a *push* action, probability that a door is *open* if it was previously closed is 0.8
- If the robot *does nothing*, the door continues to be in the previous state



Bayes Filter - Example 1

Initial Conditions

$$bel(X_0 = closed) = bel(X_0 = open) = 0.5$$

Measurement Probability

$$p(Z_t = closed | X_t = is\_closed) = 0.8$$
$$p(Z_t = open | X_t = is\_closed) = 0.2$$
$$p(Z_t = closed | X_t = is\_open) = 0.4$$
$$p(Z_t = open | X_t = is\_open) = 0.6$$

Control Action/Transition Probability

$$p(X_t = is\_closed | U_t = do\_nothing, X_{t-1} = is\_closed) = 1$$
$$p(X_t = is\_open | U_t = do\_nothing, X_{t-1} = is\_closed) = 0$$
$$p(X_t = is\_closed | U_t = do\_nothing, X_{t-1} = is\_open) = 0$$
$$p(X_t = is\_open | U_t = do\_nothing, X_{t-1} = is\_open) = 1$$

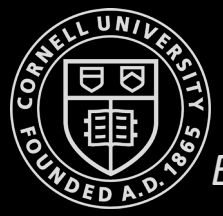
$$p(X_t = is\_closed | U_t = push, X_{t-1} = is\_closed) = 0.2$$
$$p(X_t = is\_open | U_t = push, X_{t-1} = is\_closed) = 0.8$$
$$p(X_t = is\_closed | U_t = push, X_{t-1} = is\_open) = 0$$
$$p(X_t = is\_open | U_t = push, X_{t-1} = is\_open) = 1$$

Bayes Filter - Example 1

$$u_1 = do\_nothing \; and \; z_1 = sense\_open$$

Incorporating action

$$\overline{bel}(x_1) = \sum_{x_0} p(x_1|u_1, x_0) \; bel(x_0)$$



Bayes Filter - Example 1

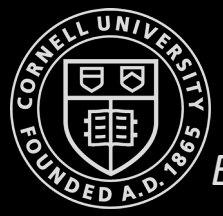
$$u_1 = do\_nothing \text{ and } z_1 = sense\_open$$

Incorporating action

$$\overline{bel}(x_1) = \sum_{x_0} p(x_1|u_1, x_0)\, bel(x_0)$$

$$= p(x_1|U_1 = do\_nothing, X_0 = is\_open)\, bel(X_0 = is\_open)$$

$$+ p(x_1|U_1 = do\_nothing, X_0 = is\_closed)\, bel(X_0 = is\_closed)$$



Bayes Filter - Example 1

$$u_1 = do\_nothing \text{ and } z_1 = sense\_open$$

Incorporating action

$$\overline{bel}(x_1) = \sum_{x_0} p(x_1|u_1, x_0)\, bel(x_0)$$

$$= p(x_1|U_1 = do\_nothing, X_0 = is\_open)\, bel(X_0 = is\_open)$$

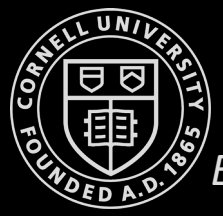
$$+ p(x_1|U_1 = do\_nothing, X_0 = is\_closed)\, bel(X_0 = is\_closed)$$

For the hypothesis $X_1 = is\_closed$:

$$\overline{bel}(X_1 = is\_closed) = p(X_1 = is\_closed|U_1 = do\_nothing, X_0 = is\_open)\, bel(X_0 = is\_open)$$

$$+ p(X_1 = is\_closed|U_1 = do\_nothing, X_0 = is\_closed)\, bel(X_0 = is\_closed)$$

$$= 0 \times 0.5 + 1 \times 0.5 = 0.5$$



Bayes Filter - Example 1

$u_1 = do\_nothing \; and \; z_1 = sense\_open$

Incorporating measurement

$\overline{bel}(X_1 = is\_open) = 0.5$

$\overline{bel}(X_1 = is\_closed) = 0.5$

$bel(x_1) = \eta \, p(Z_1 = sense\_open | \, x_1) \overline{bel}(x_1)$

For two possible cases, $X_1 = is\_open$ and $X_1 = is\_closed$, we get

$$bel(X_1 = is\_open) = \eta \, p(Z_1 = sense\_open \, | X_1 = is\_open) \, \overline{bel}(X_1 = is\_open)$$
$$= \eta \times 0.6 \times 0.5 = \eta \, 0.3$$

$$bel(X_1 = is\_closed) = \eta \, p(Z_1 = sense\_open \, | X_1 = is\_closed) \, \overline{bel}(X_1 = is\_closed)$$
$$= \eta \times 0.2 \times 0.5 = \eta \, 0.1$$

Bayes Filter - Example 1

$$u_1 = do\_nothing \text{ and } z_1 = sense\_open$$

Incorporating measurement

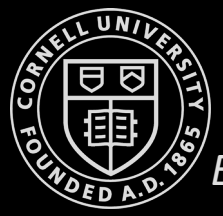
$$bel(X_1 = is\_open) = \eta\ 0.3$$
$$bel(X_1 = is\_closed) = \eta\ 0.1$$

The normalizer η is now calculated as follows:

$$\eta = (0.3 + 0.1)^{-1} = 2.5$$

$$bel(X_1 = is\_closed) = \eta 0.1 = 0.25$$
$$bel(X_1 = is\_open) = \eta 0.3 = 0.75$$

Better than initial belief at time t=0!



Bayes Filter - Example 1

$$u_2 = push \; and \; z_2 = sense\_open$$

Prediction update:

$$\overline{bel}(X_2 = is\_closed) = 0 \times 0.75 + 0.2 \times 0.25 = 0.05$$

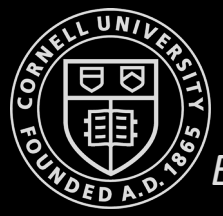
$$\overline{bel}(X_2 = is\_open) = 1 \times 0.75 + 0.8 \times 0.25 = 0.95$$

Measurement Update:

$$bel(X_2 = is\_closed) = 0 \times 0.2 \times 0.05 \simeq 0.017$$

$$bel(X_2 = is\_open) = \eta \times 0.6 \times 0.95 \simeq 0.983$$

Waaaay better than the initial belief at time t=0!



Summary of Bayes Filter

- The robot is modeled as performing a series of alternating measurements and actions
- Given:
  - Sensor model $p(z|x)$
  - Action model $p(x|u, x_{t-1})$
  - Initial Conditions $p(x_0)$
- To compute:
  - Estimate state x of a dynamical system
  - Posterior of the state (Belief):
    $Bel(x_t) = p(x_t|u_1, z_1, \dots, u_t, z_t)$

1. **Algorithm Bayes_Filter** $(bel(x_{t-1}), u_t, z_t)$:
2. for all x_t do
3. $\overline{bel}(x_t) = \sum_{x_{t-1}} p(x_t|u_t, x_{t-1})\ bel(x_{t-1})$
4. $bel(x_t) = \eta\, p(z_t|x_t)\ \overline{bel}(x_t)$
5. endfor
6. return $bel(x_t)$

Summary of Bayes Filter

- Prediction Step:
  - Incorporating action, which **increases** uncertainty
  - Compute $\overline{bel}(x_t) = p(x_t|z_{1:t-1}, u_{1:t})$
  - Requires Action Model: $p(x_t|u_t, x_{t-1})$
- Measurement/Update Step:
  - Incorporating measurement, which **decreases** uncertainty
  - Compute $bel(x_t) = p(x_t|z_{1:t}, u_{1:t})$
  - Requires Sensor Model: $\mathrm{p}(z_t|x_t)$

1. **Algorithm Bayes_Filter** $(bel(x_{t-1}), u_t, z_t)$:
2. for all x_t do
3. $\overline{bel}(x_t) = \sum_{x_{t-1}} p(x_t|u_t, x_{t-1})\ bel(x_{t-1})$
4. $bel(x_t) = \eta\, p(z_t|x_t)\ \overline{bel}(x_t)$
5. endfor
6. return $bel(x_t)$

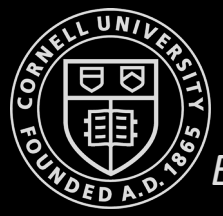
Probabilistic Motion Model

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

Bayes Filter

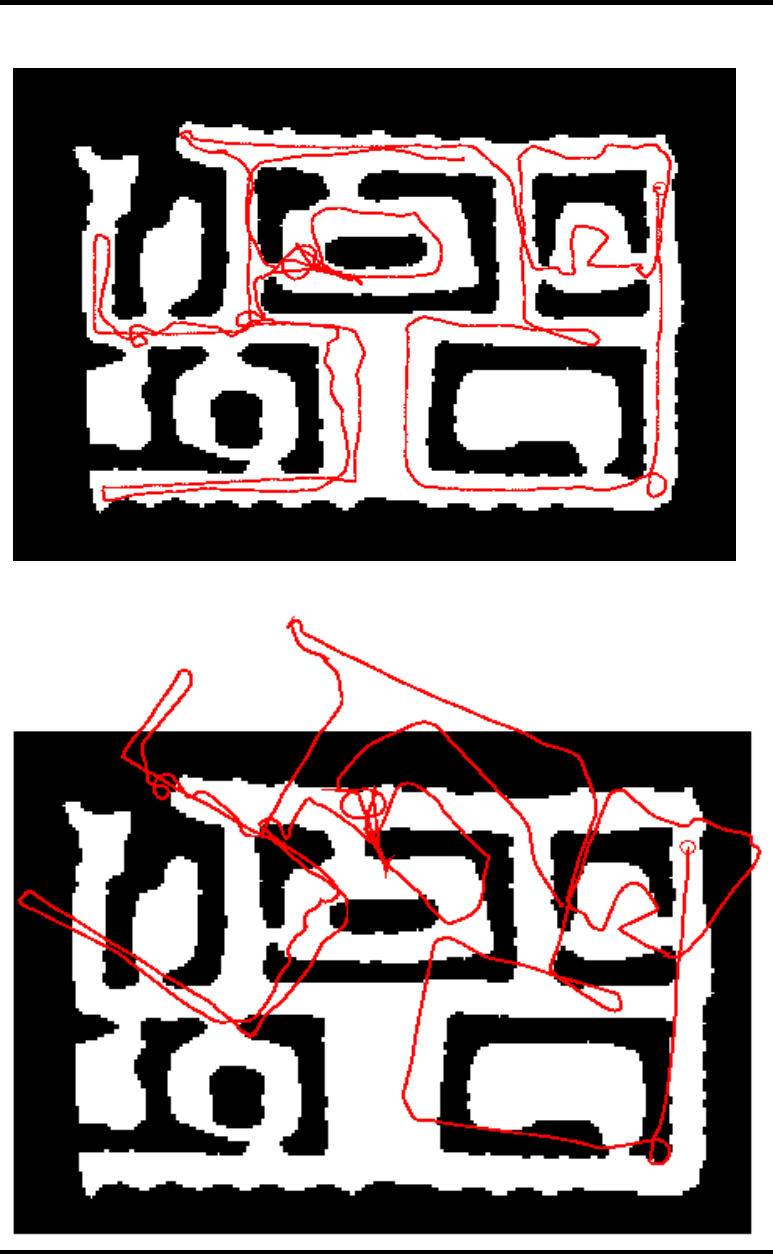
1. **Algorithm Bayes_Filter** $(bel(x_{t-1}), u_t, z_t)$:
2. for all x_t do
3. $\overline{bel}(x_t) = \sum_{x_{t-1}} p(x_t|u_t, x_{t-1})\ bel(x_{t-1})$ (Prediction step)
4. $bel(x_t) = \eta\, p(z_t|x_t)\ \overline{bel}(x_t)$
5. endfor
6. return $bel(x_t)$

Transition probability /action model



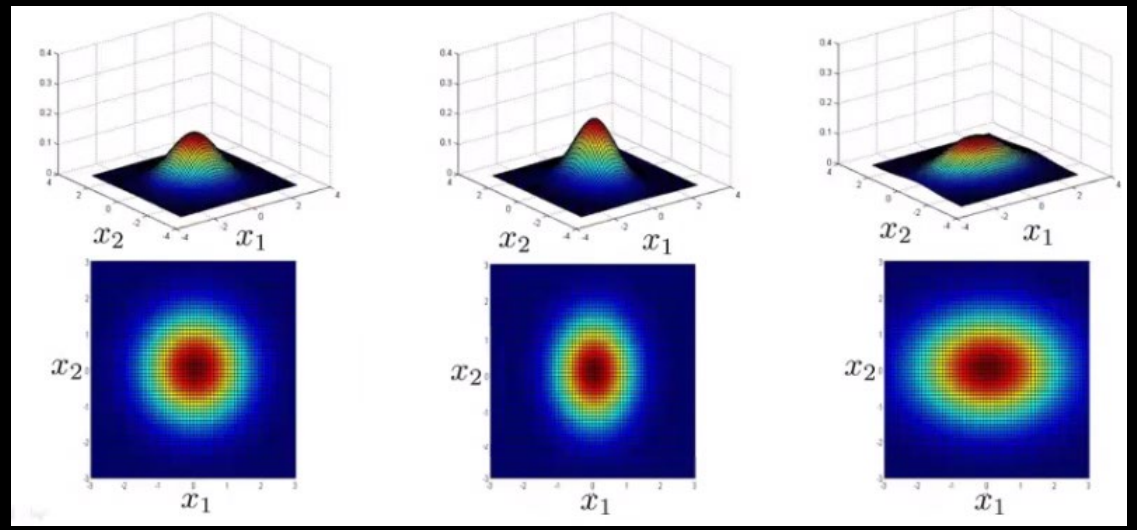
Robot Motion

- Mobile robots on a plane
  - Robot pose $x_t = (x, y, \theta)^T$
- Robot motion is inherently uncertain
- How can we model this uncertainty?
  - Velocity model
  - Odometry model

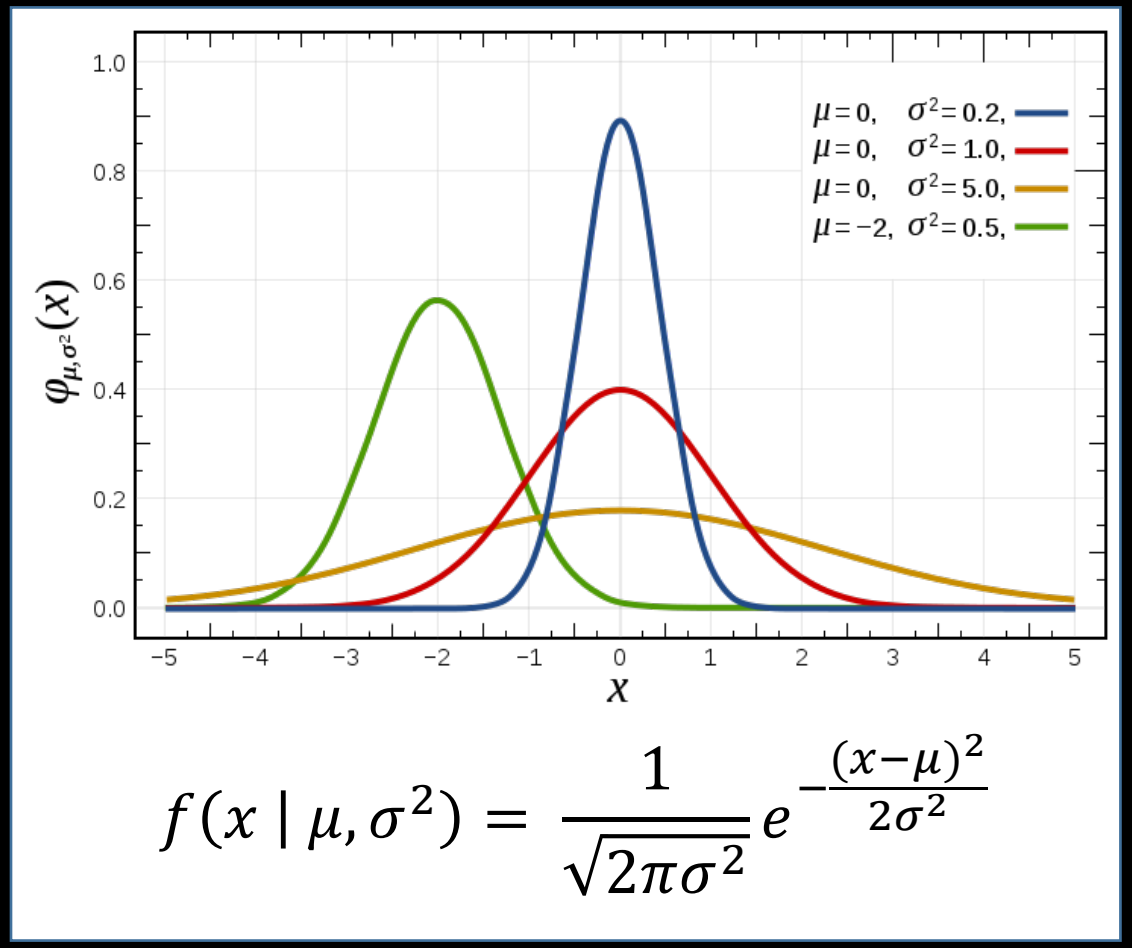


Quick Detour: Gaussian Distribution

- Or "normal distribution" or "bell curve"
- Defined by two parameters:
  - mean μ
  - standard deviation σ
- Can be defined for multidimensional data

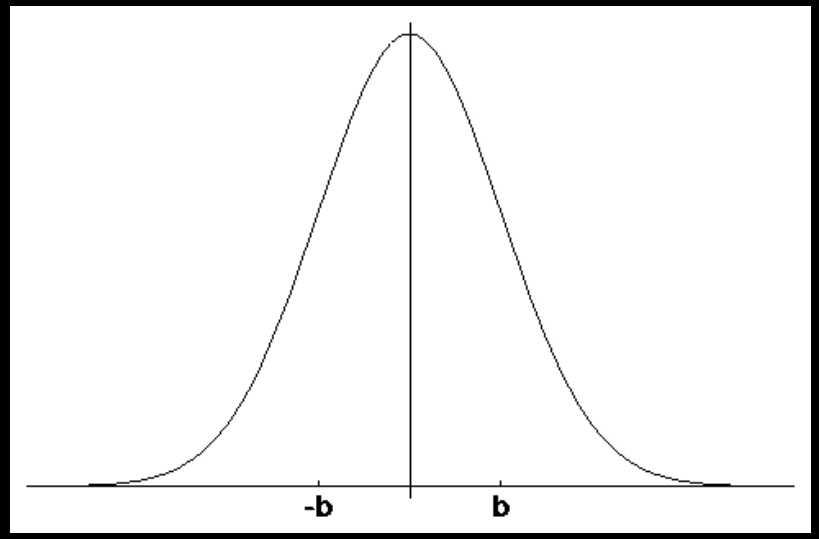


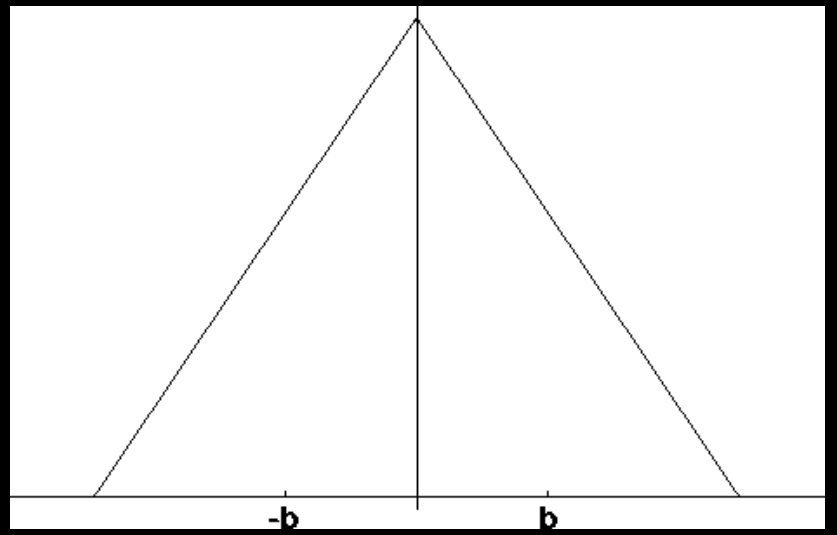
1D Gaussian Probability Density Function



$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Typical Distributions for Motion Models





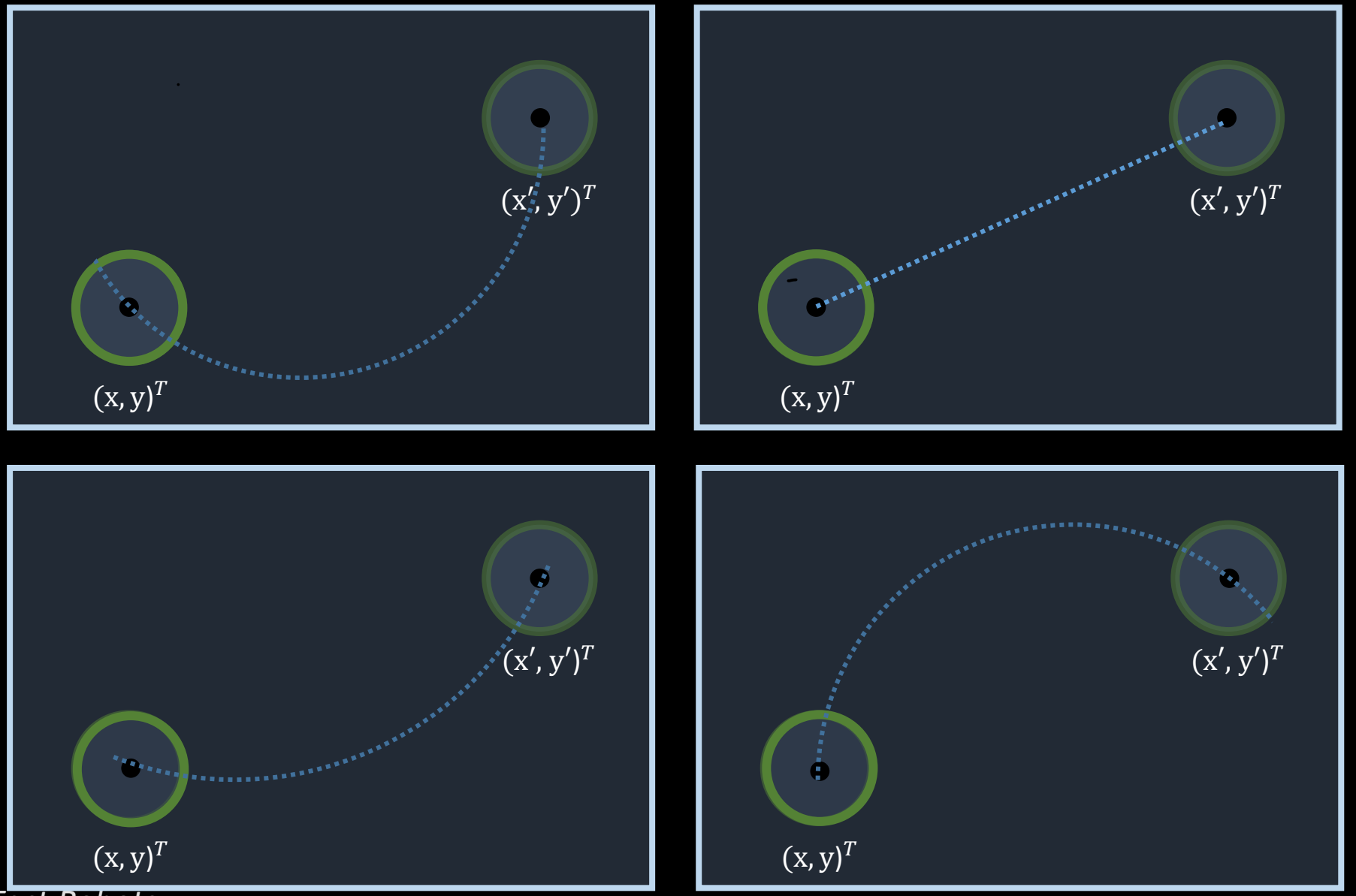
1. **Algorithm prob_normal_distribution**(a, b^2) :
2. **return** $\frac{1}{\sqrt{2\pi b^2}} \, exp\left(\frac{a^2}{2b^2}\right)$

1. **Algorithm prob_triangular_distribution**(a, b^2) :
2. **return** $max\left(0, \frac{1}{\sqrt{6}\, b} - \frac{|a|}{6\, b^2}\right)$

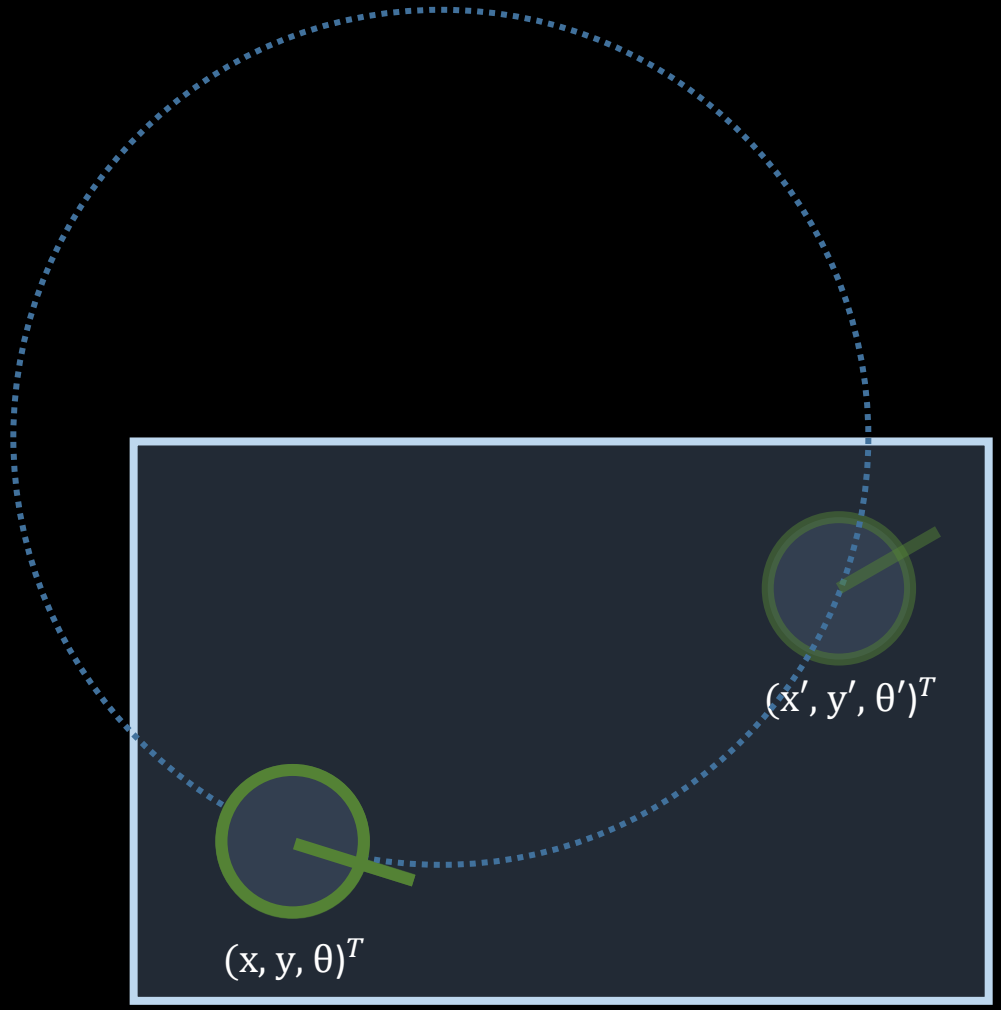
Velocity Model

- translation velocity v
- rotational velocity ω
- $u_t = (v, \omega)$

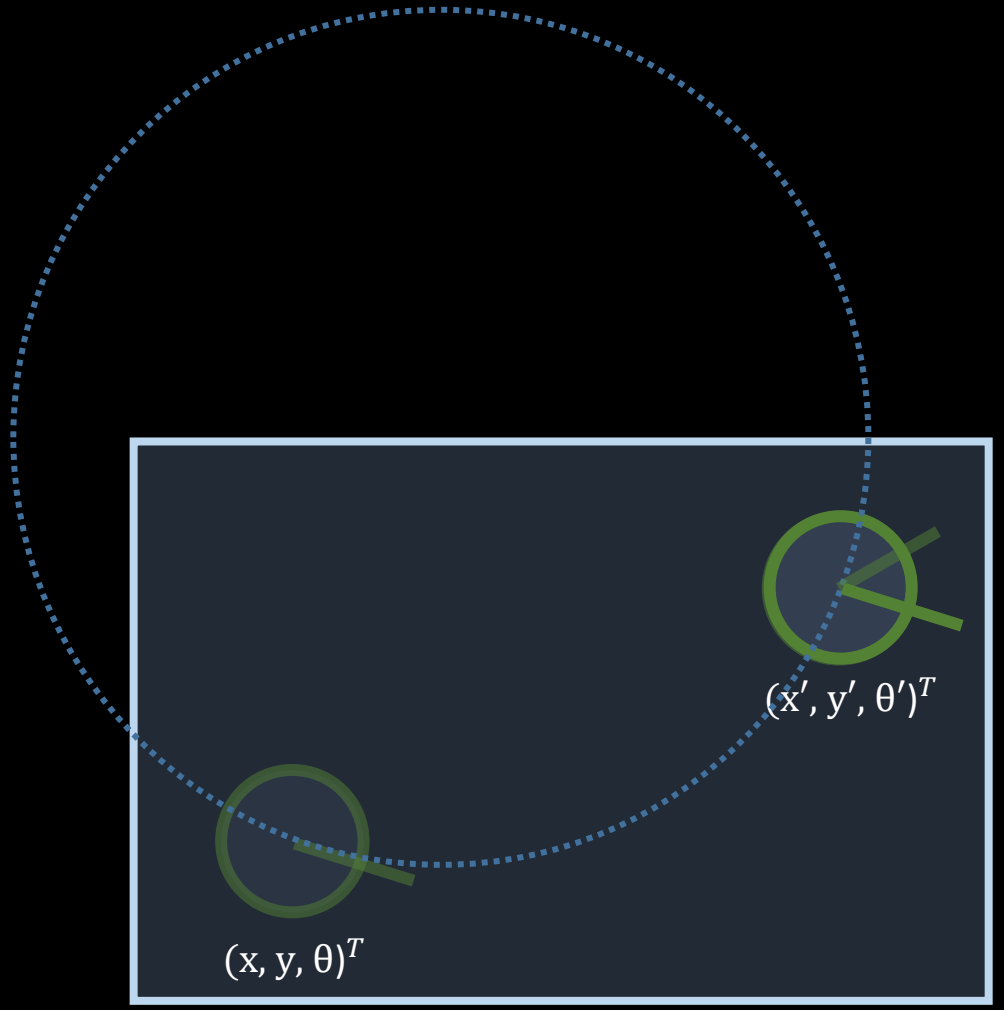
Velocity Model Parameters



Velocity Model Parameters



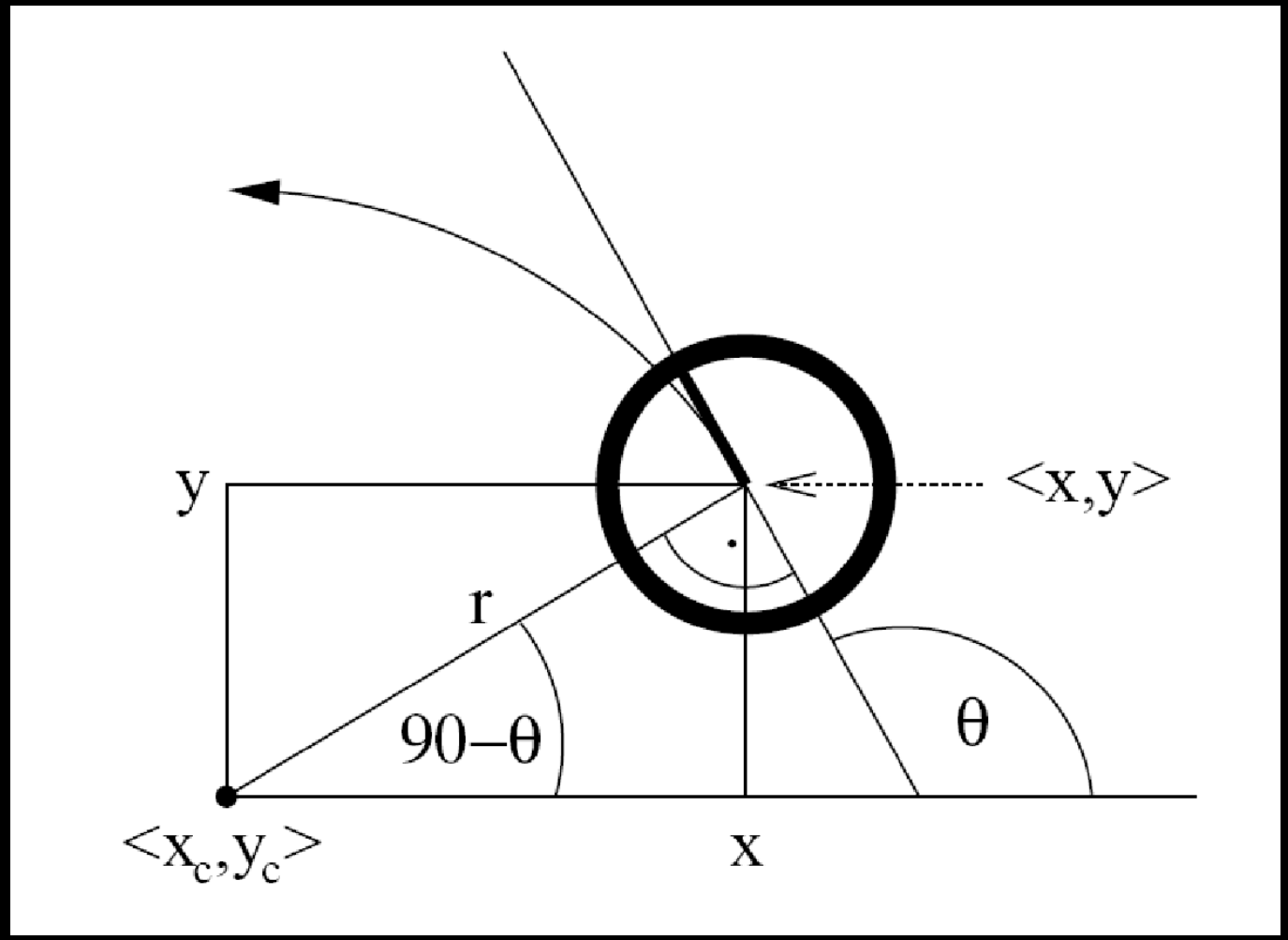
Velocity Model Parameters



(Rotation γ when at new pose)

Velocity Model

- Control data $u_t = (v_t,\ \omega_t)$
- Exact motion: $x_t = (x', y', \theta')^T$
  - $x_{t-1} = (x, y, \theta)^T$
  - $u_t = (v_t,\ \omega_t)^T$
- *(Under the assumption that both velocity components are kept fixed over the time interval)*



Velocity Model

Algorithm motion_model_velocity(x_t, u_t, x_{t-1})**:**

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

$$x^* = \frac{x + x'}{2} + \mu(y - y')$$

$$y^* = \frac{y + y'}{2} + \mu(x' - x)$$

$$r^* = \sqrt{(x - x^*)^2 + (y - y^*)^2}$$

$$\Delta\theta = \text{atan2}(y' - y^*, x' - x^*) - \text{atan2}(y - y^*, x - x^*)$$

$$\hat{v} = \frac{\Delta\theta}{\Delta t} r^*$$

$$\hat{\omega} = \frac{\Delta\theta}{\Delta t}$$

$$\hat{\gamma} = \frac{\theta' - \theta}{\Delta t} - \hat{\omega}$$

} Ideal control values

Calculate the error-free control between the states x_{t-1} and x_t

- ***How to add probability?***
  - $f(v_t | \hat{v}, \sigma_v^2)$
  - $f(\omega_t | \hat{\omega}, \sigma_\omega^2)$
  - $f(\gamma_t | \hat{\gamma}, \sigma_\gamma^2)$

Probabilistic Velocity Model

Algorithm motion_model_velocity(x_t, u_t, x_{t-1})**:**

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

$$x^* = \frac{x+x'}{2} + \mu(y-y')$$

$$y^* = \frac{y+y'}{2} + \mu(x'-x)$$

$$r^* = \sqrt{(x-x^*)^2 + (y-y^*)^2}$$

$$\Delta\theta = \text{atan2}(y'-y^*, x'-x^*) - \text{atan2}(y-y^*, x-x^*)$$

$$\hat{v} = \frac{\Delta\theta}{\Delta t} r^*$$

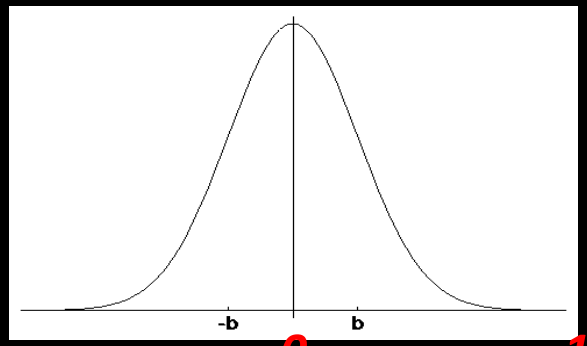
$$\hat{\omega} = \frac{\Delta\theta}{\Delta t}$$

$$\hat{\gamma} = \frac{\theta'-\theta}{\Delta t} - \hat{\omega}$$

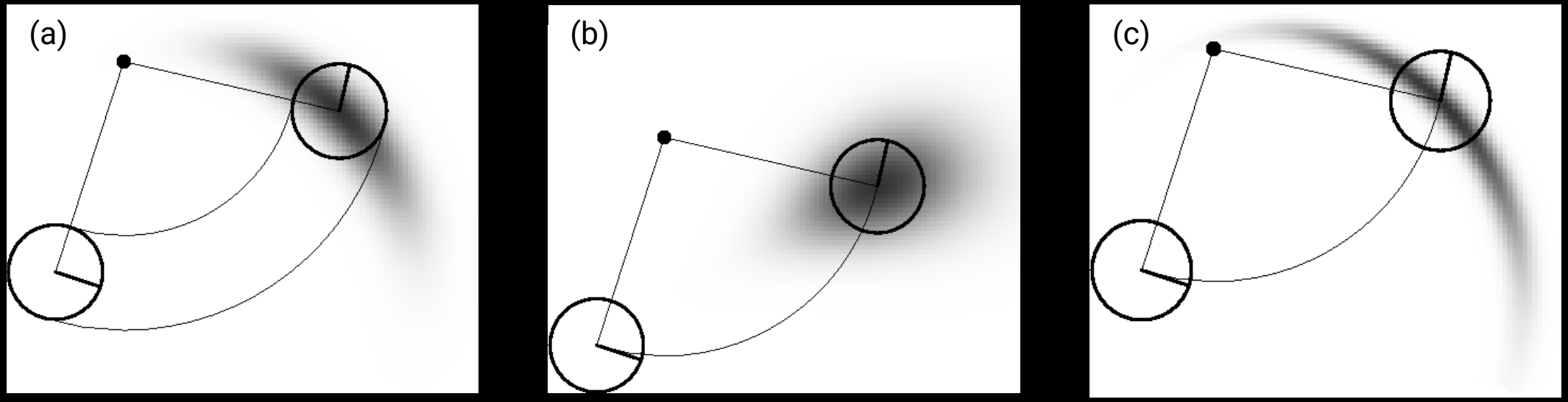
return $\textbf{prob}(v-\hat{v}, \alpha_1|v| + \alpha_2|\omega|) \cdot \textbf{prob}(\omega-\hat{\omega}, \alpha_3|v| + \alpha_4|\omega|) \cdot \textbf{prob}(\hat{\gamma}, \alpha_5|v| + \alpha_6|\omega|)$

Calculate the error-free control between the states x_{t-1} and x_t

- ***How to add probability?***
  - $f(v_t | \hat{v}, \sigma_v^2)$
    - $f(v_t - \hat{v} | 0, \sigma_v^2)$
  - $f(\omega_t | \hat{\omega}, \sigma_\omega^2)$
  - $f(\gamma_t | \hat{\gamma}, \sigma_\gamma^2)$



Velocity Motion Model

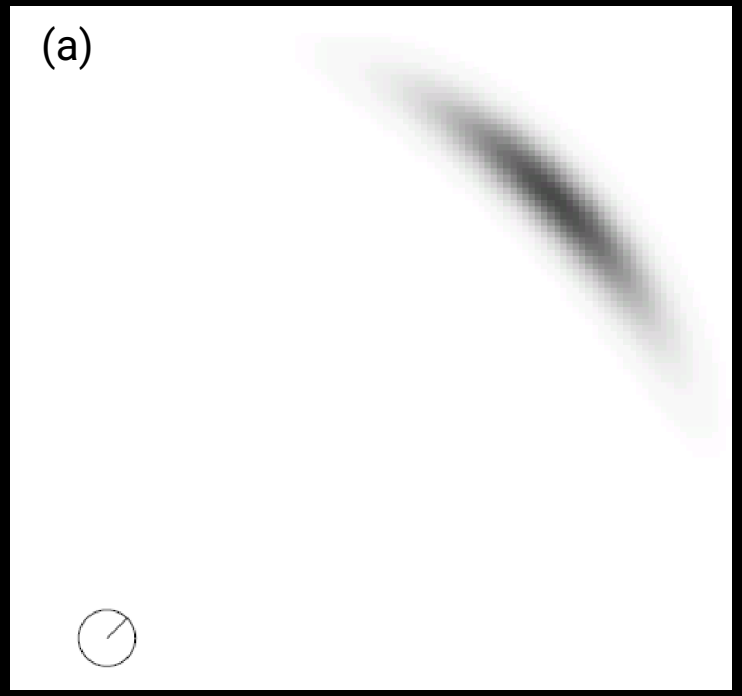


(darker regions are more probable)

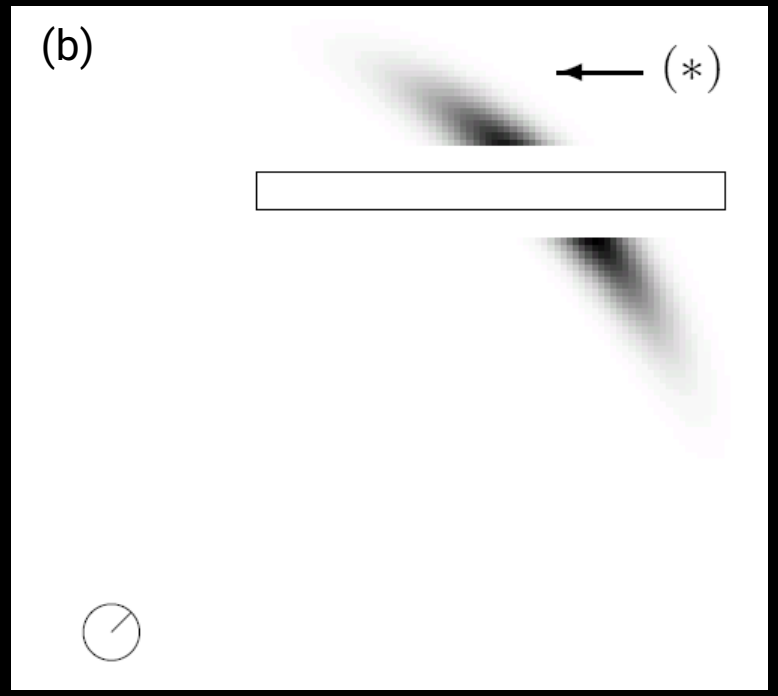
The velocity motion model for different noise parameters settings for the same control $u_t = (v_t,\ \omega_t)^T$ projected in the x-y space

a) More rotational than translational noise
b) Larger transitional noise
c) Large angular and translational noise parameters

Velocity Model with a Map



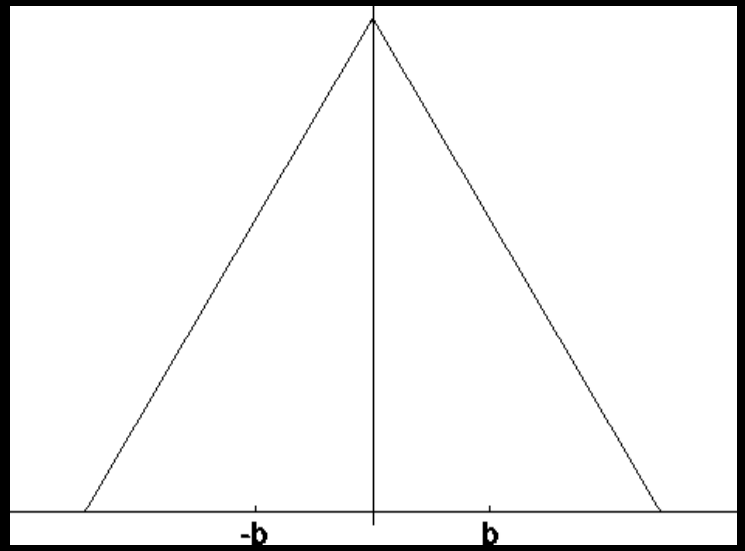
$p(x_t \mid u_t, x_{t-1})$



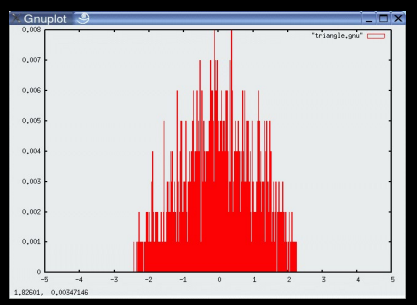
$p(x_t \mid u_t, x_{t-1}, map)$

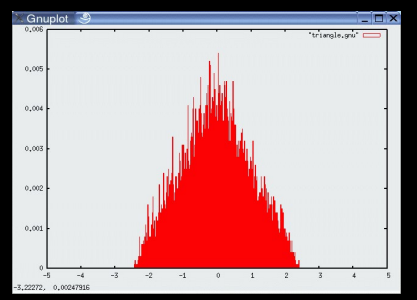
Sampling Algorithm

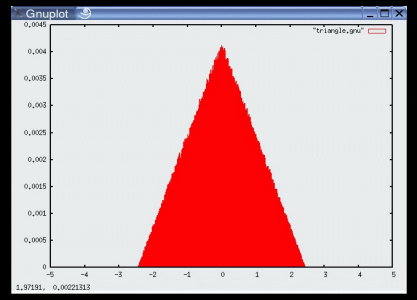
- A sampling algorithm outputs samples $y_1, y_2 \ldots$ from a given distribution P
- Often used to approximate distributions
- A triangular distribution is given by



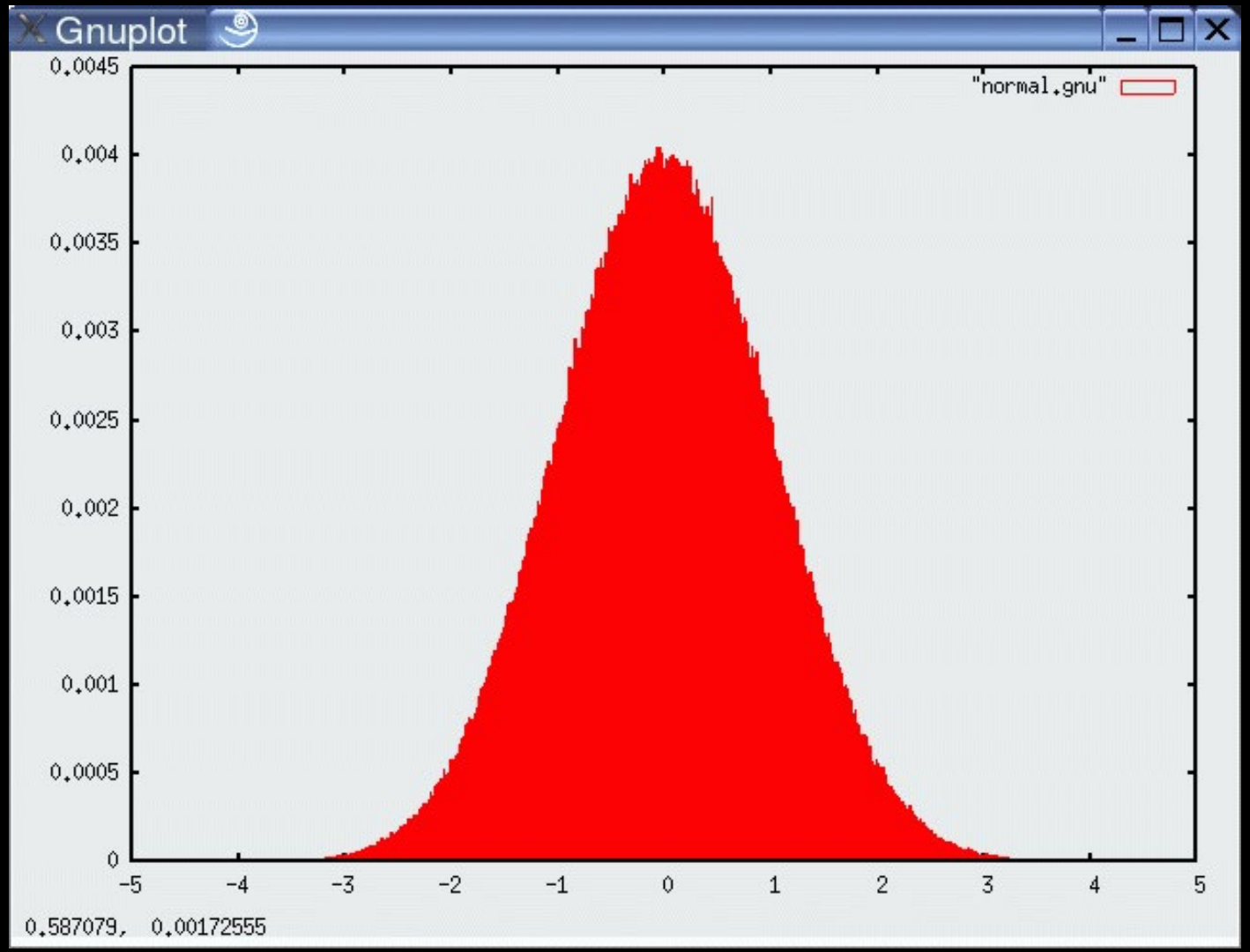
$$\varepsilon_{\sigma^2}(x) = \begin{cases} 0 \text{ if } |x| > \sqrt{6\sigma^2} \\ \dfrac{\sqrt{6\sigma^2} - |x|}{6\sigma^2} \end{cases}$$







Normally Distributed Samples



Sampling from Velocity Model

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

$\hat{v} = v + \textbf{sample}(\alpha_1|v| + \alpha_2|\omega|)$

$\hat{\omega} = \omega + \textbf{sample}(\alpha_3|v| + \alpha_4|\omega|)$

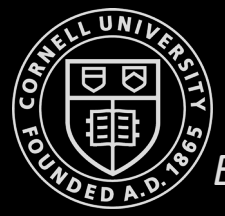
$\hat{\gamma} = \textbf{sample}(\alpha_5|v| + \alpha_6|\omega|)$

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

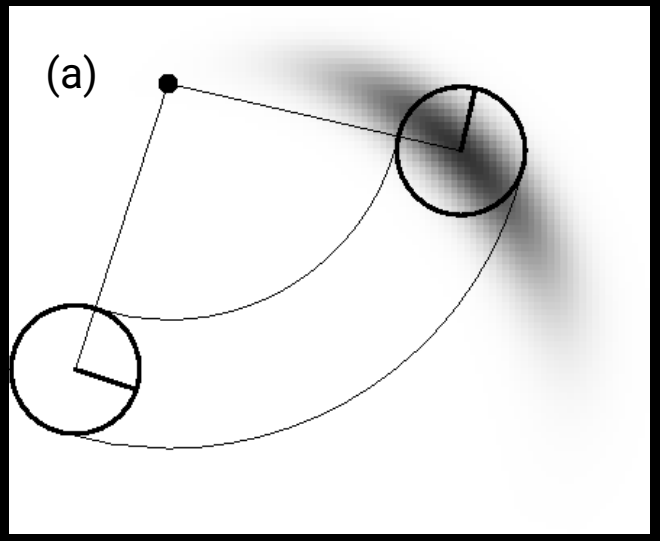
$y' = y + \frac{\hat{v}}{\hat{\omega}} \cos\theta - \frac{\hat{v}}{\hat{\omega}} \, cos(\theta + \hat{\omega}\Delta t)$

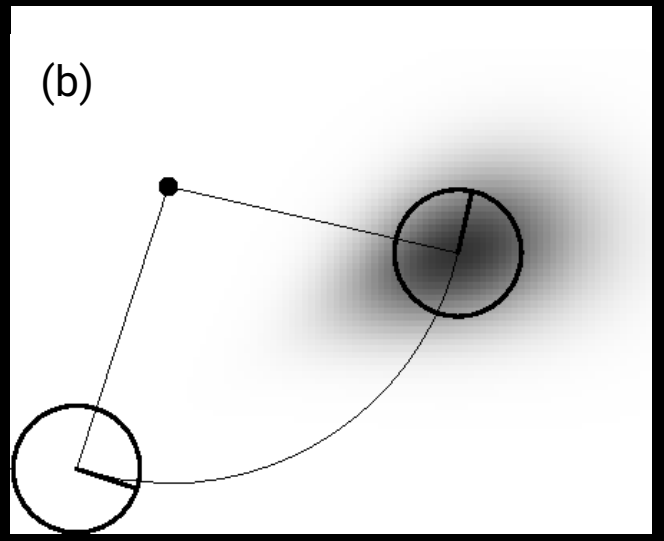
$\theta' = \theta + \hat{\omega}\Delta t + \hat{\gamma}\Delta t$

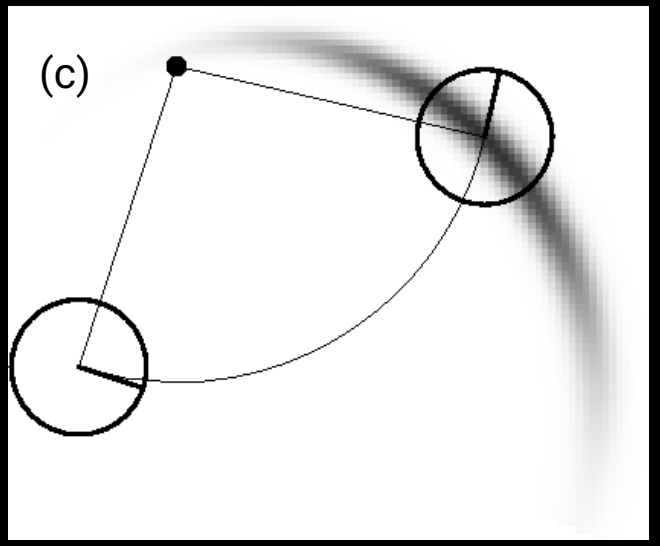
return $x_t = (x', y', \theta')^T$

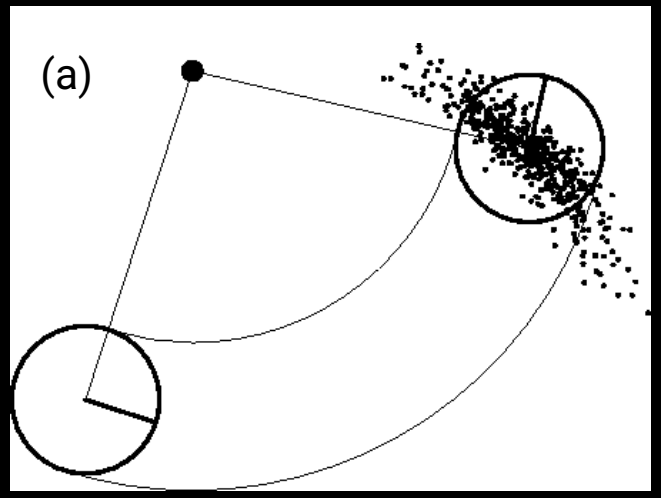


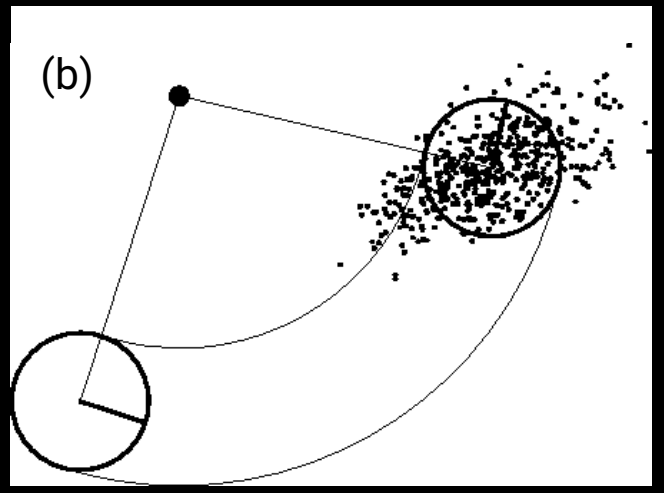
Sampling from Velocity Model

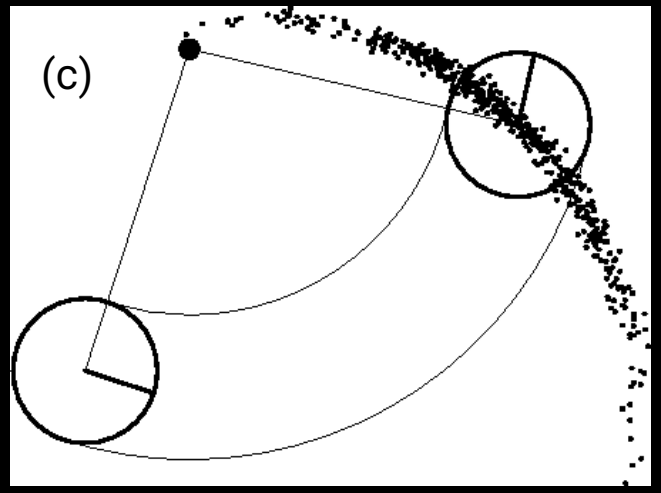








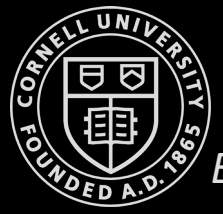




Odometry Model

Odometry Model

- Odometry is the use of data from motion sensors to estimate change in position over time
  - Odometry obtained by integrating wheel encoder information
- However, odometry is available **after the robot has moved**
  - It cannot be used for planning, since they need to predict the effects of motion
  - Can still be used for filter algorithms such as localization and mapping algorithms
- Odometry models are usually applied for estimation while velocity models are used for probabilistic motion planning



Odometry Model

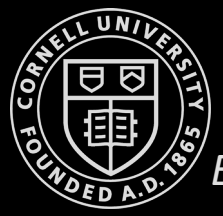
- From $(t-1, t]$, the robot advances from x_t to x_{t-1}
- The odometry reports back to us a related advance from

$$\overline{x_{t-1}} = (\bar{x}, \bar{y}, \bar{\theta})^T \quad \text{to} \quad \bar{x_t} = (\bar{x'}, \bar{y'}, \bar{\theta'})^T$$

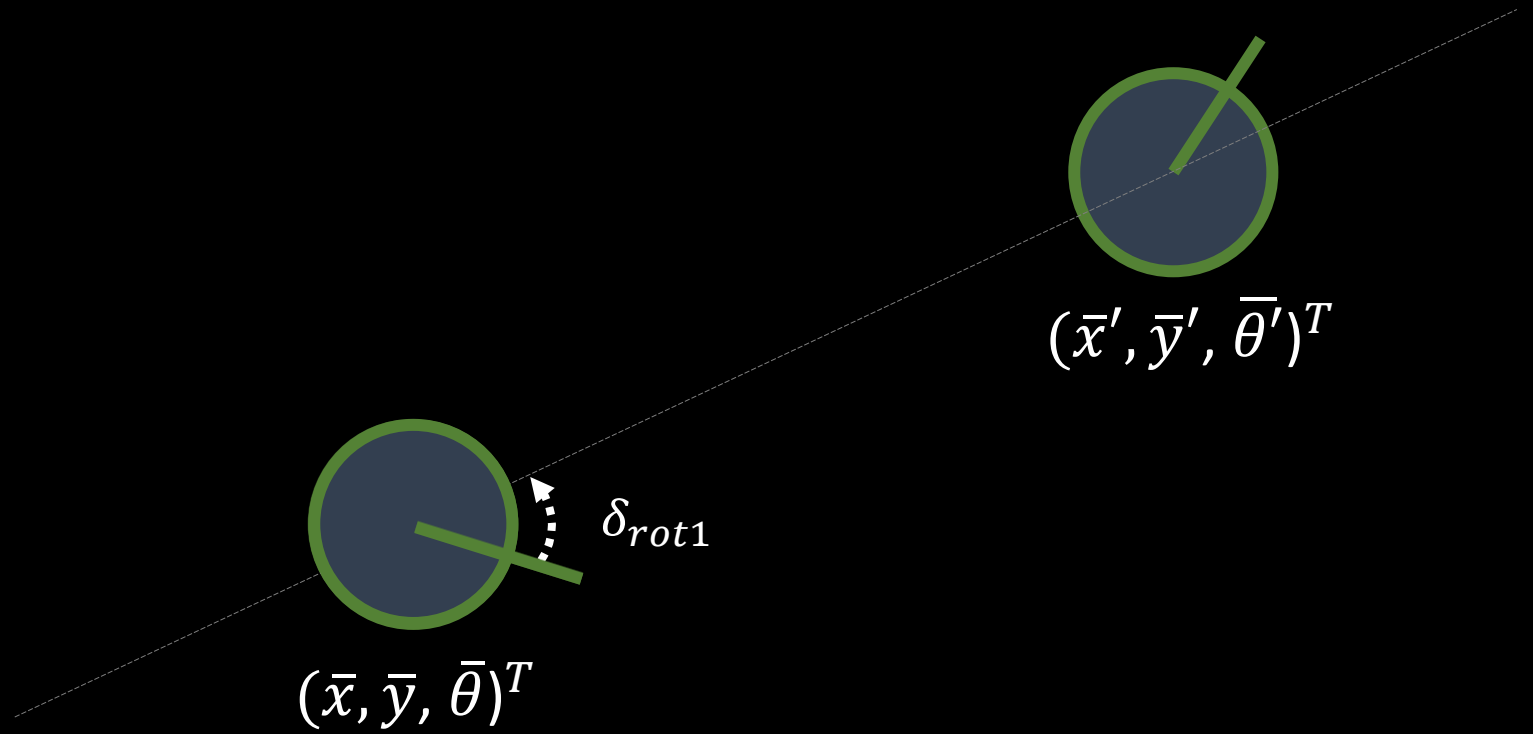
(bar indicates they are odometry measurements in the robot's local frame)

- **Key idea:** In state estimation, the relative difference between $\overline{x_{t-1}}$ and $\bar{x_t}$ is a good estimator for the difference of the true poses x_{t-1} and x_t
- Motion information is given by:

$$u_t = (\overline{x_{t-1}}, \bar{x_t})^T$$



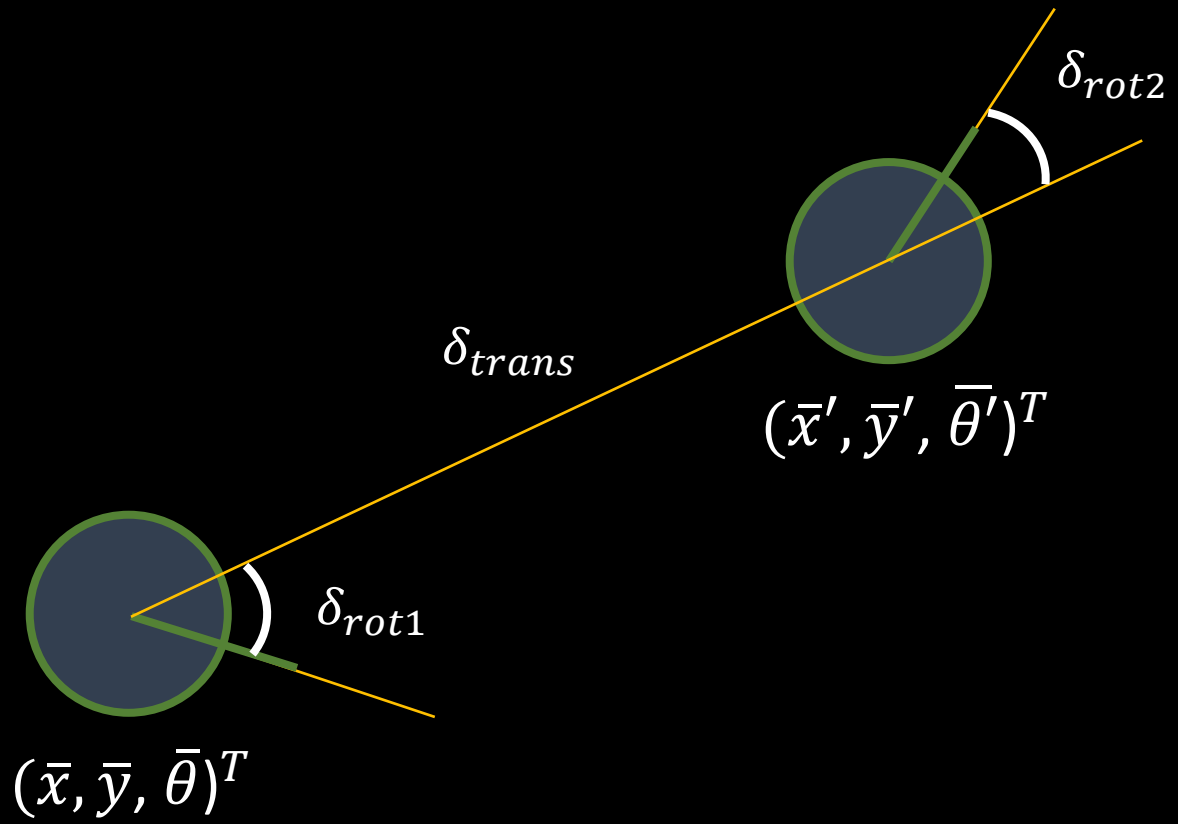
Odometry Model Parameters



Odometry Model Parameters

- Relative odometry motion is transformed into a sequence of three steps
  - Initial rotation δ_{rot1}
  - Translation δ_{trans}
  - Final Rotation δ_{rot2}
- These three parameters are sufficient to reconstruct the relative motion between two robot states

$$u_t = (\delta_{rot1}, \delta_{trans}, \delta_{rot2})^T$$

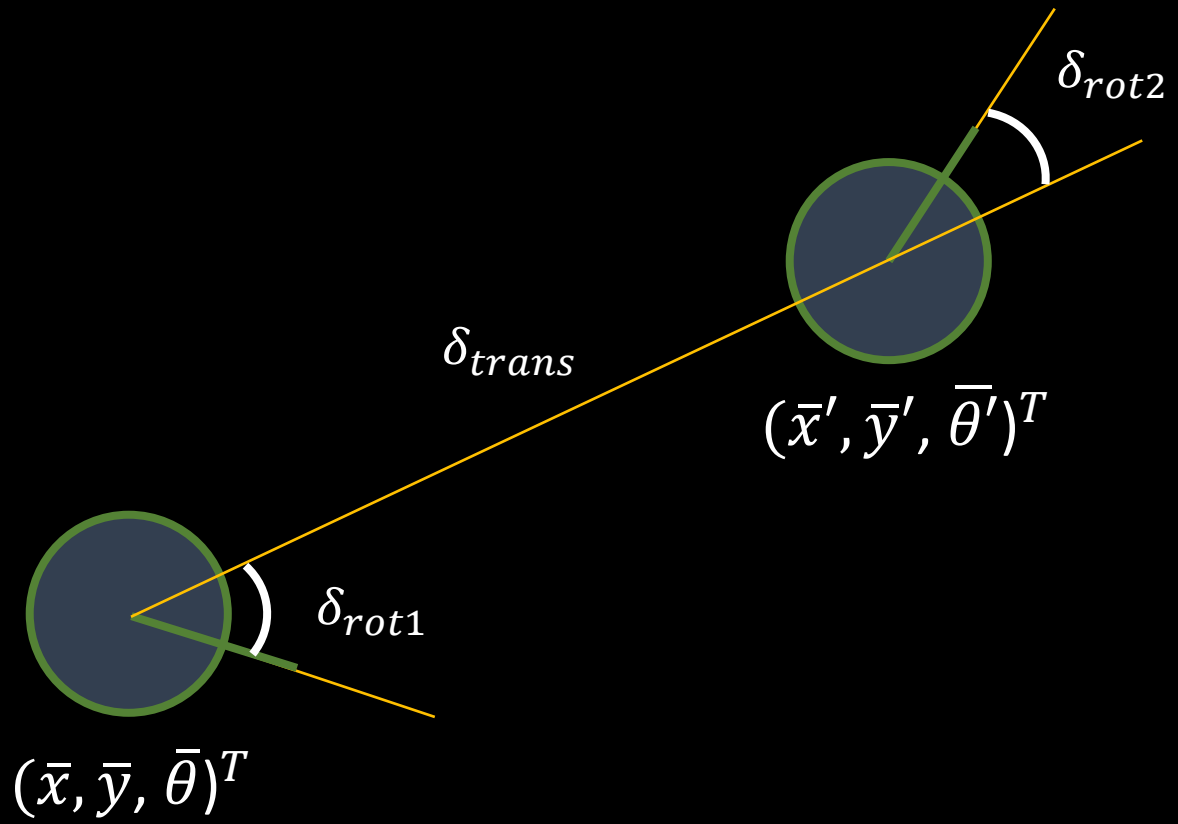


Odometry Model Parameters

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

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

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



1. **Algorithm motion_model_odometry**(x_t, u_t, x_{t-1}) :

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

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

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

5. $\hat{\delta}_{rot1} = \text{atan2}(y' - y, x' - x) - \theta$

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

7. $\hat{\delta}_{rot2} = \theta' - \theta - \hat{\delta}_{rot1}$

8. $p_1 = \textbf{prob}(\delta_{rot1} - \hat{\delta}_{rot1}, \alpha_1 \hat{\delta}^2_{rot1} + \alpha_2 \hat{\delta}^2_{trans})$

9. $p_2 = \textbf{prob}(\delta_{trans} - \hat{\delta}_{trans}, \alpha_3 \hat{\delta}^2_{trans} + \alpha_4 \hat{\delta}^2_{rot1} + \alpha_4 \hat{\delta}^2_{rot2})$

10. $p_3 = \textbf{prob}(\delta_{rot2} - \hat{\delta}_{rot2}, \alpha_1 \hat{\delta}^2_{rot2} + \alpha_2 \hat{\delta}^2_{trans})$

11. return $p_1.p_2.p_3$

Calculate the relative motion parameters from odometry readings

Calculate the relative motion parameters for the given states x_{t-1} and x_t

Algorithm for computing $p(x_t \mid u_t, x_{t-1})$ based on odometry information. Here the control $u_t = (\overline{x_{t-1}}, \overline{x_t})^T$ with $\overline{x_{t-1}} = (\bar{x}, \bar{y}, \bar{\theta})^T$ and $\overline{x_t} = (\bar{x}', \bar{y}', \bar{\theta}')^T$

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

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

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

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

Calculate the relative motion parameters from odometry readings

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

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

7. $\hat{\delta}_{rot2} = \delta_{rot1} - \mathbf{sample}(\alpha_1 \hat{\delta}^2_{rot2} + \alpha_2 \hat{\delta}^2_{trans})$

Add noise to calculated motion parameters

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

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

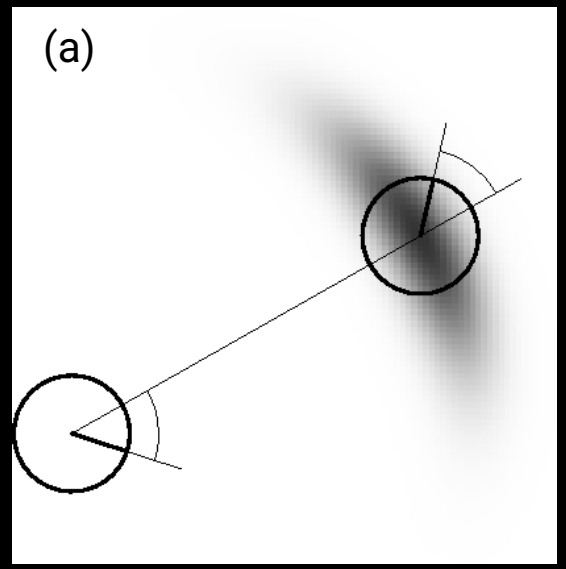
10. $\theta' = \theta + \hat{\delta}_{rot1} + \hat{\delta}_{rot2}$

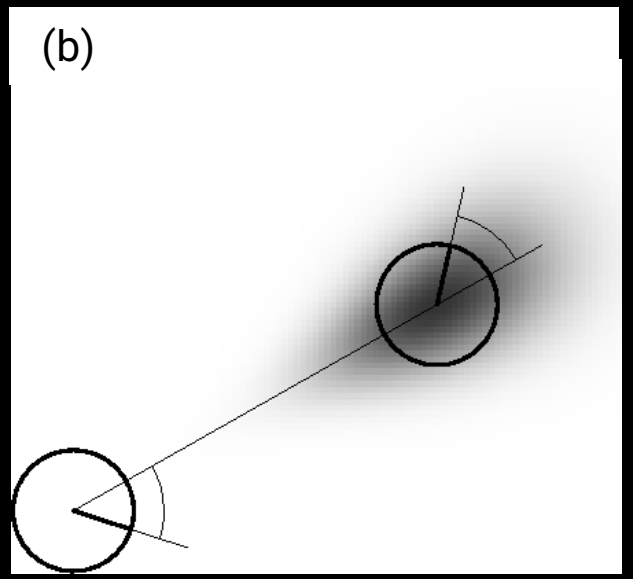
Calculate the sample state

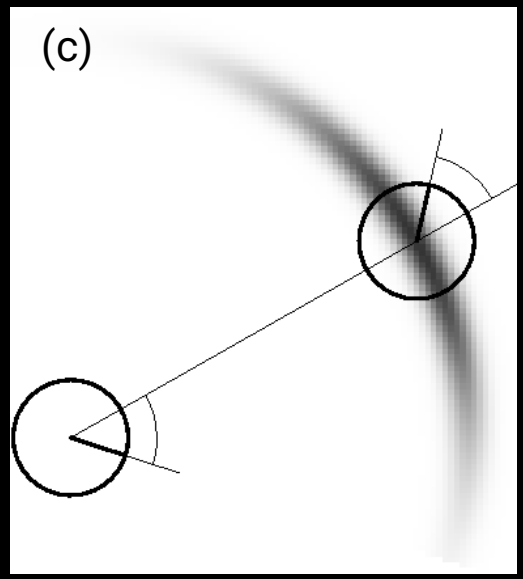
11. **return** $x_t = (x', y', \theta')^T$

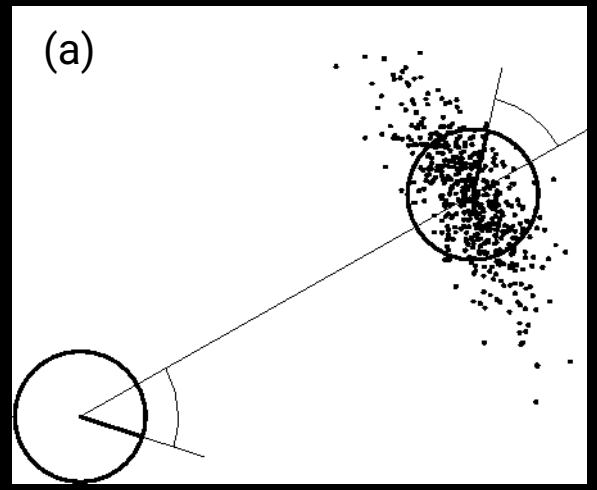
Algorithm for sampling from $p(x_t \mid u_t, x_{t-1})$ based on odometry information

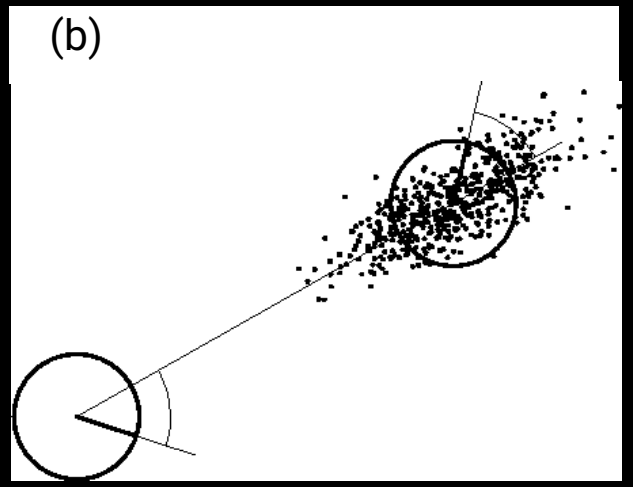
Sampling from Velocity Model

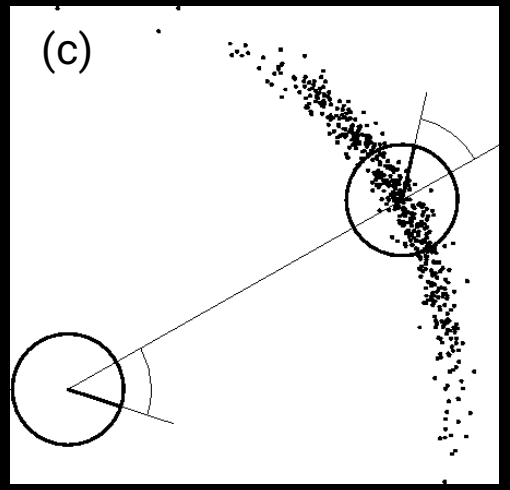






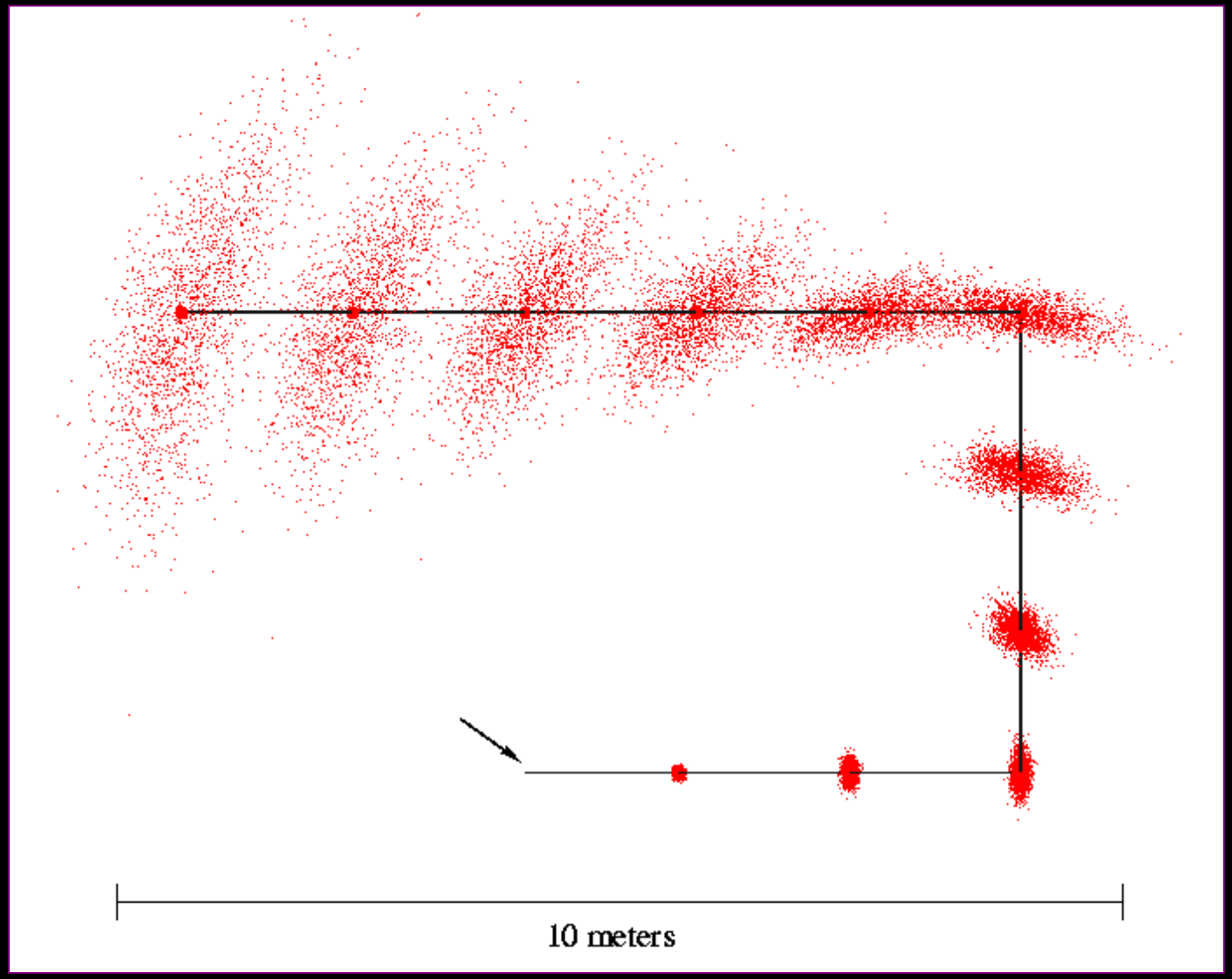






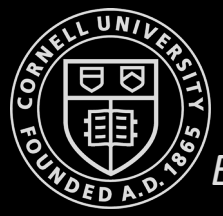
Sampling from the odometry model, using the same error parameters as in the previous slides with 500 samples in each.

Repeated Sampling from Our Motion Model



Summary

- We discussed motion models for odometry-based and velocity-based systems
- We discussed ways to calculate the posterior probability $p(x_t \mid u_t, x_{t-1})$
- We also described how to sample from $p(x_t \mid u_t, x_{t-1})$
- In practice, the parameters of the models have to be learned
- We also briefly discussed an extended motion model that takes the map into account



Reference

1. Thrun, Sebastian, Wolfram Burgard, and Dieter Fox. Probabilistic robotics. MIT press, 2005.
2. http://ais.informatik.uni-freiburg.de/teaching/ss11/robotics/slides/06-motion-models.pdf