Motion models

Fast Robots, ECE4160/5160, MAE 4190/5190

E. Farrell Helbling, 3/26/26

Class Action Items

- Happy Spring Break! Please enjoy your break and get some much needed rest.
- Lab 8 is due after break. For those on campus, we put out a sticky mat on the fifth floor of Upson, check Ed for a picture of the location!
- Today motion models, measurement models after spring break.

Bayes Filter

Markov Assumption

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

- **If** we can model our robot as a Markov process…
  - We can recursively estimate x_t
  - 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
    - $p(z_t \mid x_{0:t}, z_{1:t-1}, u_{1:t}) = p(z_t \mid x_t)$

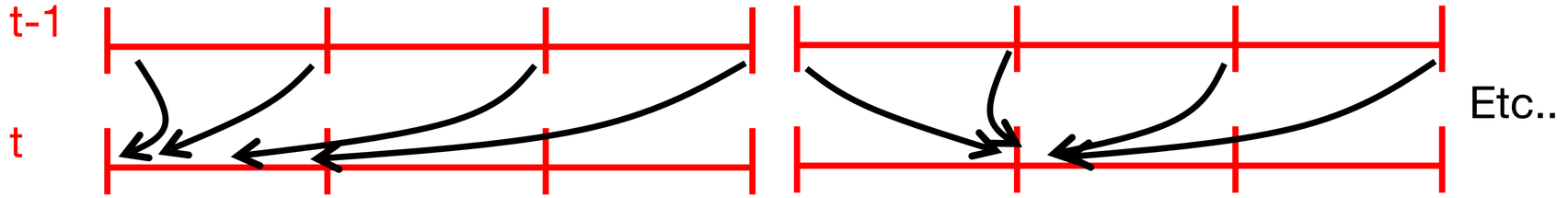
Andrey Markov (1856–1922) was a Russian mathematician best known for his work on stochastic processes

Bayes Filter

- A recursive algorithm that calculates the belief distribution from measurements and control data

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

Bayes Filter



Correct estimate with sensor measurement..

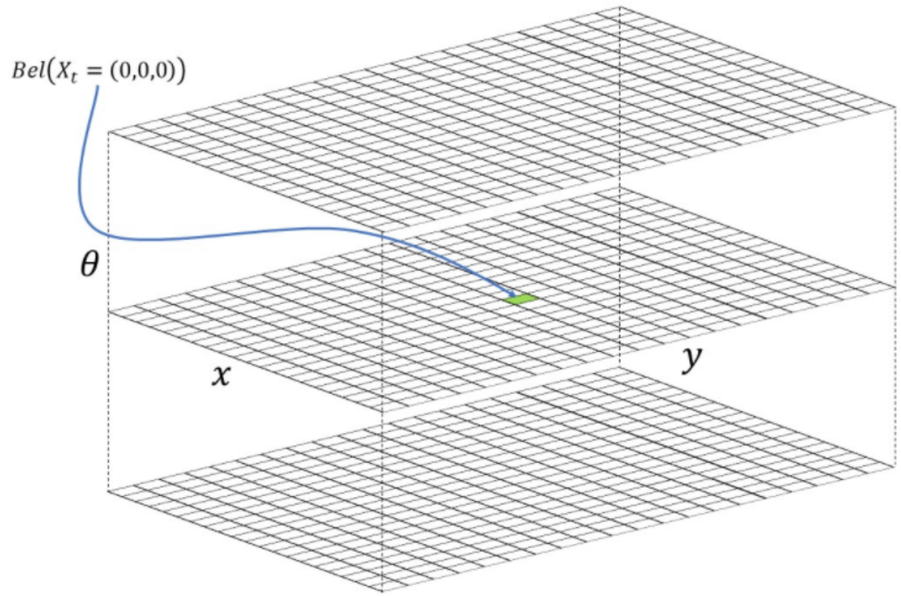
1. **Algorithm Bayes_Filter** $(bel(x_{t-1}),\ u_t,\ z_t)$:
2. **for** all x_t do
3. $\overline{bel}(x_t) = \Sigma_{x_{t-1}}\ p(x_t \mid u_t,\ x_{t-1})\ bel(x_{t-1})$ **(Prediction step)**
4. $bel(x_t) = \eta\ p(z_t \mid x_t)\ \overline{bel}(x_t)$ **(Update/measurement step)**
5. end **for**
6. **return**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

Transition probability/ action model

Measurement probability/ sensor model

Bayes Filter

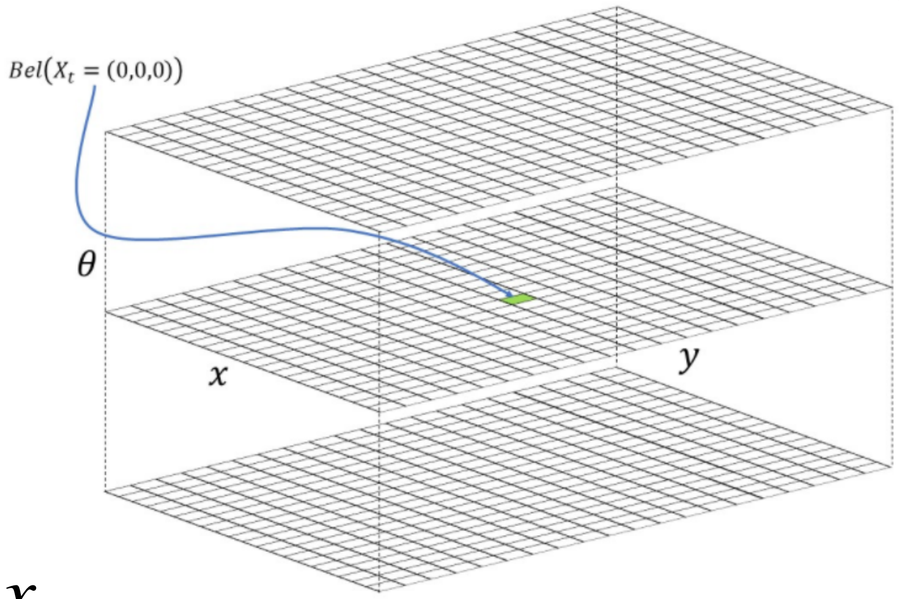
This is a lot of computation!



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

Bayes Filter

Markov Assumption Violations



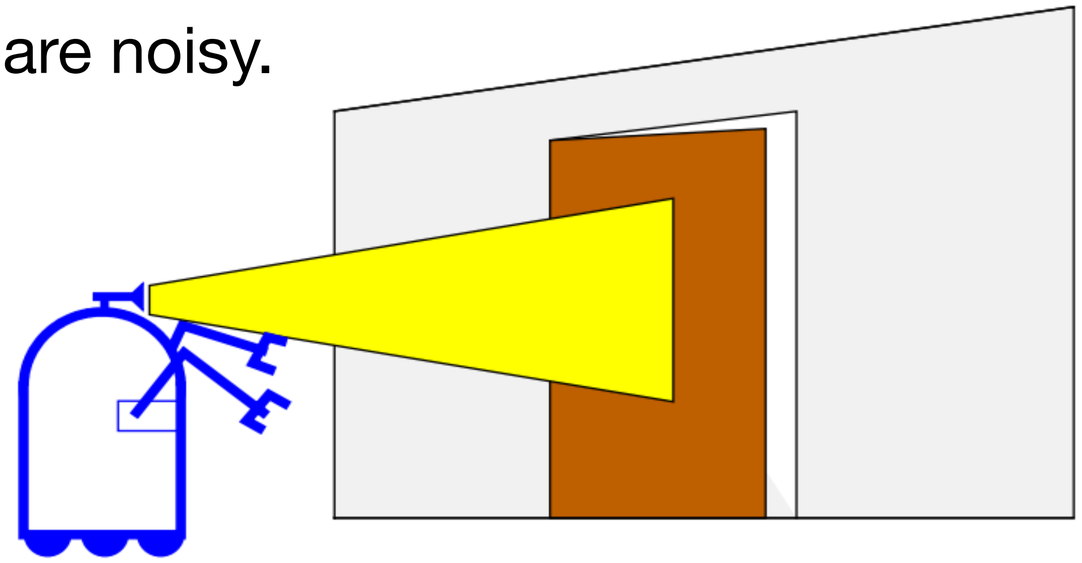
This is a lot of computation!

- Typical violations include:
  - Environmental dynamics not included in x_t
  - Inaccuracies in the probabilist models $\mathrm{p}(x_t \mid x_{t-1}, u_t)$, and $\mathrm{p}(z_t \mid x_t)$
  - 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 Example

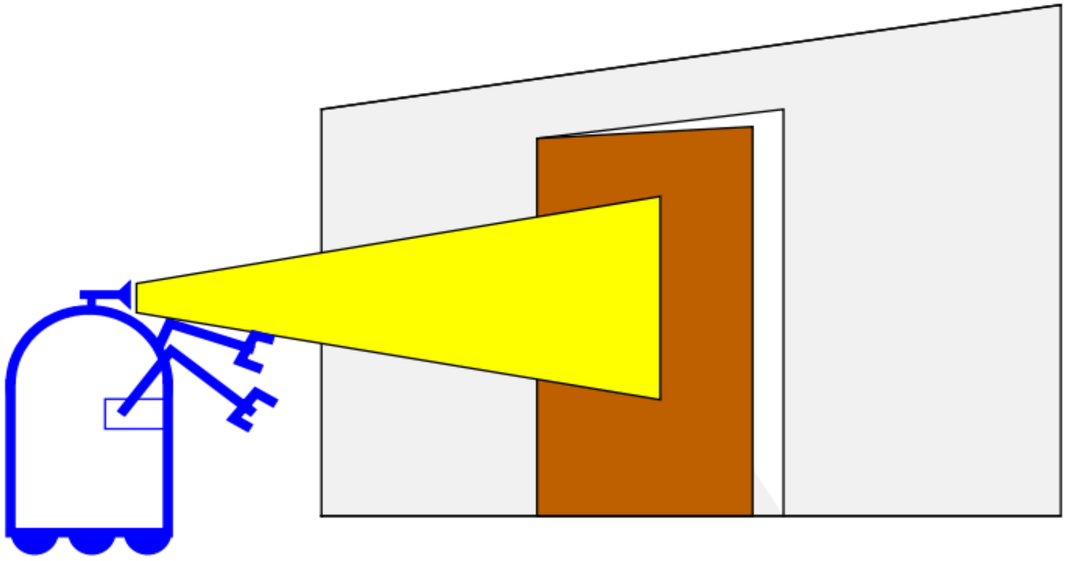
Bayes Filter — Example

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



Bayes Filter — Example

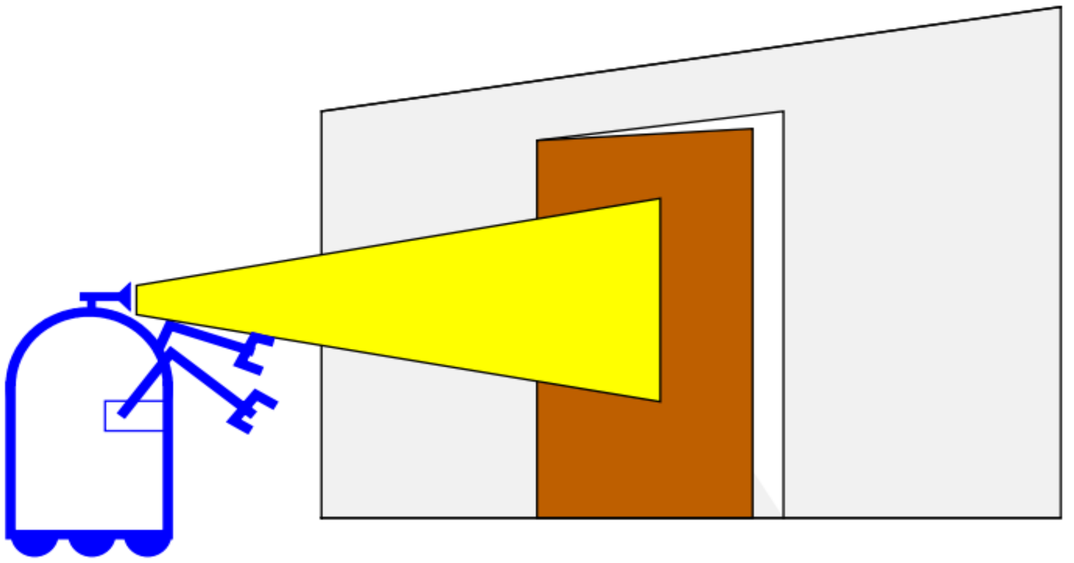
- The probability that the robot can sense an **open** door is 0.6
- The probability that the robot can sense an **closed** door is 0.8
- After a **push** action, probability that a door is **open** if it was previously open is 1
- After a **push** action, probability that a door is **open** if it was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



Bayes Filter — Example

Measurement model

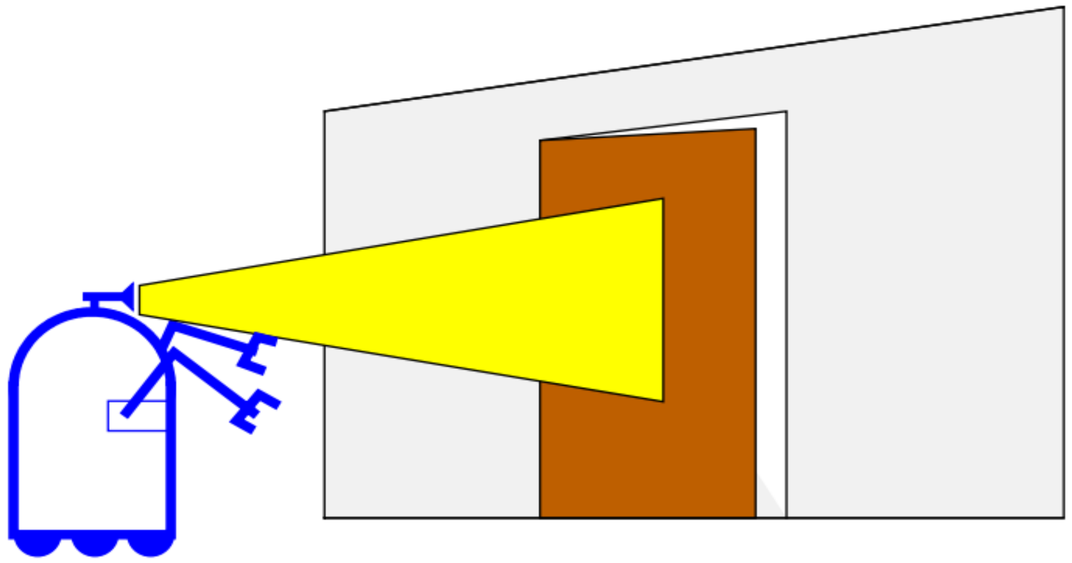
- The probability that the robot can sense an **open** door is 0.6
- The probability that the robot can sense a **closed** door is 0.8
- Measurement model: $\mathrm{p}(z_t \mid x_t)$
  - $p(Z_t = \text{open} \mid X_t = \text{is\_open})$
  - $p(Z_t = \text{closed} \mid X_t = \text{is\_open})$
  - $p(Z_t = \text{closed} \mid X_t = \text{is\_closed})$
  - $p(Z_t = \text{open} \mid X_t = \text{is\_closed})$



Bayes Filter — Example

Action model

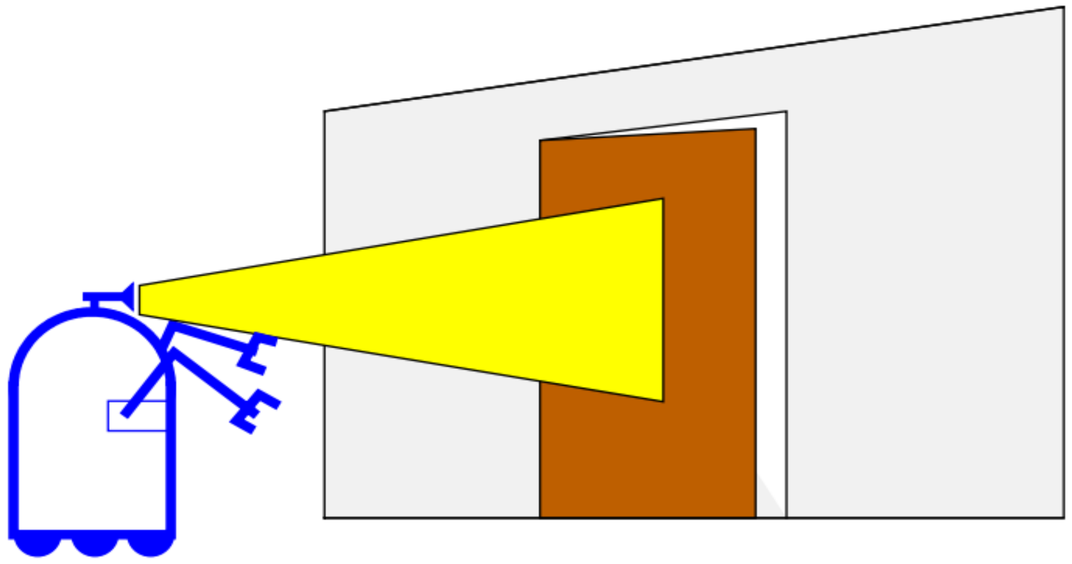
- After a **push** action, probability that a door is **open** if it was previously open is 1
- After a **push** action, probability that a door is **open** if it was previously closed is 0.8
- If the robot **does nothing**, the door continues to be in the previous state.
- Action model: $\mathrm{p}(x_t \mid u_t, x_{t-1})$
  - $p(X_t = \text{is\_open} \mid U_t = \text{push}, X_{t-1} = \text{is\_open})$
  - $p(X_t = \text{is\_closed} \mid U_t = \text{push}, X_{t-1} = \text{is\_open})$
  - $p(X_t = \text{is\_open} \mid U_t = \text{push}, X_{t-1} = \text{is\_closed})$
  - $p(X_t = \text{is\_closed} \mid U_t = \text{push}, X_{t-1} = \text{is\_closed})$



Bayes Filter — Example

Action model

- 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.
- Action model: $\mathrm{p}(x_t \mid x_{t-1}, u_t)$
  - $p(X_t = \text{is\_open} \mid U_t = \text{NOP}, X_{t-1} = \text{is\_open})$
  - $p(X_t = \text{is\_closed} \mid U_t = \text{NOP}, X_{t-1} = \text{is\_open})$
  - $p(X_t = \text{is\_open} \mid U_t = \text{NOP}, X_{t-1} = \text{is\_closed})$
  - $p(X_t = \text{is\_closed} \mid U_t = \text{NOP}, X_{t-1} = \text{is\_closed})$



Bayes Filter — Example

Problem Setup

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

Bayes Filter — Example

Prediction Step - incorporate action

$$bel(X_0 = \text{is\_open}) = bel(X_0 = \text{is\_closed}) = 0.5 \qquad U_1 = \text{NOP} \qquad Z_1 = \text{open}$$

$$\overline{bel}(x_1) = \Sigma_{x_0}\ p(x_1 \mid u_1, x_0)\ bel(x_0)$$

$$\overline{bel}(x_1) =$$

Let's suppose $X_1 = \text{is\_closed}$

$$\overline{bel}(X_1 = \text{is\_closed}) = p(X_1 = \text{is\_closed} \mid U_1 = \text{NOP}, X_0 = \text{is\_open})\ bel(X_0 = \text{is\_open})$$
$$+ p(X_1 = \text{is\_closed} \mid U_1 = \text{NOP}, X_0 = \text{is\_closed})\ bel(X_0 = \text{is\_closed})$$

Bayes Filter — Example

Update Step - incorporate measurement

$$\overline{bel}(X_1 = \text{is\_open}) = \overline{bel}(X_1 = \text{is\_closed}) = 0.5 \qquad U_1 = \text{NOP} \qquad Z_1 = \text{open}$$

$$bel(x_1) = \eta \; p(Z_1 = \text{open} \,|\, x_1) \; \overline{bel}(x_1)$$

For two possible cases, $X_1 = \text{is\_open}$ and $X_1 = \text{is\_closed}$, we compute:

$$\begin{aligned} bel(X_1 = \text{is\_open}) &= \eta \; p(Z_1 = \text{open} \,|\, X_1 = \text{is\_open}) \; \overline{bel}(X_1 = \text{is\_open}) \\ &= \eta \times 0.6 \times 0.5 = \eta \; 0.3 \\ bel(X_1 = \text{is\_closed}) &= \eta \; p(Z_1 = \text{open} \,|\, X_1 = \text{is\_closed}) \; \overline{bel}(X_1 = \text{is\_closed}) \\ &= \eta \times 0.2 \times 0.5 = \eta \; 0.1 \end{aligned}$$

Normalizing constant, $\eta = (0.3 + 0.1)^{-1} = 2.5$:

$$\begin{aligned} bel(X_1 = \text{is\_open}) &= \eta \; 0.3 = 0.75 \\ bel(X_1 = \text{is\_closed}) &= \eta \; 0.1 = 0.25 \end{aligned}$$

Better than initial belief at t=0!

Bayes Filter — Example

Time step 2

$$bel(X_1 = \text{is\_open}) = 0.75$$
$$bel(X_1 = \text{is\_closed}) = 0.25$$
$$U_2 = \text{push}$$
$$Z_2 = \text{open}$$

Prediction step:

$$\overline{bel}(X_2 = \text{is\_open}) = 1 \times 0.75 + 0.8 \times 0.25 = 0.95$$
$$\overline{bel}(X_2 = \text{is\_closed}) = 0 \times 0.75 + 0.2 \times 0.25 = 0.05$$

Measurement update:

$$bel(X_2 = \text{is\_open}) = \eta \times 0.6 \times 0.95 \approx 0.983$$
$$bel(X_2 = \text{is\_closed}) = \eta \times 0.2 \times 0.05 \approx 0.017$$

Way better than the initial belief at t=0!

Summary of Bayes Filter

- The robot performs a series of alternating actions/ measurements
- Given:
  - Sensor model: $p(z_t \mid x_t)$
  - Action model: $p(x_t \mid u_t, x_{t-1})$
  - Initial conditions: $p(x_0)$
- Compute:
  - State of dynamic system
  - Posterior of the state (belief): $bel(x_t) = p(x_t \mid u_1, z_1, \ldots, u_t, z_t)$

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

Summary of Bayes Filter

- **Prediction Step:**
  - Incorporate action, which **increases** uncertainty
  - Compute $\overline{bel}(x_t) = p(x_t \mid u_{1:t}, z_{1:t-1})$
  - Requires action model: $p(x_t \mid u_t, x_{t-1})$
- **Measurement/ update step:**
  - **Decreases** uncertainty
  - Compute $bel(x_t) = p(x_t \mid u_{1:t}, z_{1:t})$
  - Requires sensor model: $p(z_t \mid x_t)$

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

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

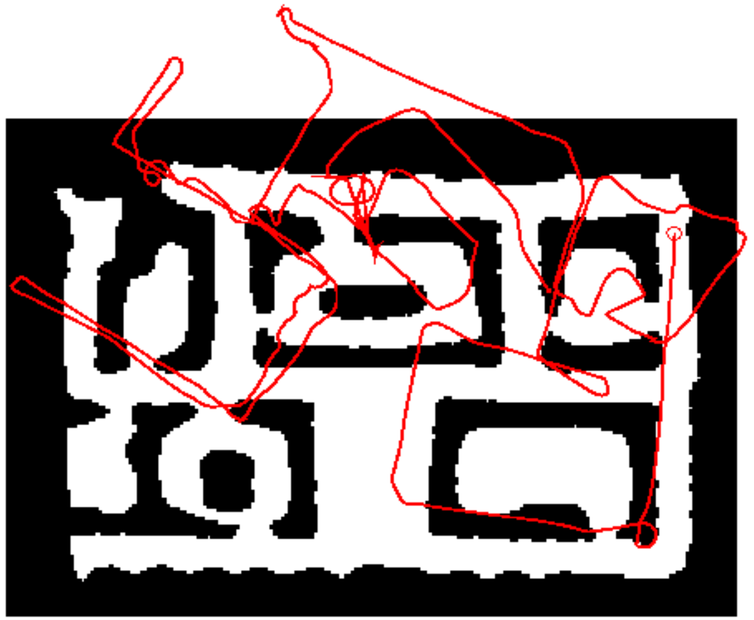
Bayes Filter

1. **Algorithm Bayes_Filter** $(bel(x_{t-1}),\ u_t,\ z_t)$:
2. **for** all x_t do
3. $\overline{bel}(x_t) = \Sigma_{x_{t-1}}\ p(x_t \mid u_t,\ x_{t-1})\ bel(x_{t-1})$ **(Prediction step)**
4. $bel(x_t) = \eta\ p(z_t \mid x_t)\ \overline{bel}(x_t)$
5. end **for**
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
  - Transition model: $p(x_t \mid u_t, x_{t-1})$
- How can we model $p(x_t \mid u_t, x_{t-1})$ based on kinematic equations?
  - Velocity model
  - Odometry model

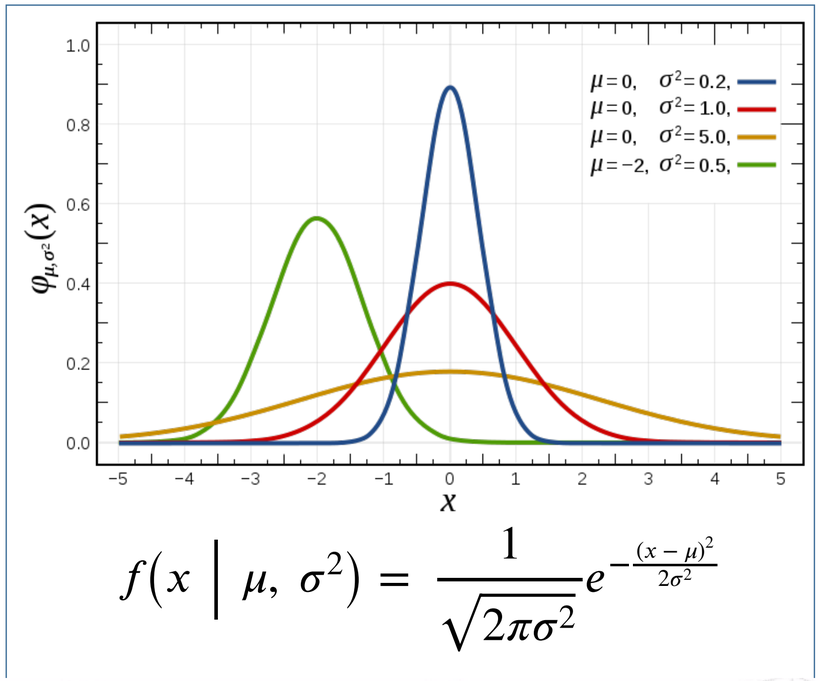


Probability Distributions

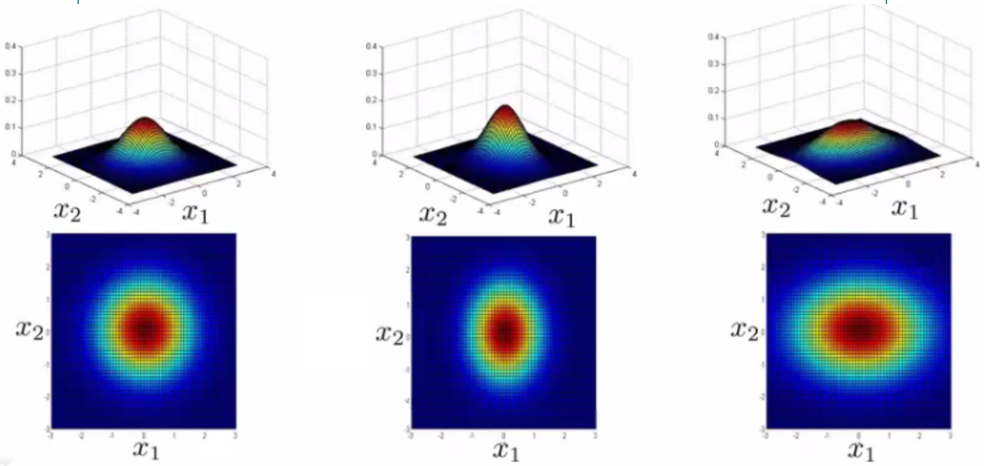
Quick Detour

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

1D Gaussian Probability Density Function



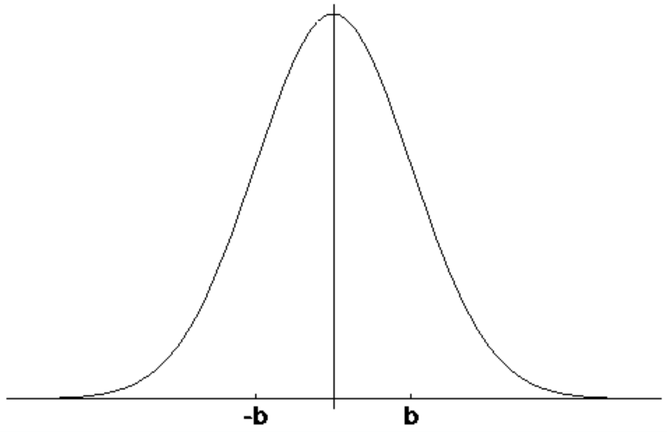
$$f\left(x \;\middle|\; \mu,\ \sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



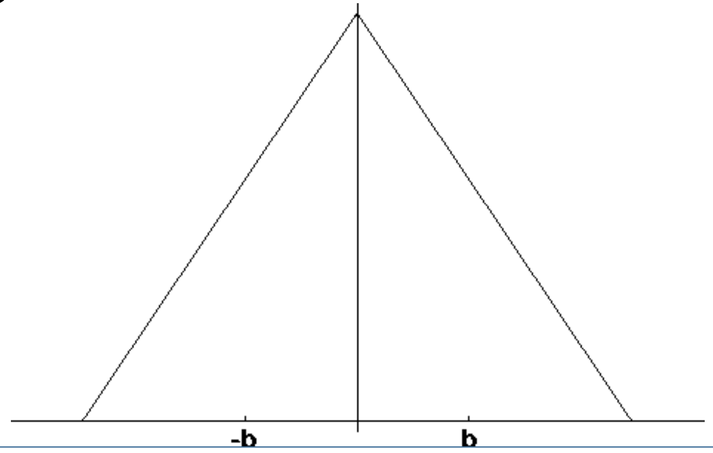
Probability Distributions

Quick Detour

- 3 inputs: $f(x \mid \mu, \sigma^2)$
- 2 inputs: $f(x - \mu \mid 0,\ \sigma^2)$
- Computationally cheaper alternative: triangular distributions



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

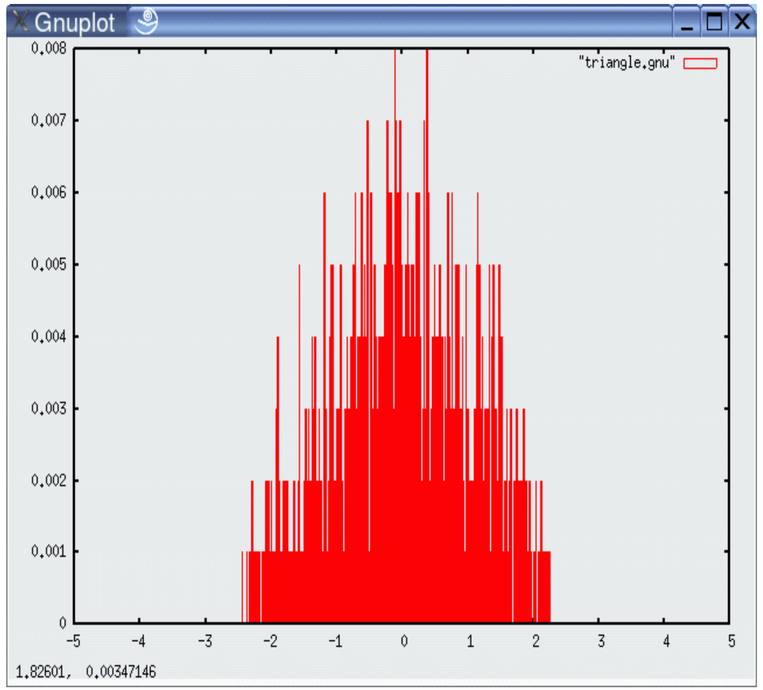


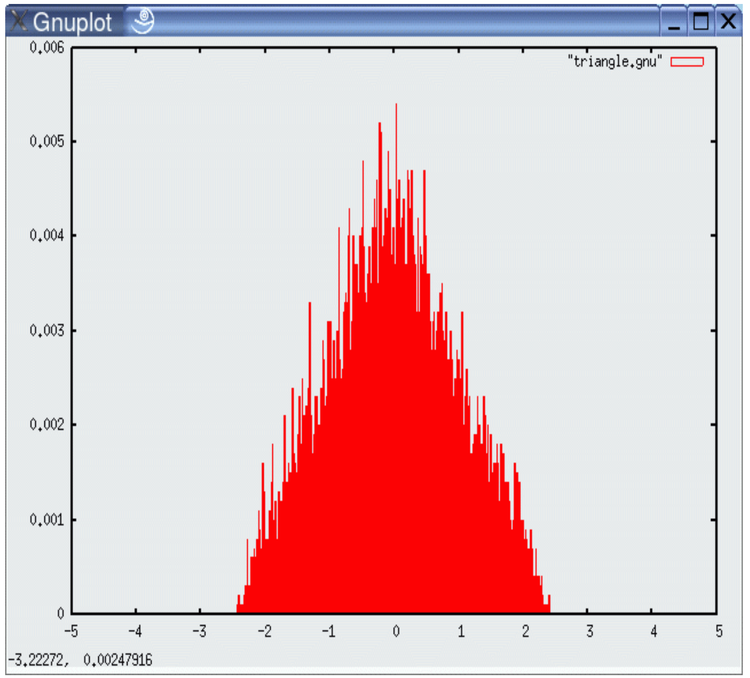
1. **Algorithm prob_triangular_distribution**(a, b^2) :
2. **return** $max\left(0, \frac{1}{\sqrt{6}\ b} - \frac{|a|}{6\ b^2}\right)$

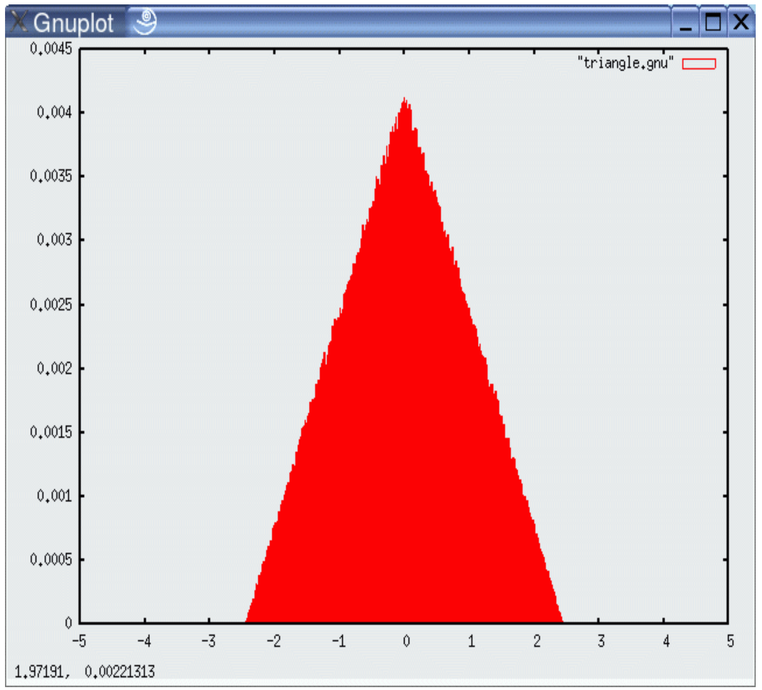
Probability Distributions

Quick Detour

- Sampling algorithms output samples from a given distribution
- Often used to approximate distributions

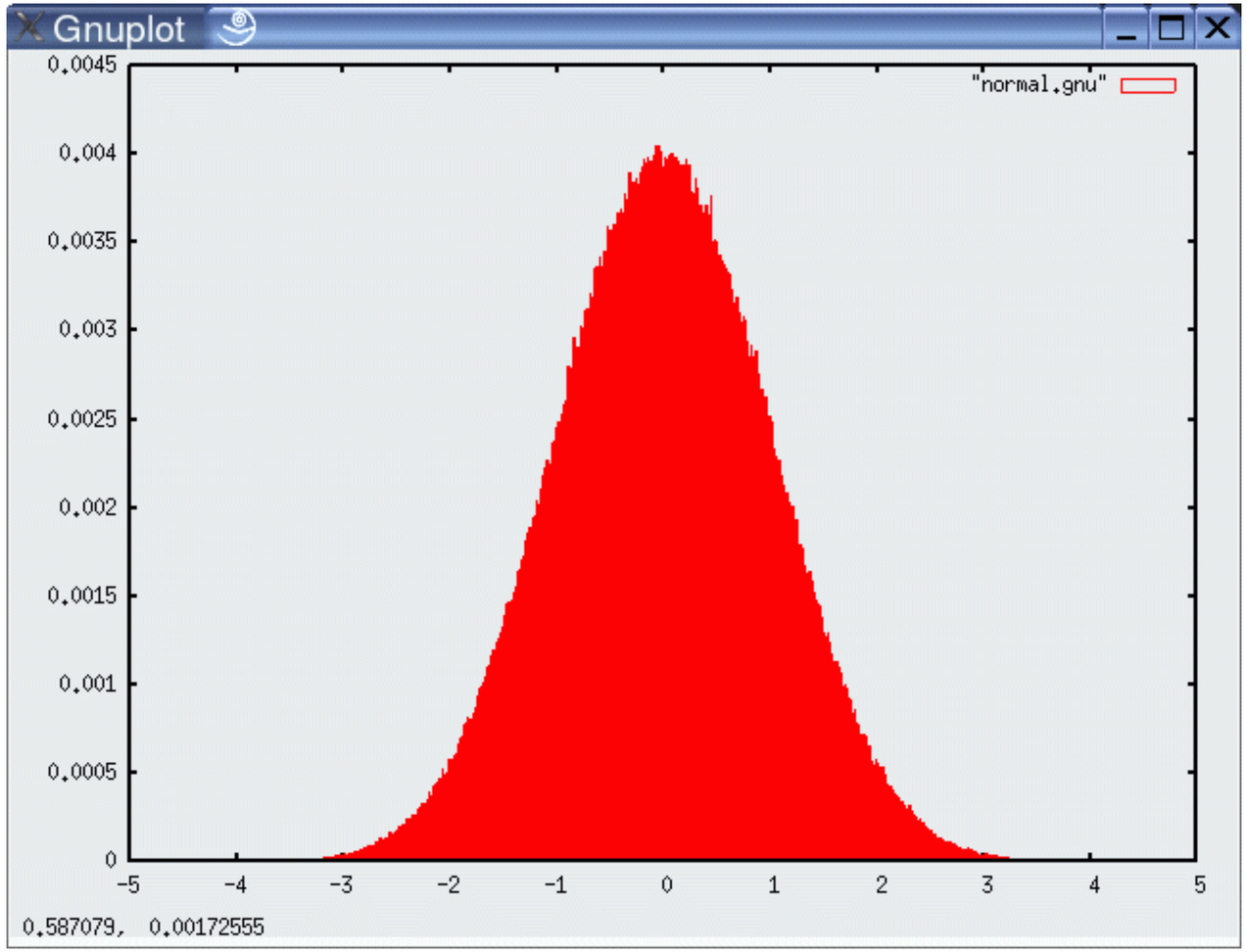






Probability Distributions

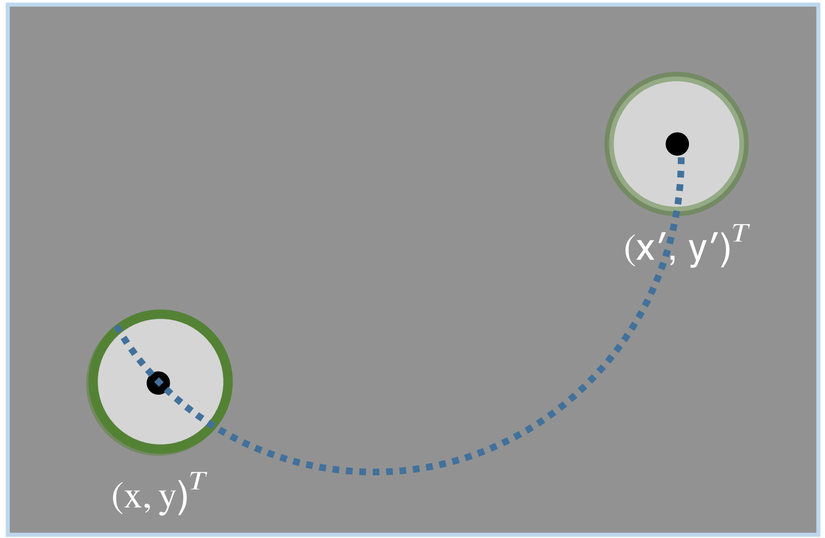
Quick Detour

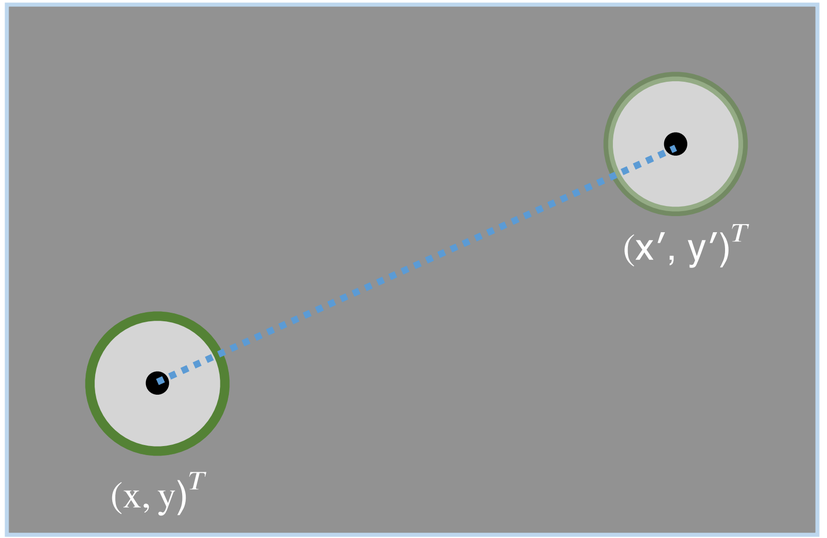


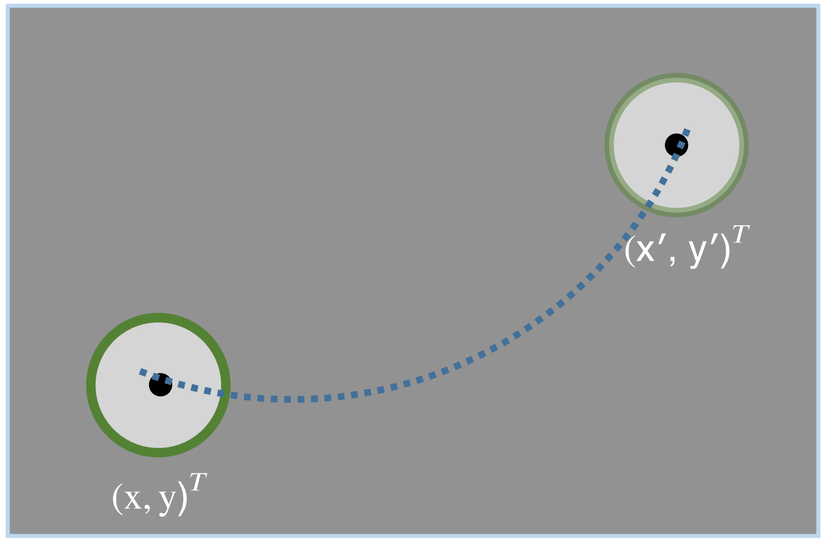
Velocity Model

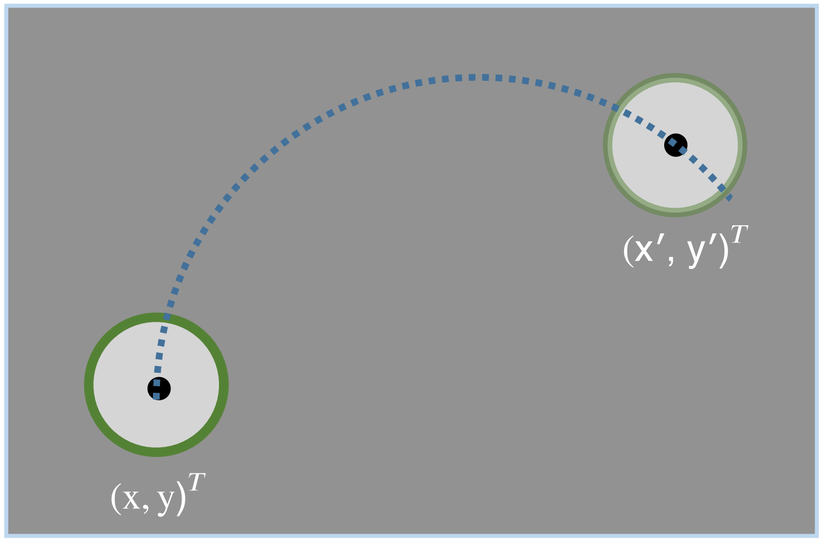
Parameters

- $u = (v_{right}, v_{left})$
- $u = (v_{COM}, \omega_{COM})$

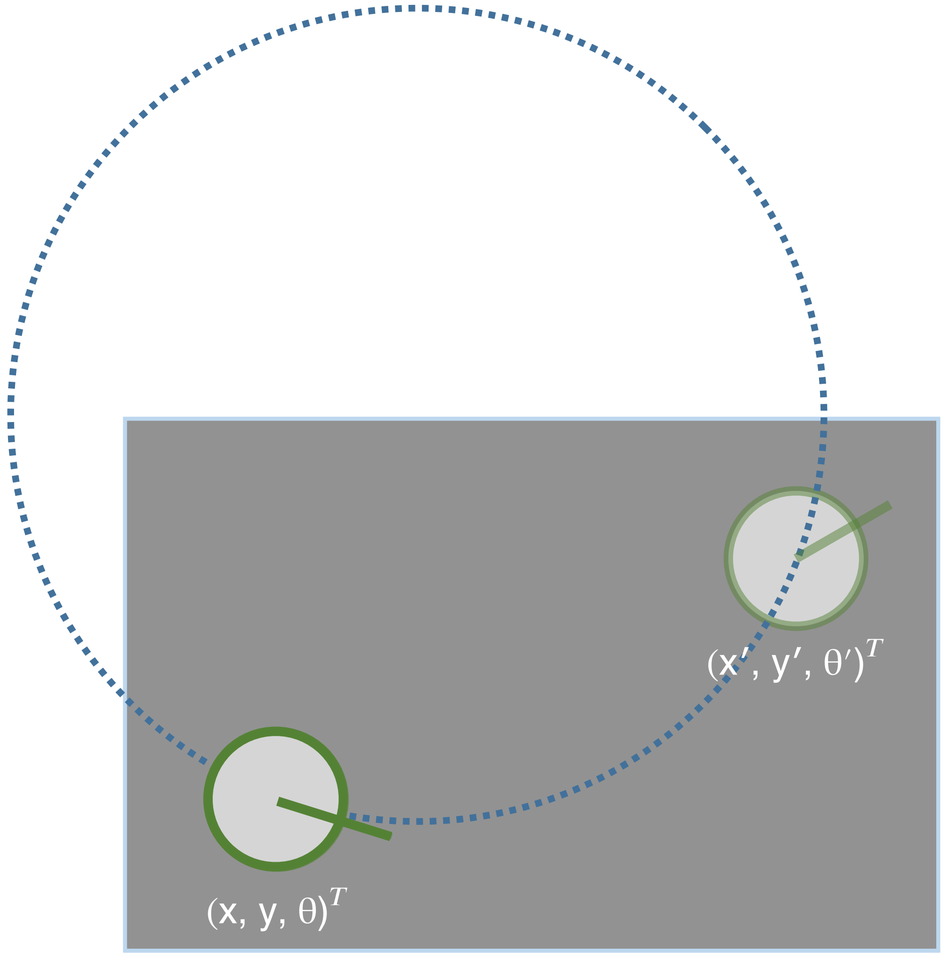




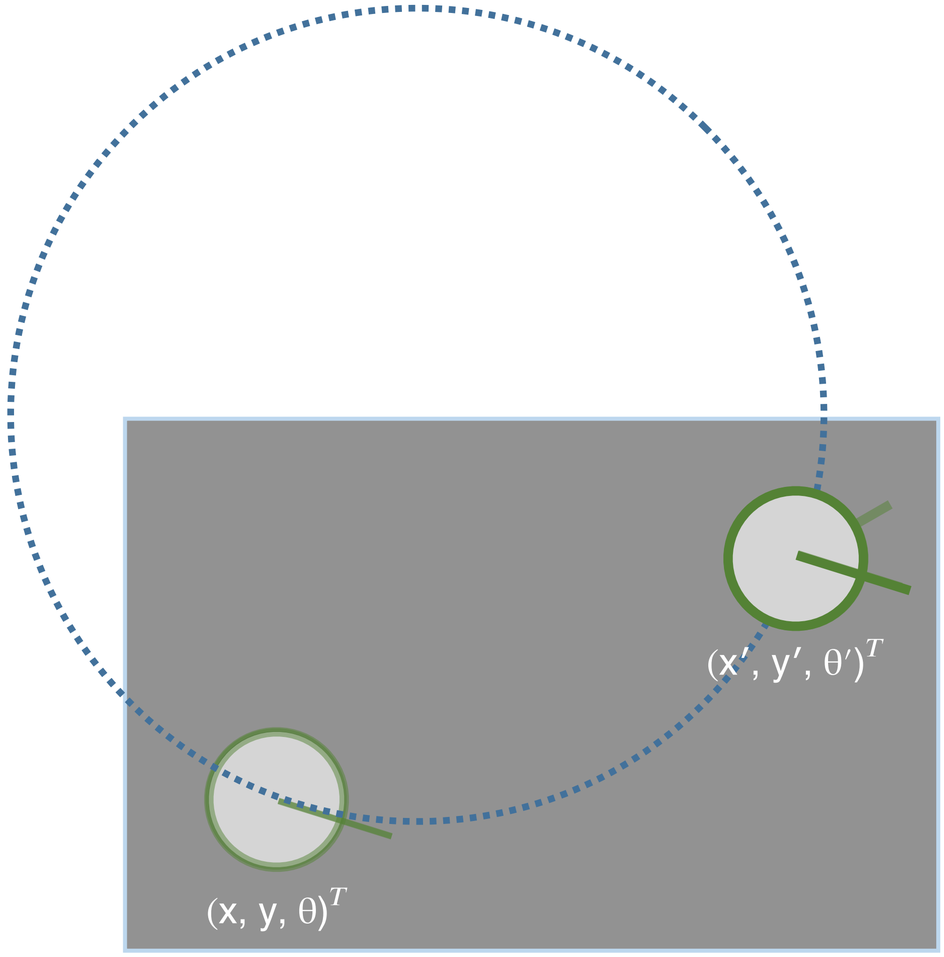




Parameters



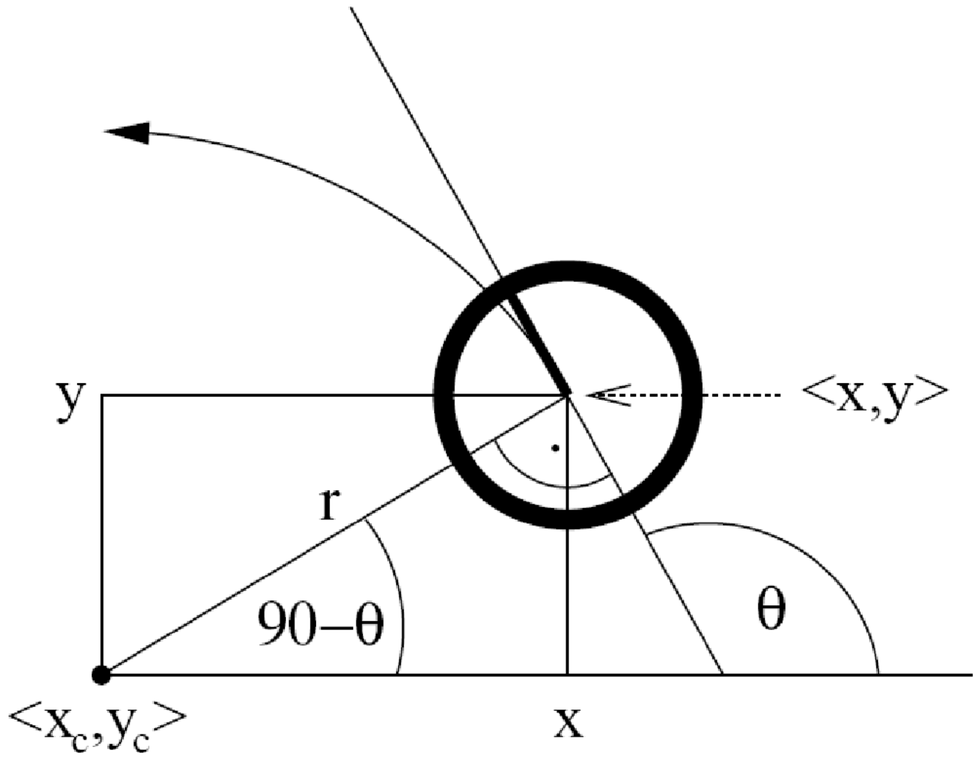
Parameters



Rotation γ at new pose

Velocity model

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



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?
  - $f(v_t \mid \hat{v}, \sigma_v^2)$
  - $f(\omega_t \mid \hat{\omega}, \sigma_\omega^2)$
  - $f(\gamma_t \mid \hat{\gamma}, \sigma_\gamma^2)$

Velocity model

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

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

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

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

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

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

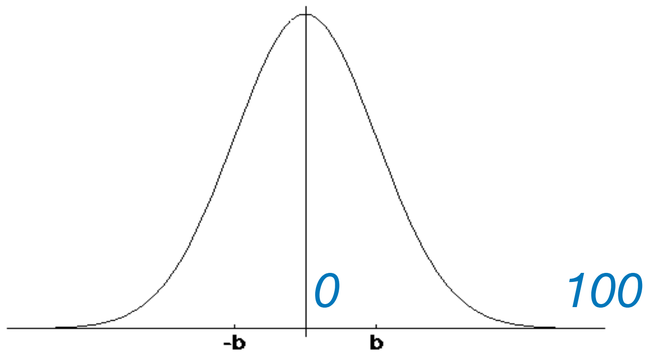
$$\hat{v} = \frac{\Delta\theta}{\Delta t} r^*$$

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

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

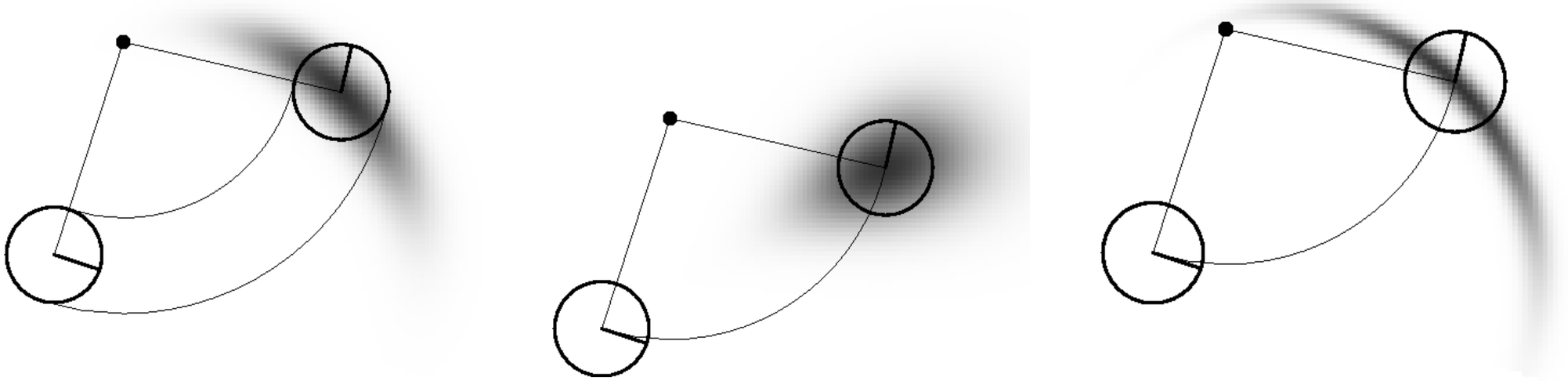
return $\textbf{prob}(v-\hat{v}, \alpha_1|v| + \alpha_2|\omega|) \cdot \textbf{prob}(\omega-\hat{\omega}, \alpha_3|v| + \alpha_4|\omega|) \cdot \textbf{prob}(\hat{\gamma}, \alpha_5|v| + \alpha_6|\omega|)$

- Calculate the error-free control between the states x_{t-1} and x_t
- How to add probability?
  - $f(v_t \mid \hat{v}, \sigma_v^2)$
    - $f(v_t - \hat{v} \mid 0, \sigma_v^2)$
  - $f(\omega_t \mid \hat{\omega}, \sigma_\omega^2)$



Velocity motion model

(darker regions are more probable)



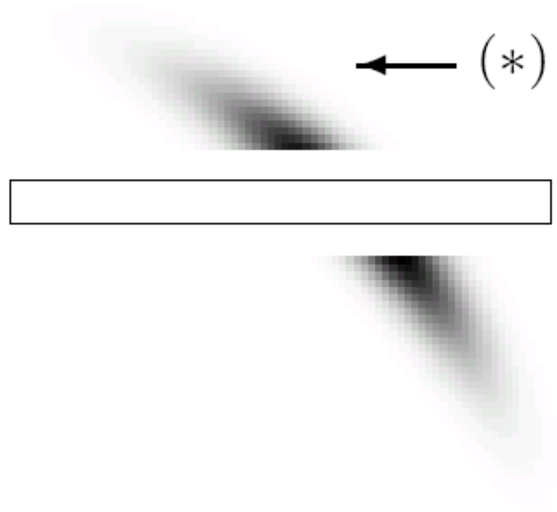
- The velocity motion model for different noise parameter settings for the same control projected in the x-y space

Velocity motion model with a map

(a)

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

(b)



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

Sampling from velocity model

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

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

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

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

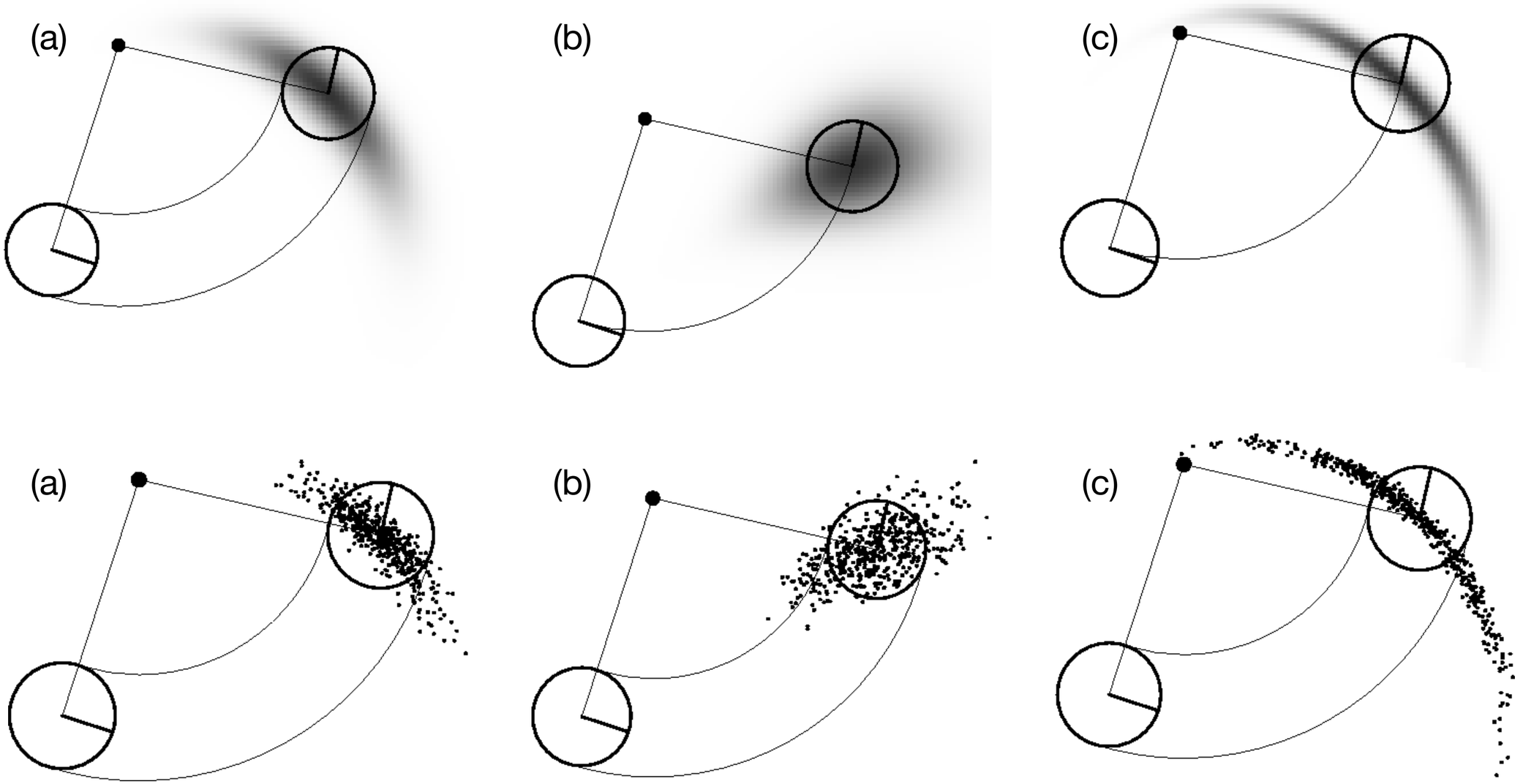
$x' = x - \frac{\hat{v}}{\hat{\omega}} \sin\theta + \frac{\hat{v}}{\hat{\omega}} \, sin(\theta + \hat{\omega}\Delta t)$

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

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

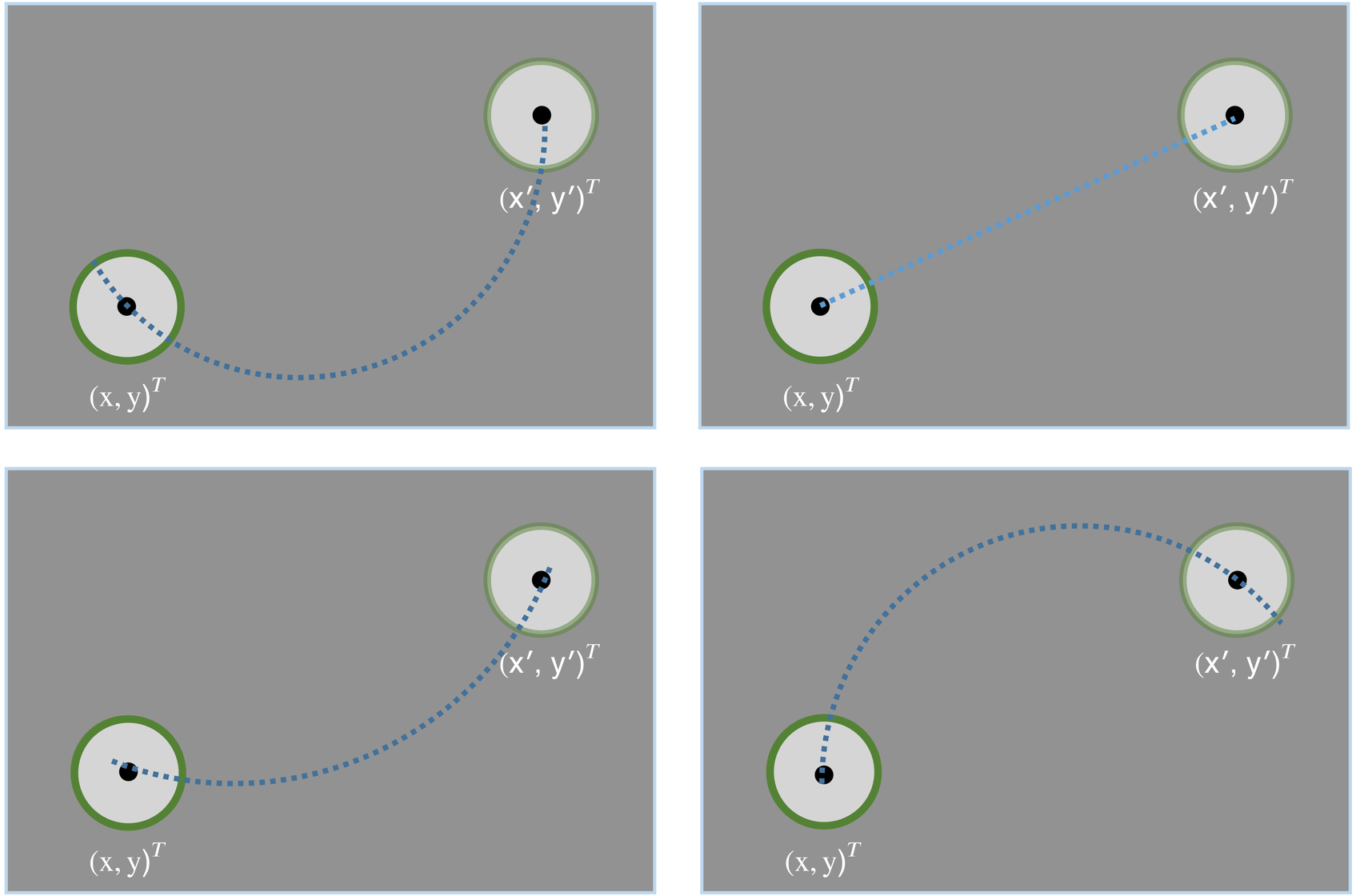
return $x_t = (x', y', \theta')^T$

Sampling from velocity model



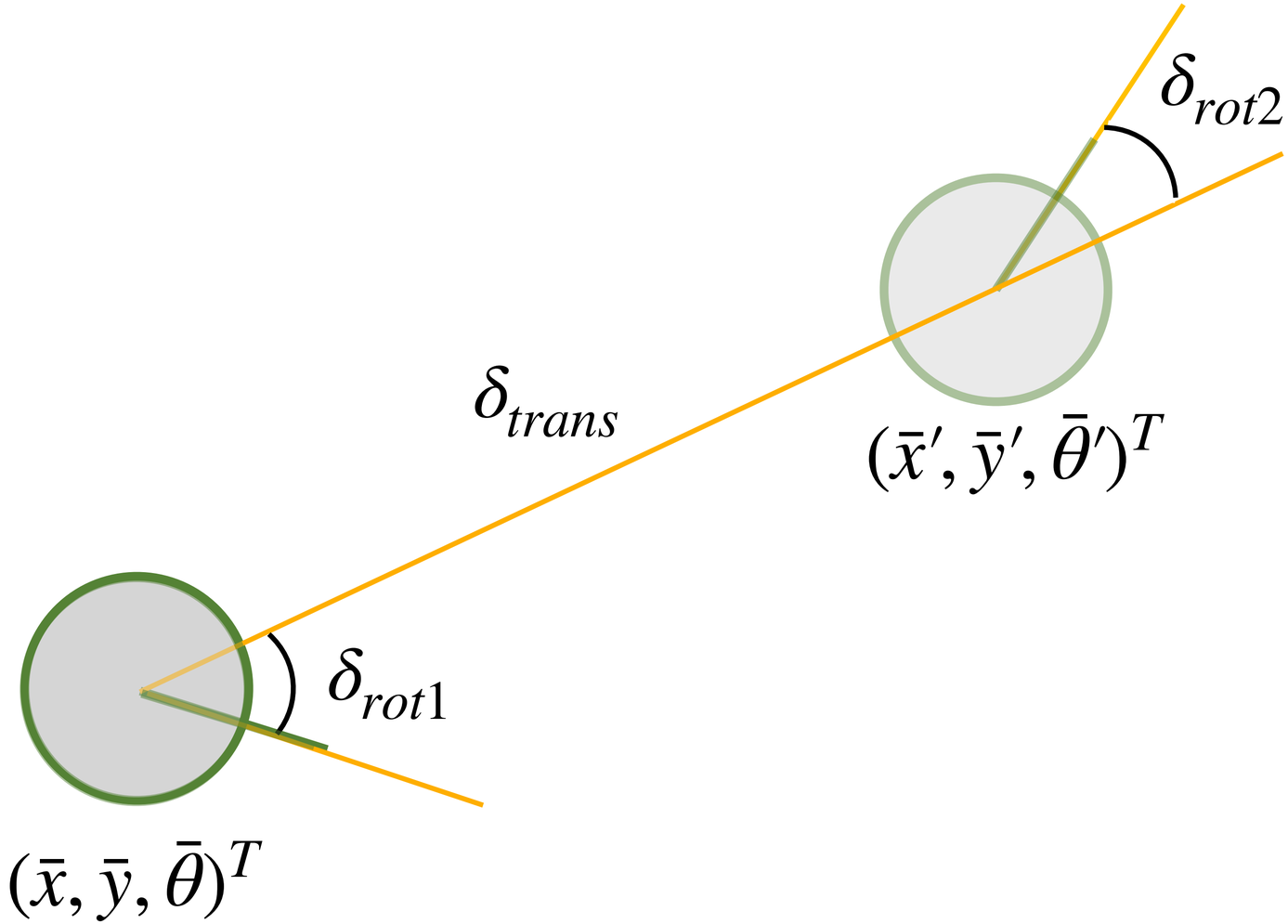
Parameters

- $u = (v_{right}, v_{left})$
- $u = (v_{COM}, \omega_{COM})$
- How would you use this in your system?
- Pros
  - Prediction/planning
- Cons
  - Parameter tuning
  - Inaccurate



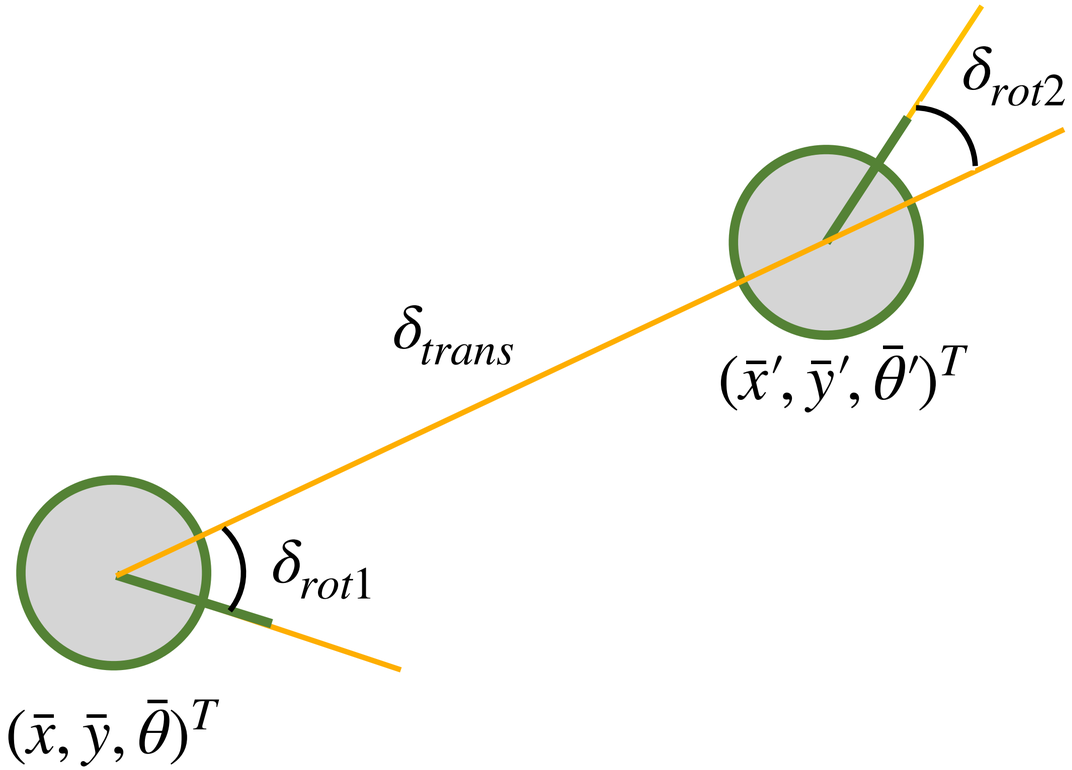
Odometry Model $u_t = (\overline{x_{t-1}}, \bar{x}_t)^T$

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

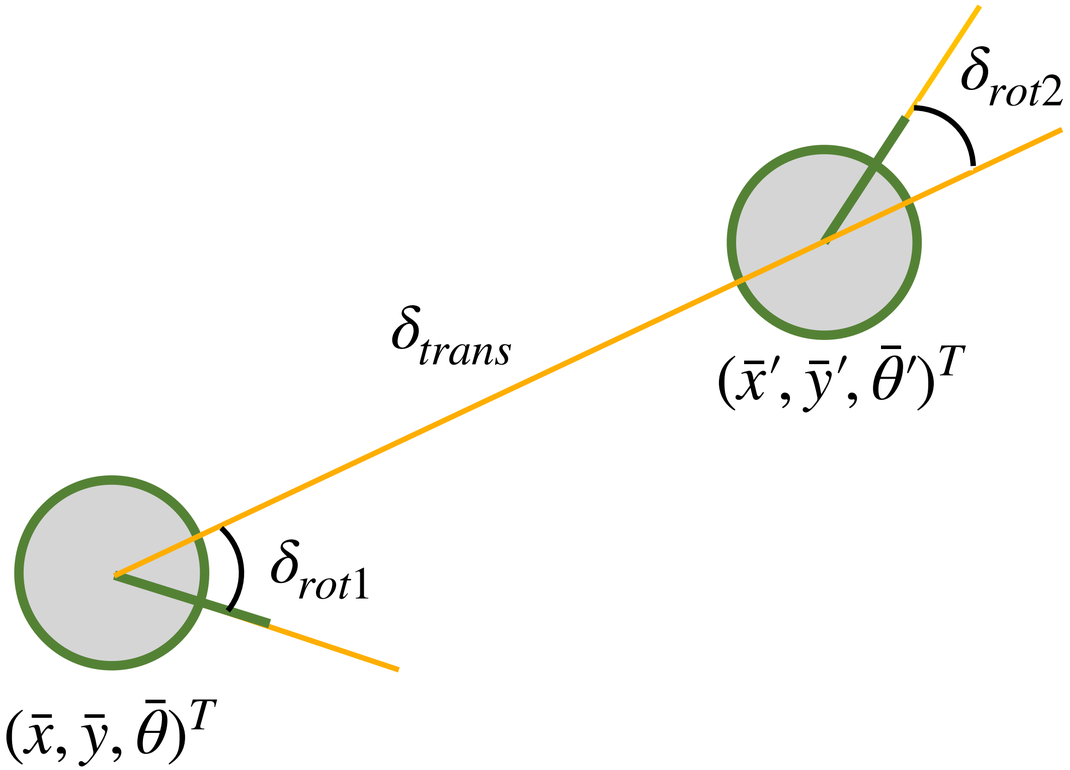


Odometry Model Parameters

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

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

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



Odometry Model Algorithm

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
(what the robot did)

Calculate the relative motion parameters for the given states x_{t-1} and x_t
(what the robot did ideally)

Odometry Sampling Model Algorithm

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

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

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

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

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

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

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

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

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

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

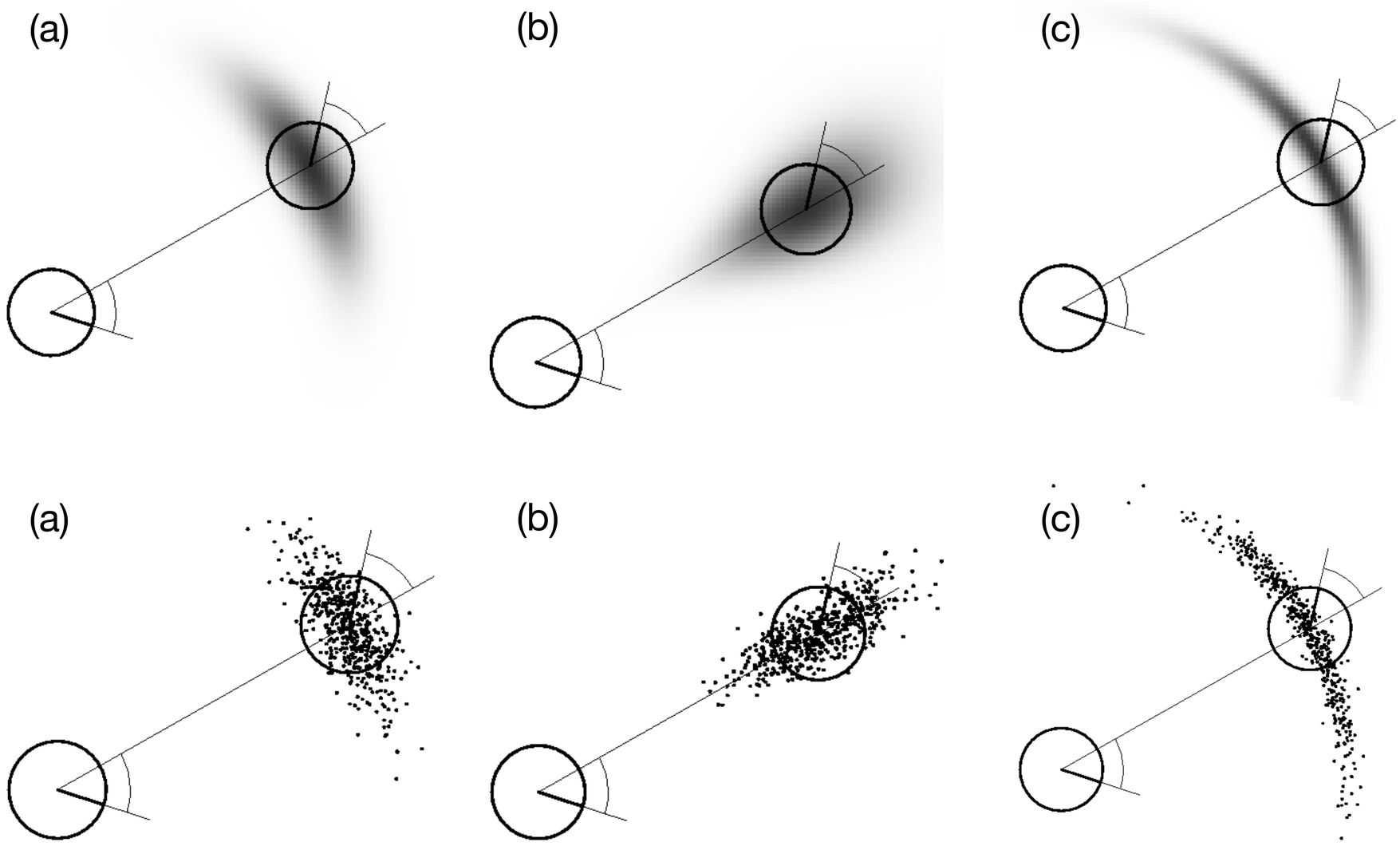
11. **return** $x_t = (x', y', \theta')^T$

Calculate the relative motion parameters from odometry readings

Add noise to calculated motion parameters

Calculate the sample state

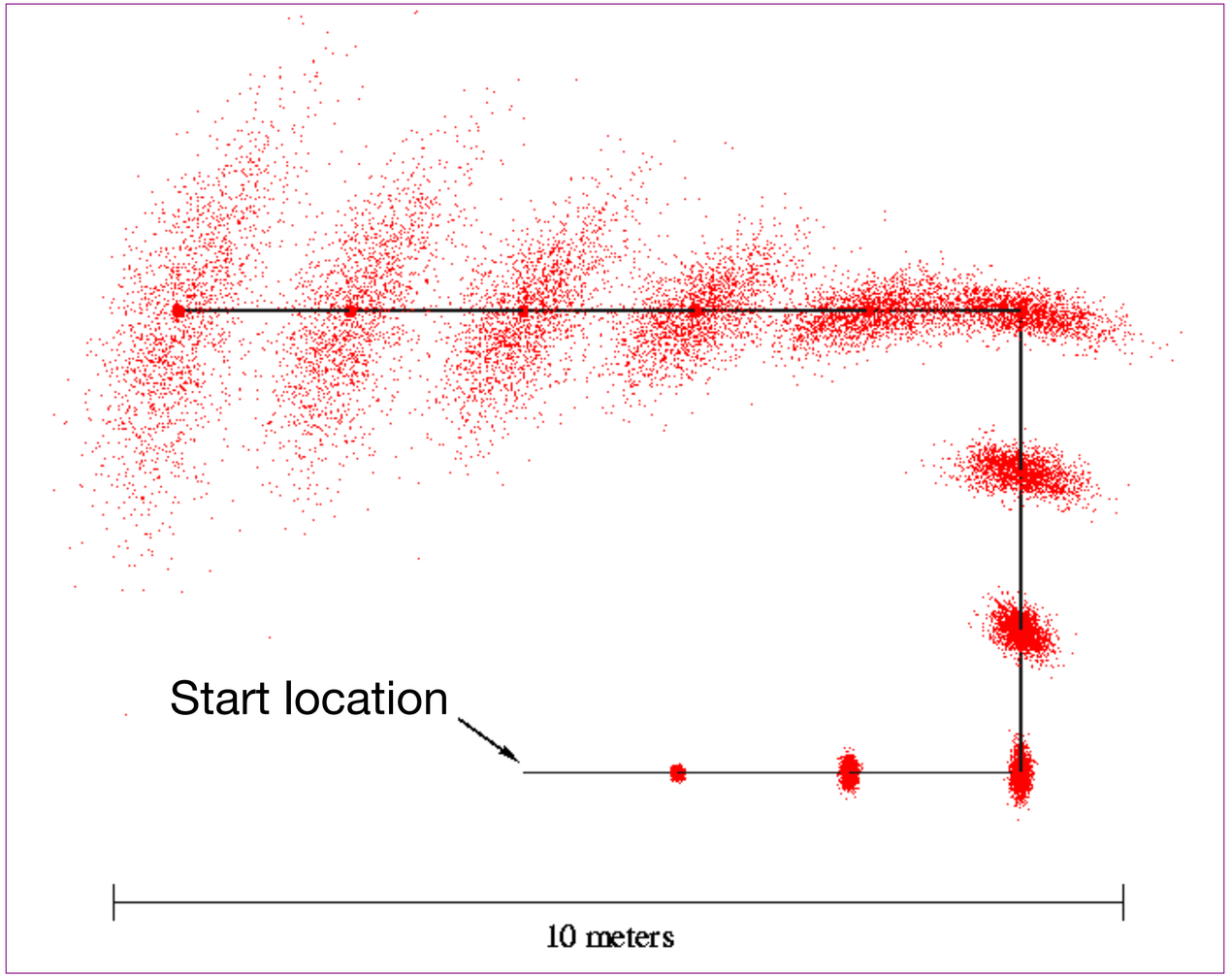
Sampling from Odometry Model



Odometry model

- $u_t = (\bar{x}_{t-1}, \bar{x}_t)^T$
- How would you use this model in your system?
- Odometry is available **after the robot has moved**
  - Can be used for estimation algorithms (e.g., localization and mapping)
  - Cannot be used for prediction (e.g., probabilistic motion planning)

Repeated sampling from our odometry motion model



References

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