# ECE 4960

Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

Vivek Thangavelu
vs353@cornell.edu

# Fast Robots

# Probabilistic Motion Model

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

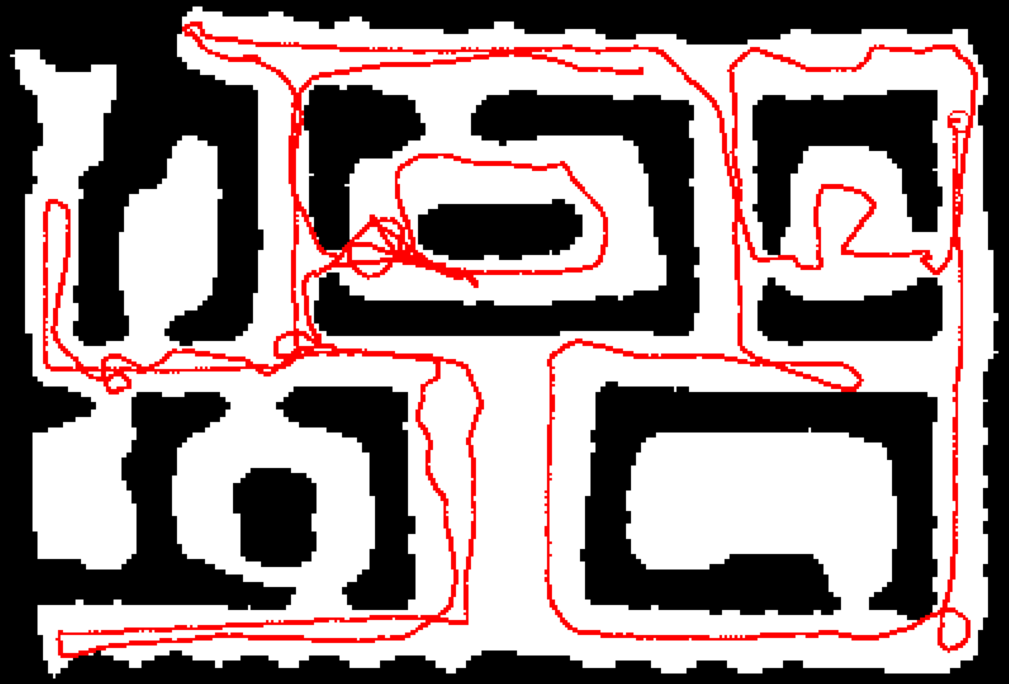
# Bayes Filter

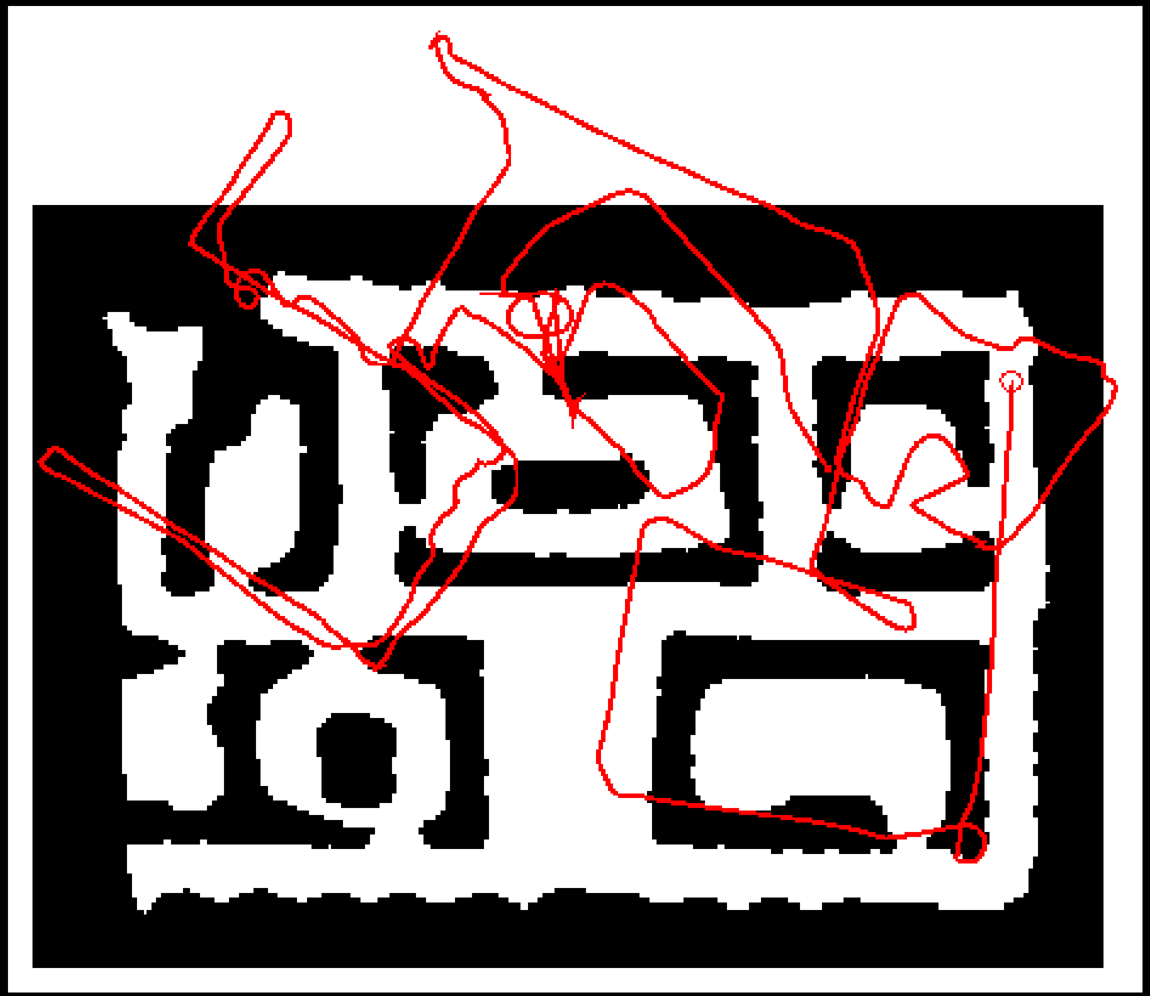
1. **Algorithm Bayes_Filter** $(bel(x_{t-1}), u_t, z_t)$:
2. for all $x_t$ do
3. $\overline{bel}(x_t) = \sum_{x_{t-1}} \boxed{p(x_t|u_t, x_{t-1})}\; bel(x_{t-1})$ [ Prediction Step ]
4. $bel(x_t) = \eta\, p(z_t|x_t)\; \overline{bel}(x_t)$
5. endfor
6. return $bel(x_t)$

Transition Probability / Action Model

# Robot Motion

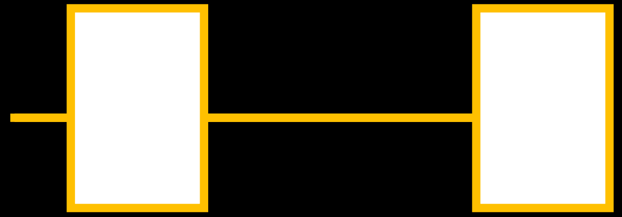
- Robot motion is inherently uncertain.
- How can we model this uncertainty?



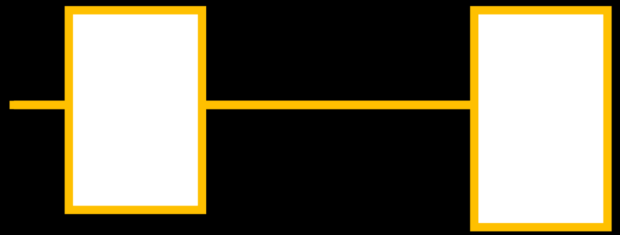


# Robot Motion

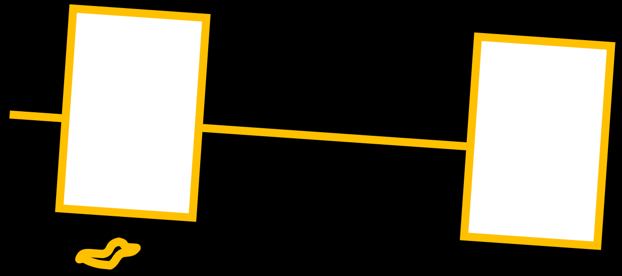
# Some reasons for Motion Errors



ideal case



different wheel diameters



bump

carpet

and many more …

# Probabilistic Motion Model

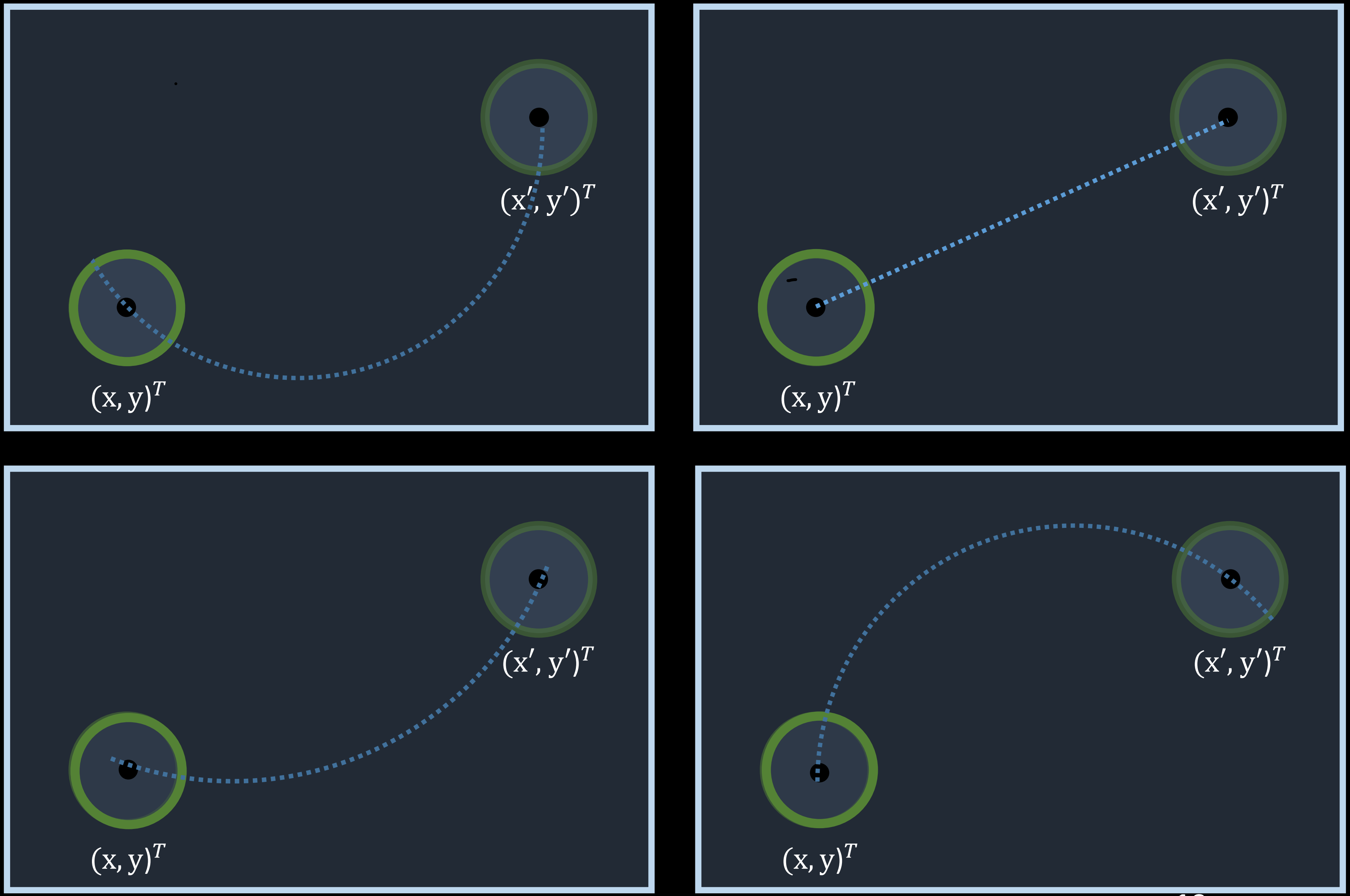
- We consider mobile robot kinematics for robots operating in planar environments
- Robot pose $x_t = (x, y, \theta)^T$
- Can be easily extended to other types of mobile robots or manipulators

# Probabilistic Motion Model

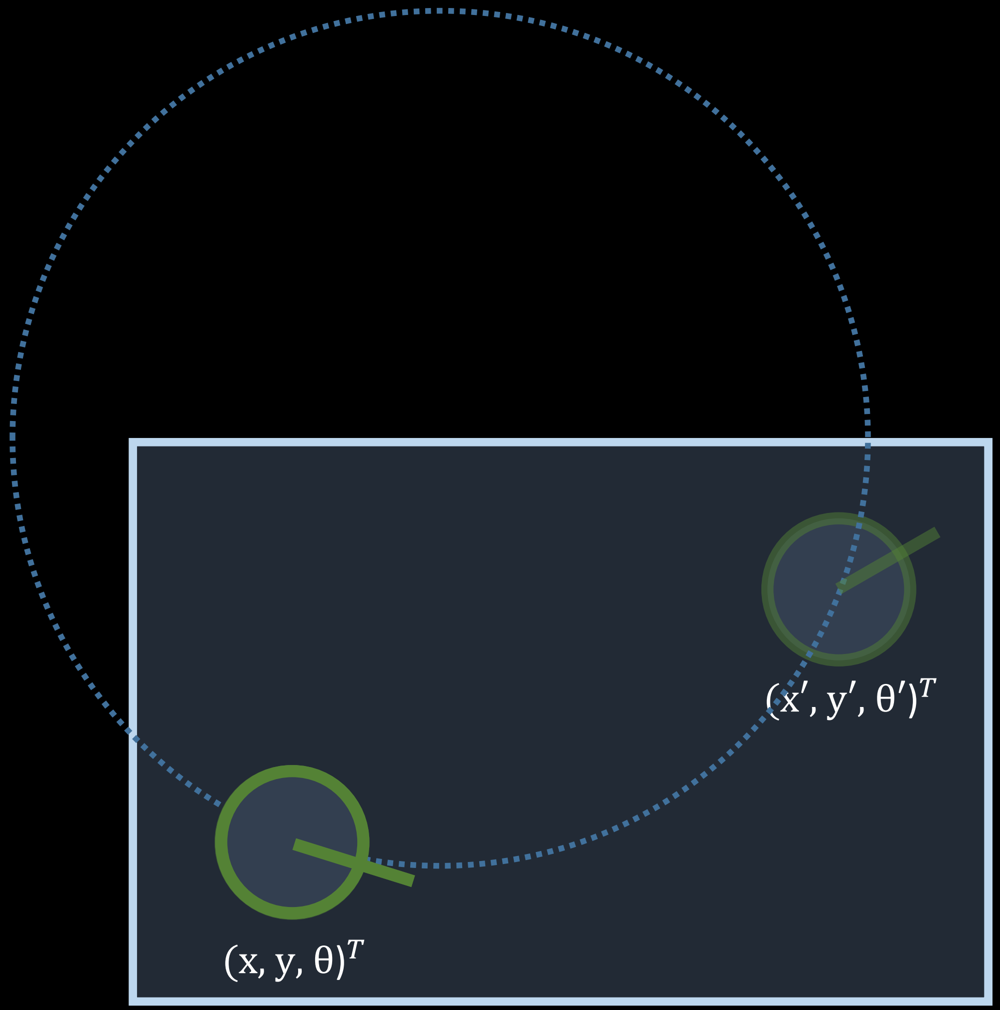
- Generalizes kinematic equations to the fact that the outcome of a control action is uncertain, due to control noise or other non-modelled external factors
- To implement the Bayes Filter, we need the state transition model $p(x_t \mid u_t, x_{t-1})$
- $p(x_t \mid u_t, x_{t-1})$ specifies a posterior probability, that action $u_t$ carries the robot from $x_t$ to $x_{t-1}$
- How can we model $p(x_t \mid u_t, x_{t-1})$ based on the kinematic equations?
- Two Motion Models:
  - Velocity Model
  - Odometry Model

# Velocity Model

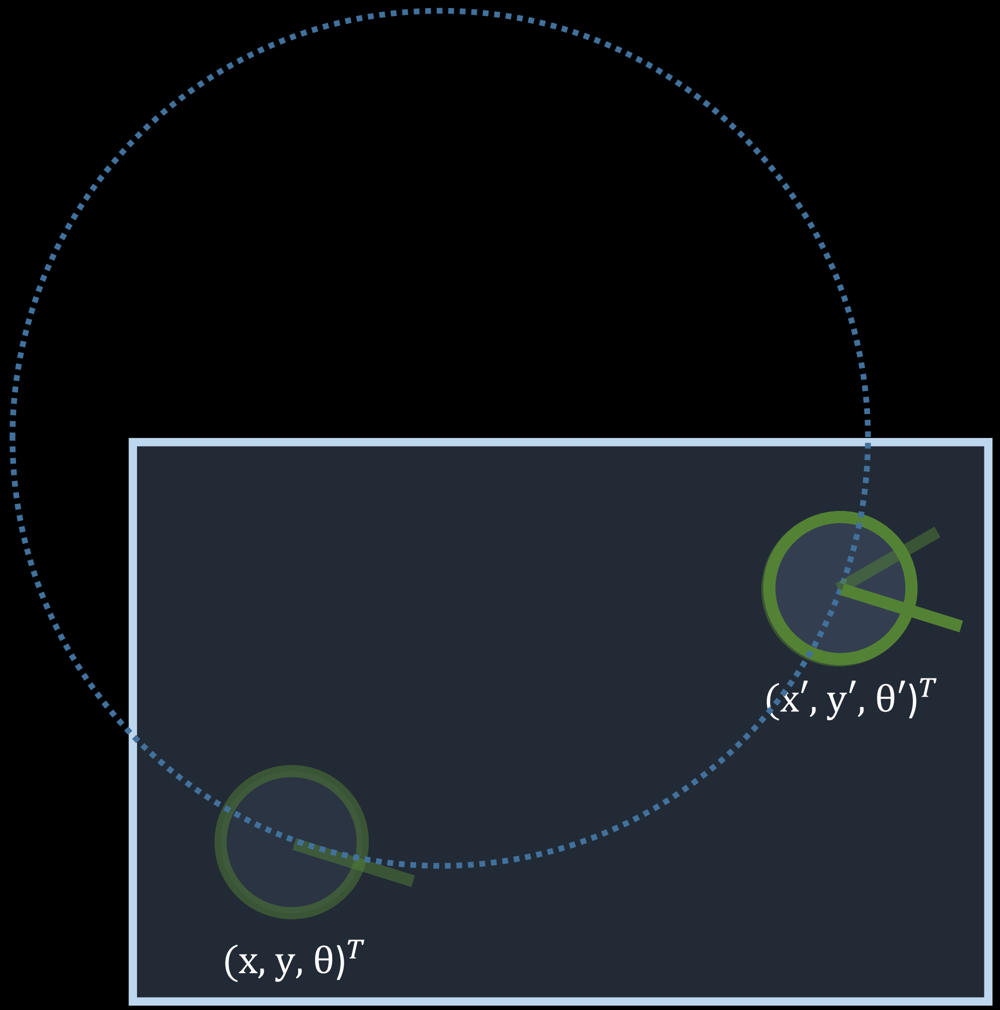
# Odometry Model Parameters



# Odometry Model Parameters

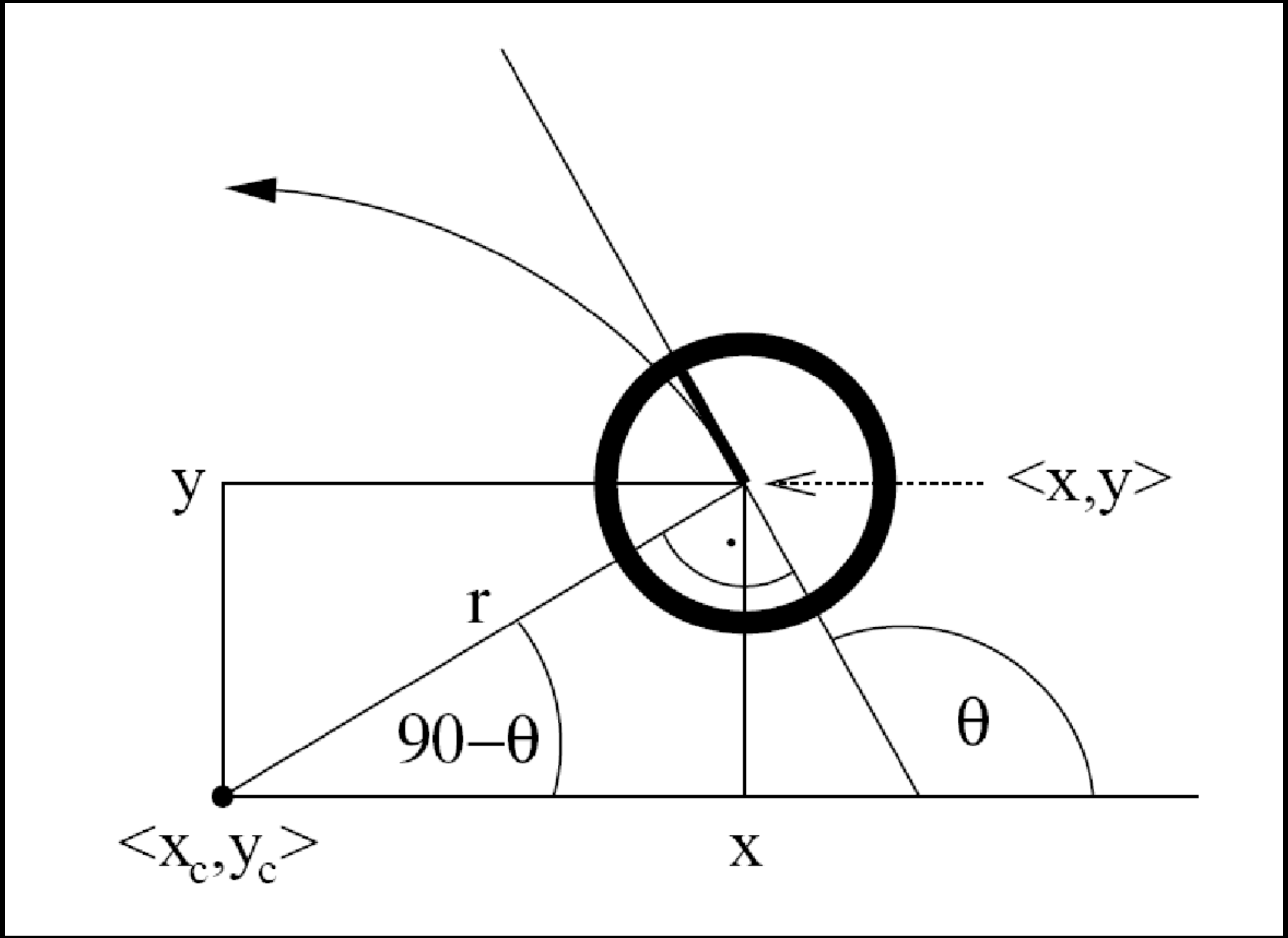


# Odometry Model Parameters



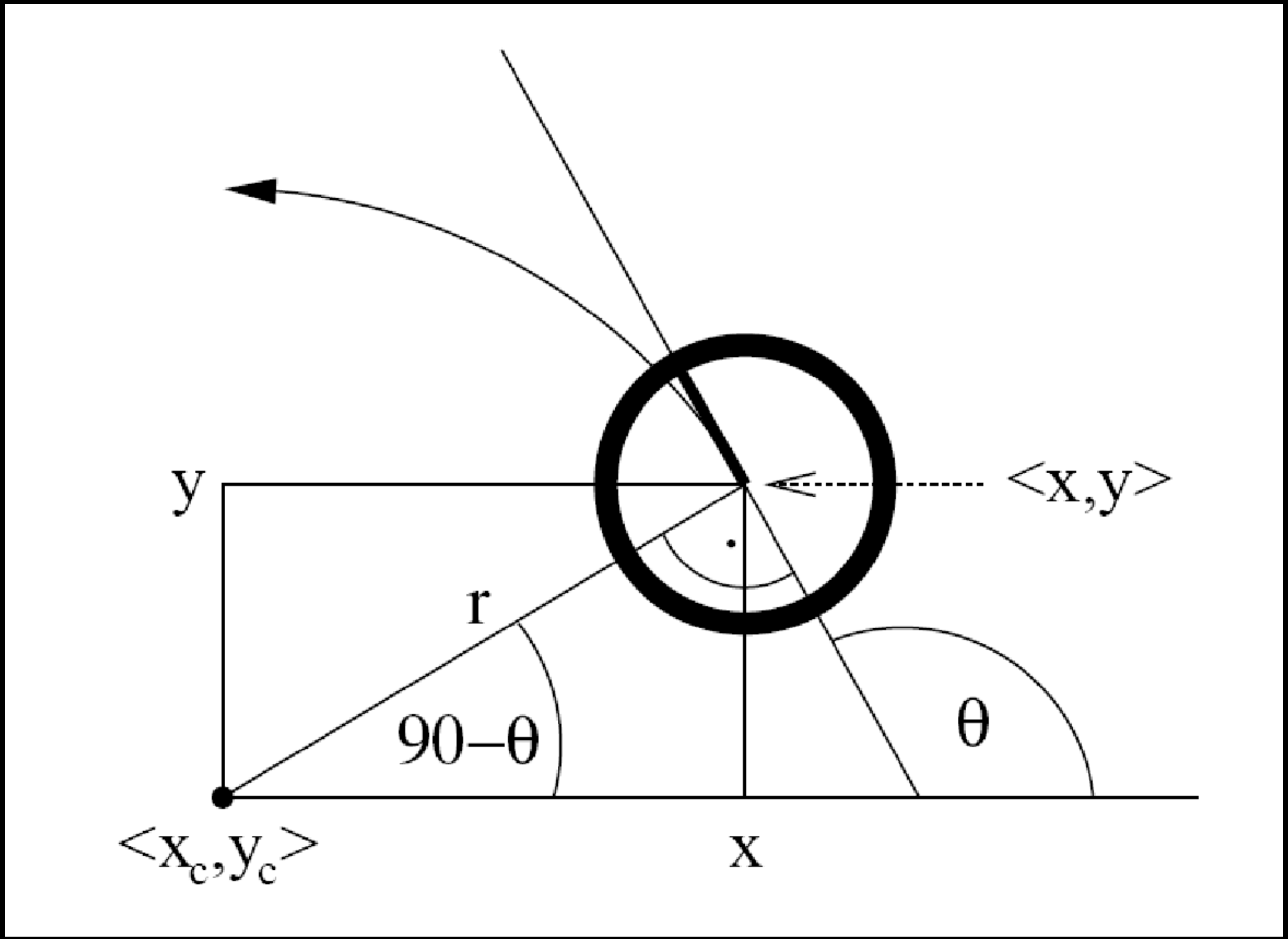
# Velocity Model

- The control data $u_t$ is specified by velocity commands given to the robot
- Velocities given to the robot can be translational $v_t$ or rotational $\omega_t$
- Control data $u_t = (v_t,\ \omega_t)$



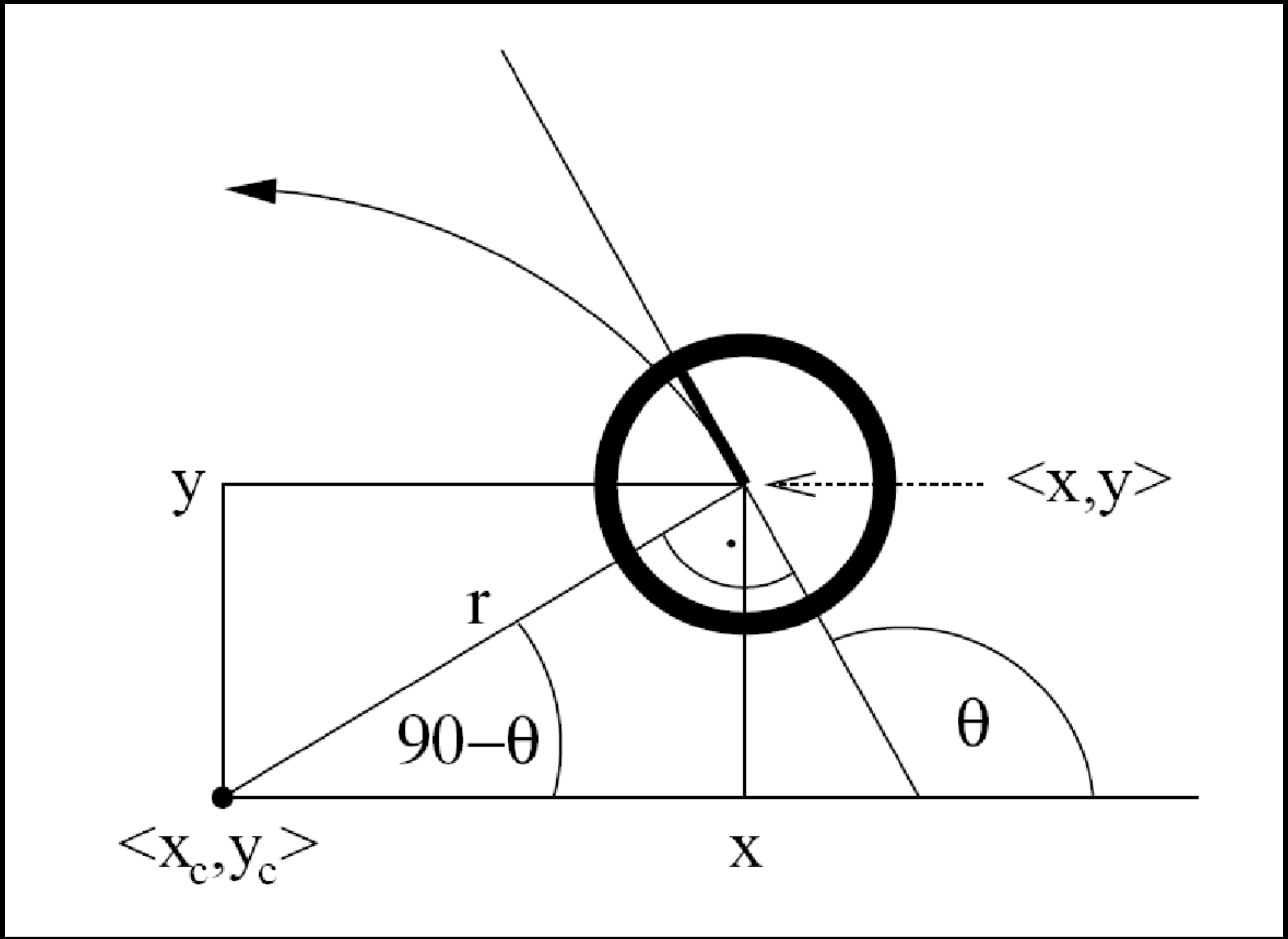
# Velocity Model

- If both velocity components are kept at a fixed value for the entire time interval $(\mathrm{t}-1, \mathrm{t}]$ then the robot moves in a circular with radius r and center $(x_c, y_c)$
- Exact motion $x_t = (x', y', \theta')^T$ may be calculated given $x_{t-1} = (x, y, \theta)^T$ and $u_t = (v_t,\ \omega_t)^T$ using trigonometric equations



# Velocity Model - Degeneracy

- The robot moves in exact circular paths
- The velocity controls is 2d dimensional leading to state changes in a 3d pose space
- We perform a final orientation $\gamma$ when it arrives at its final pose



# Probabilistic Motion Model

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

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

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

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

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

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

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

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

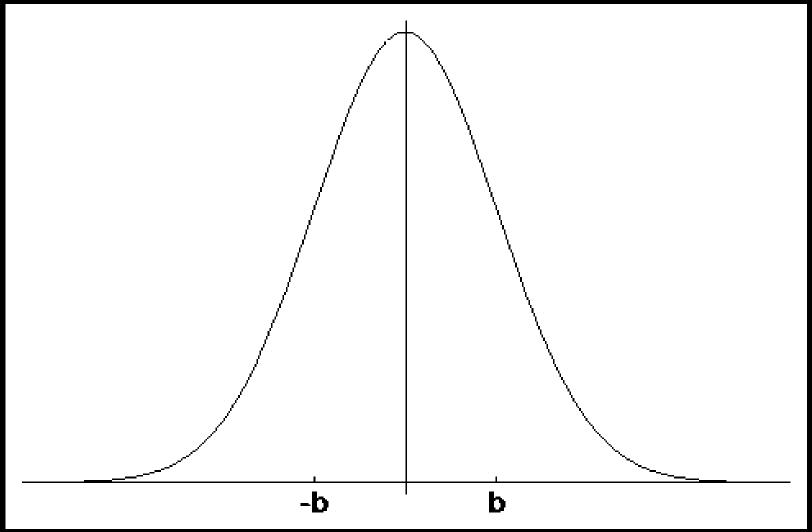
$$\hat{\gamma} = \frac{\theta' - \theta}{\Delta t} - \hat{\omega}$$

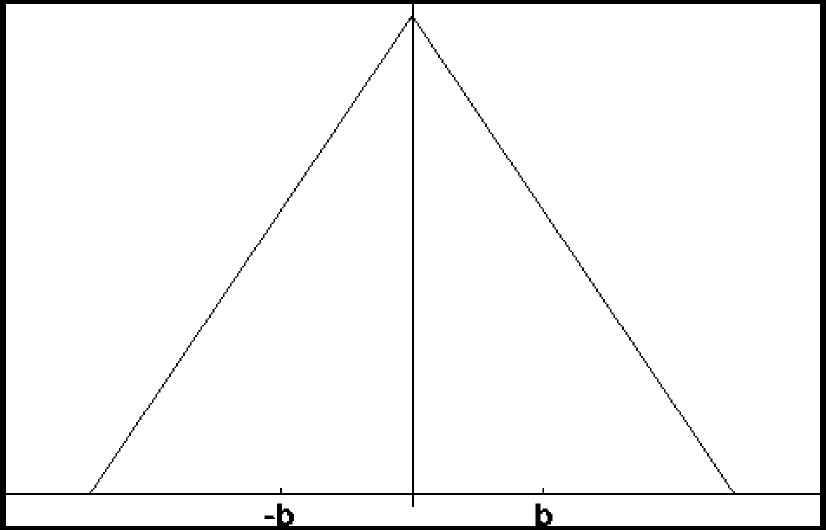
Calculate the error-free control between the states $x_{t-1}$ and $x_t$

For completeness, the robot performs an additional rotation

Algorithm for computing $p(x_t \mid u_t, x_{t-1})$, based on velocity information, where $x_{t-1} = (x, y, \theta)^T$, $x_t = (x', y', \theta')^T$ and $u_t = (v_t, \omega_t)^T$.
The function prob($a,b^2$) computes the probability of its argument a under a zero-centered distribution with variance $b^2$. (prob can represented by a gaussian or a triangular distribution)

# 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` $max\left(0, \frac{1}{\sqrt{6}\ b} - \frac{|a|}{6\ b^2}\right)$

# Probabilistic Motion Model

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

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

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

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

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

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

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

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

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

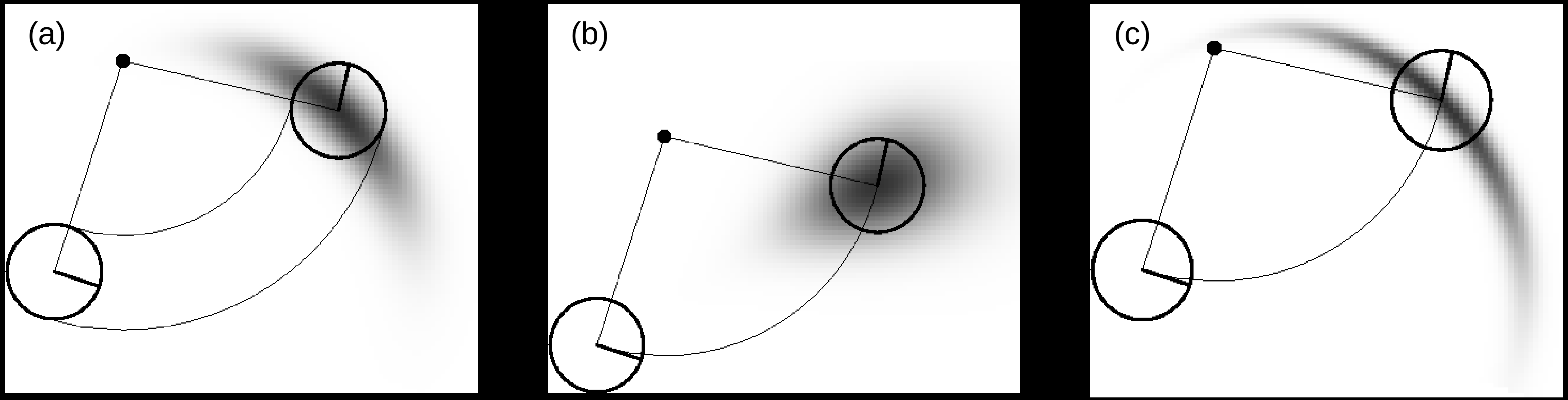
*return* $\mathbf{prob}(v - \hat{v}, \alpha_1|v| + \alpha_2|\omega|) \cdot \mathbf{prob}(\omega - \hat{\omega}, \alpha_3|v| + \alpha_4|\omega|) \cdot \mathbf{prob}(\hat{\gamma}, \alpha_5|v| + \alpha_6|\omega|)$

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

For completeness, the robot performs an additional rotation

Algorithm for computing $p(x_t \mid u_t, x_{t-1})$, based on velocity information, where $x_{t-1} = (x, y, \theta)^T$, $x_t = (x', y', \theta')^T$ and $u_t = (v_t, \omega_t)^T$.
The function prob$(a,b^2)$ computes the probability of its argument a under a zero-centered distribution with variance $b^2$. (prob can represented by a gaussian or a triangular distribution)

# Velocity Motion Model

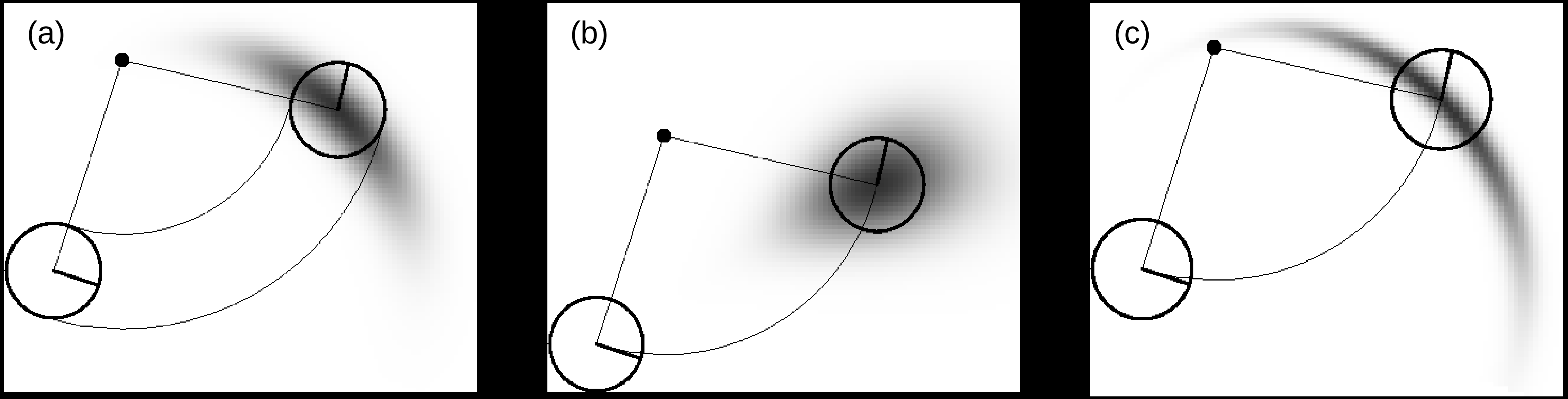



(darker regions are more probable)

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) Moderate error parameters

# Velocity Motion Model




(darker regions are more probable)

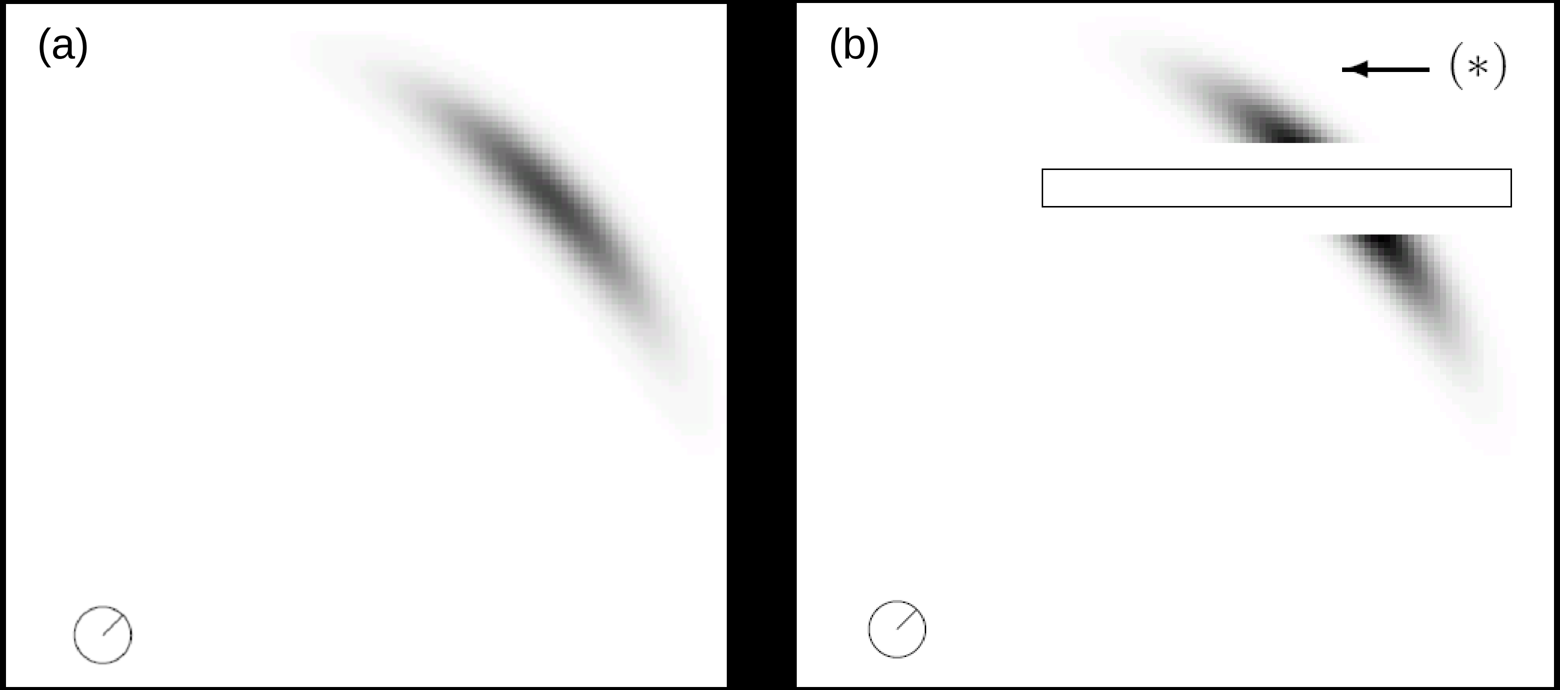
The velocity motion model for different noise parameters settings for the same control $u_t = (v_t,\ \omega_t)^T$ projected in the x-y space

a) Moderate error parameters

b) Smaller angular error parameters but larger transitional errors

c) Large angular and translational error parameters

# Velocity Model with a Map




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

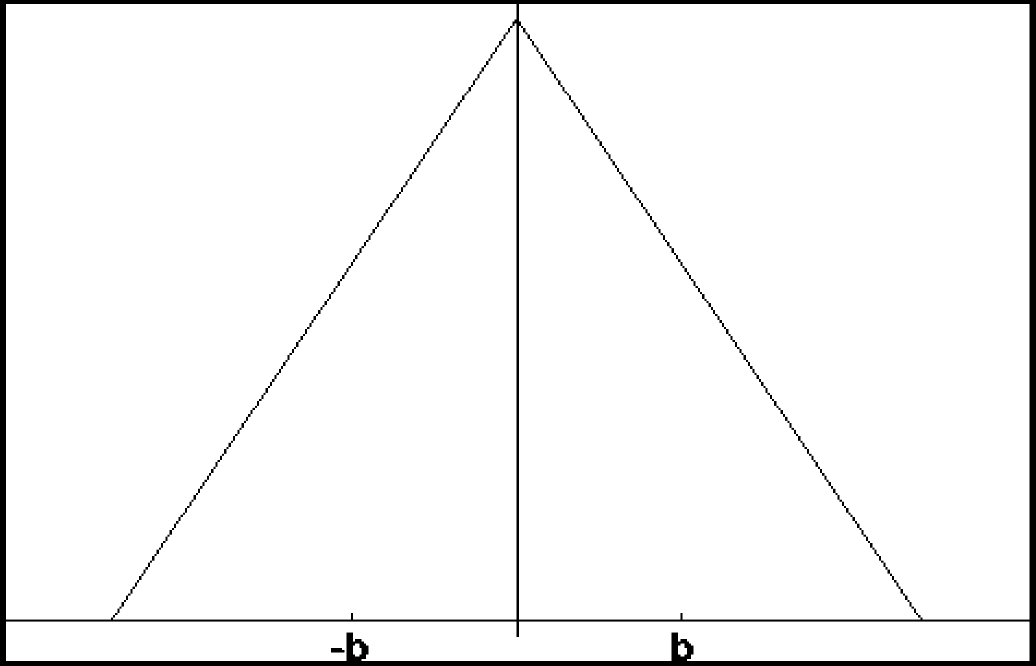
(a) Velocity model without a map

(b) Velocity model with a map: $p(x_t \mid u_t\,, x_{t-1}, m)$ is calculated from the normal product of $p(x_t \mid u_t\,, x_{t-1})$ and $p(x_t \mid m)$. It is zero in the extended obstacle error and $p(x_t \mid u_t\,, x_{t-1})$ everywhere else

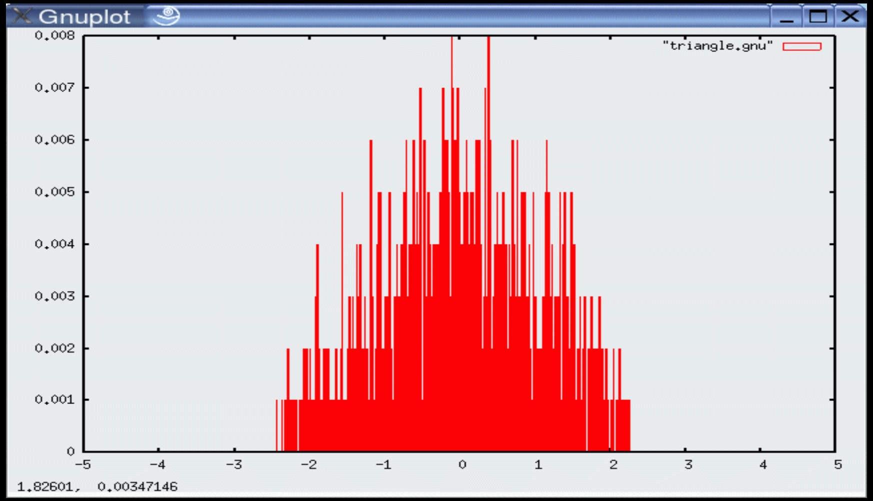
# Sampling

- A sampling algorithm is an algorithm that outputs samples y1, y2, . . . from a given distribution P
- A sampling algorithm is a procedure that allows us to select randomly a subset of units (samples) from a distribution without enumerating all the possible samples of the distribution
- Sampling algorithms are often used to approximate distributions
- A triangular distribution is given by:

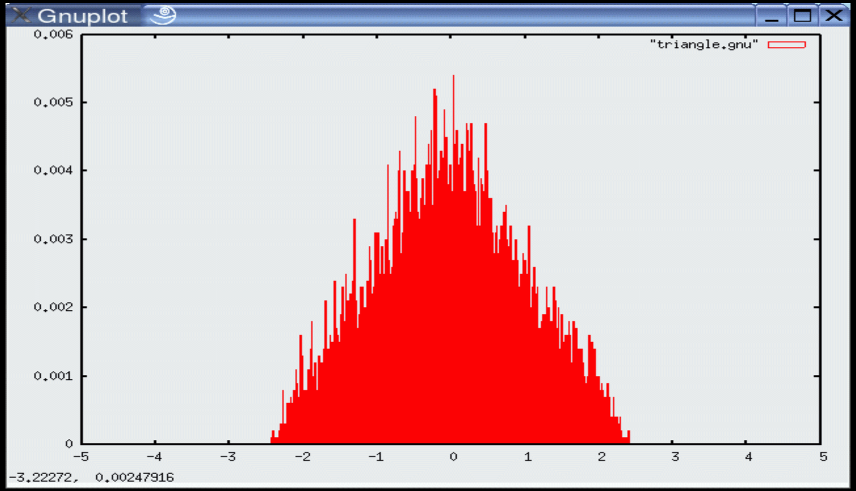
$$\varepsilon_{\sigma^2}(x) = \begin{cases} 0 \text{ if } |x| > \sqrt{6\sigma^2} \\ \dfrac{\sqrt{6\sigma^2} - |x|}{6\sigma^2} \end{cases}$$



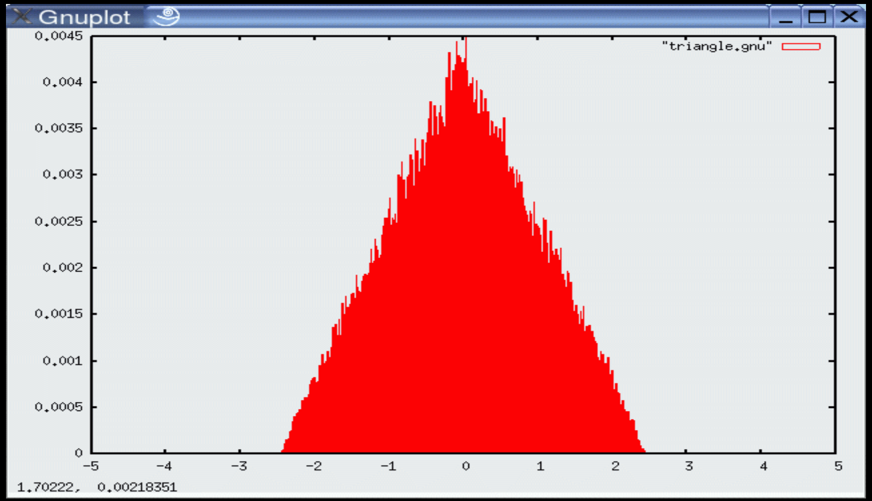
# Sampling from a Triangular Distribution



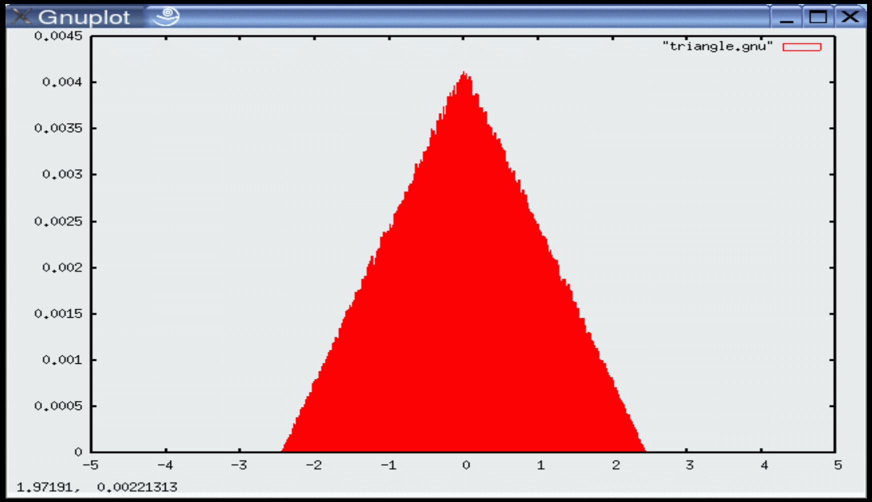

$10^3$ samples




$10^4$ samples

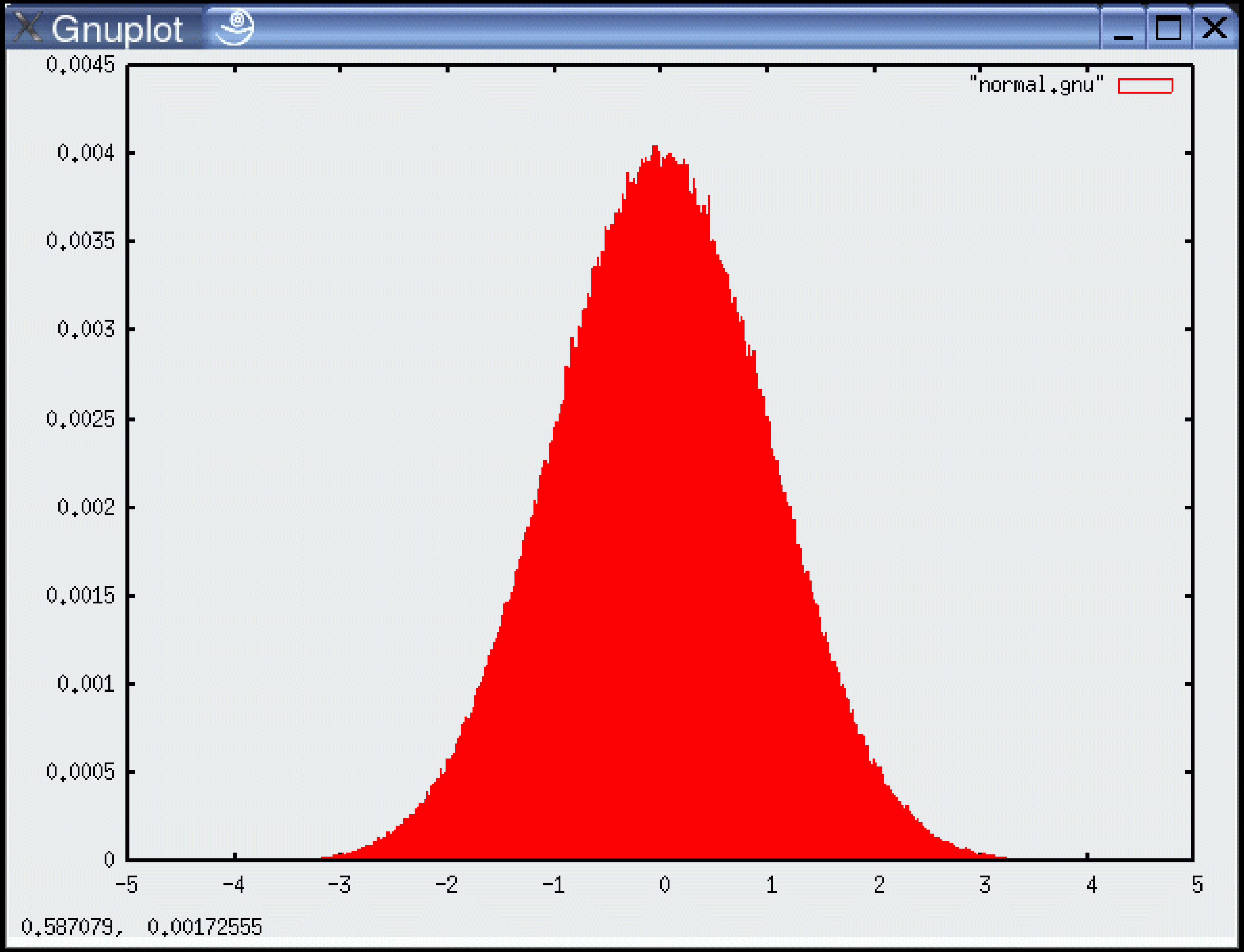



$10^5$ samples




$10^6$ samples

# Normally Distributed Samples



# Sampling from Velocity Model

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

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

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

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

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

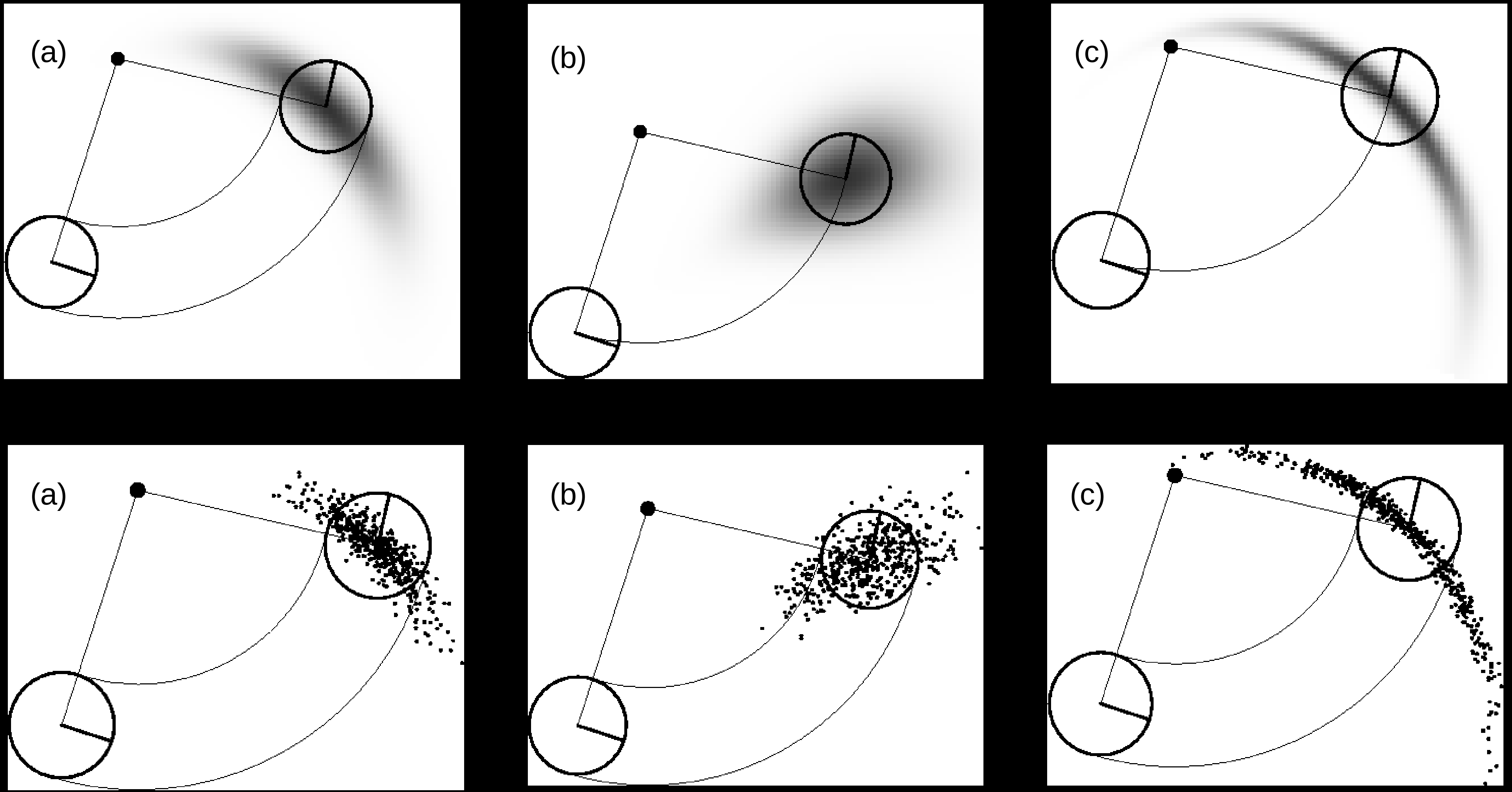
$y' = y + \frac{\hat{v}}{\hat{\omega}} \cos\theta - \frac{\hat{v}}{\hat{\omega}}\ cos(\theta + \hat{\omega}\Delta t)$

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

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

Algorithm for sampling poses $x_{t-1} = (x, y, \theta)^T$, $x_t = (x', y', \theta')^T$ and $u_t = (v_t, \omega_t)^T$. Not we are perturbing the final orientation by an additional random term γ. The variables $\alpha_1$ through $\alpha_6$ are parameters of the motion noise. sample($b^2$) generates a random sample from a zero-centered distribution with variance $b^2$

# Sampling from Velocity Model




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

# Odometry Model

# Odometry Model

- Odometry is the use of data from motion sensors to estimate change in position over time
- Uses the odometry measurements as the basis for calculating the robot's motion over time
- Odometry obtained by integrating wheel encoder information
- Hence, odometry model uses odometry information in lieu of velocity controls

# Odometry Model

- Practical experience suggests odometry, while still erroneous, is more accurate than velocity
- Though both suffer from drift and slippage, velocity model suffers from mismatch between actual motion controllers and our crude mathematical model
- However, odometry is available **after the robot has moved**
  - It cannot be used for motion planning algorithms 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

- Uses the relative motion information as measured by the robot's internal odometry
- From $(t-1, t]$, the robot advances from $x_t$ to $x_{t-1}$
- The odometry reports back to us a related advance from

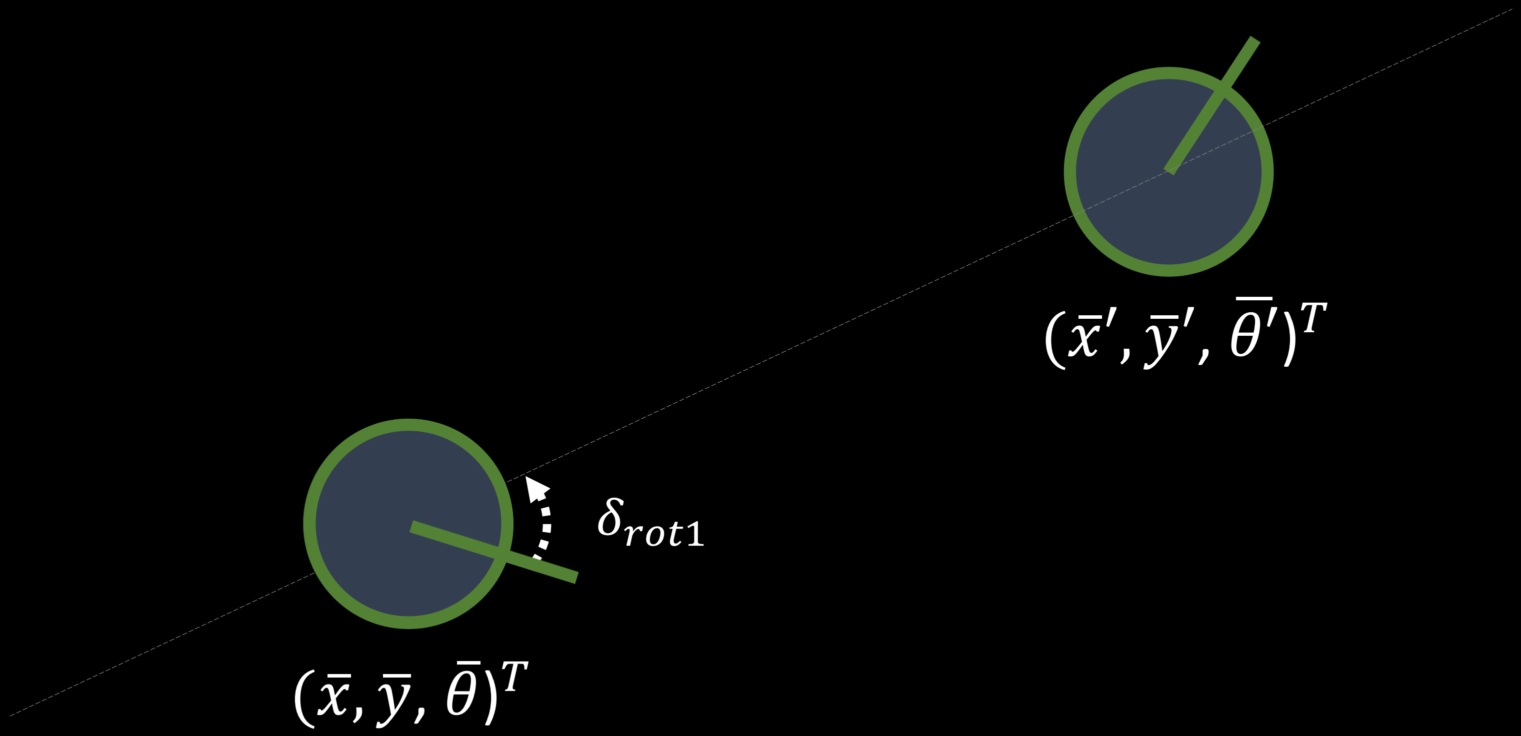
$$\overline{x_{t-1}} = (\bar{x}, \bar{y}, \bar{\theta})^T \quad \text{to} \quad \bar{x_t} = (\bar{x'}, \bar{y}', \bar{\theta'})^T$$

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

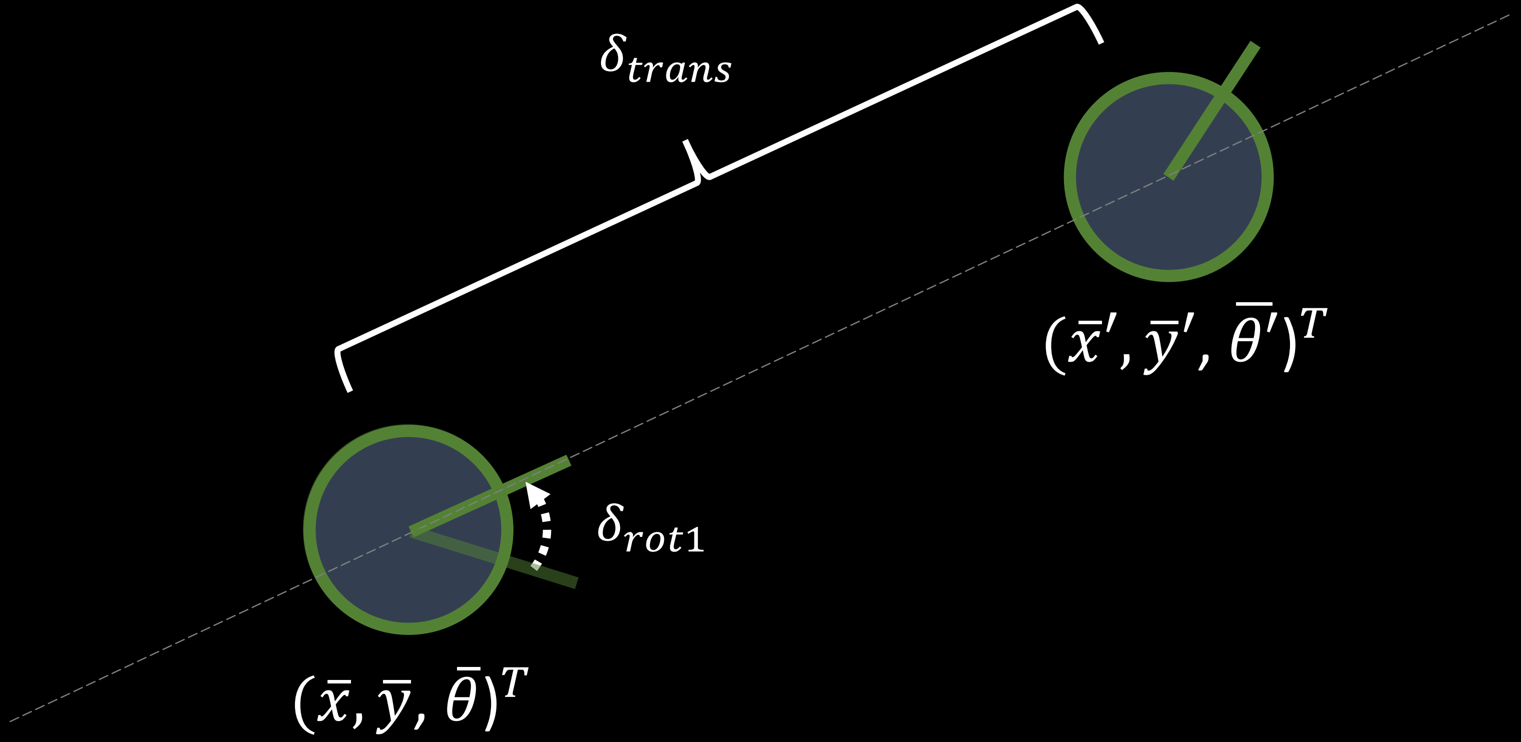
- **Key idea:** In state estimation, the relative difference between $\overline{x_{t-1}}$ and $\bar{x_t}$ is a good estimator for the difference of the true poses $x_{t-1}$ and $x_t$
- Motion information is given by:

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

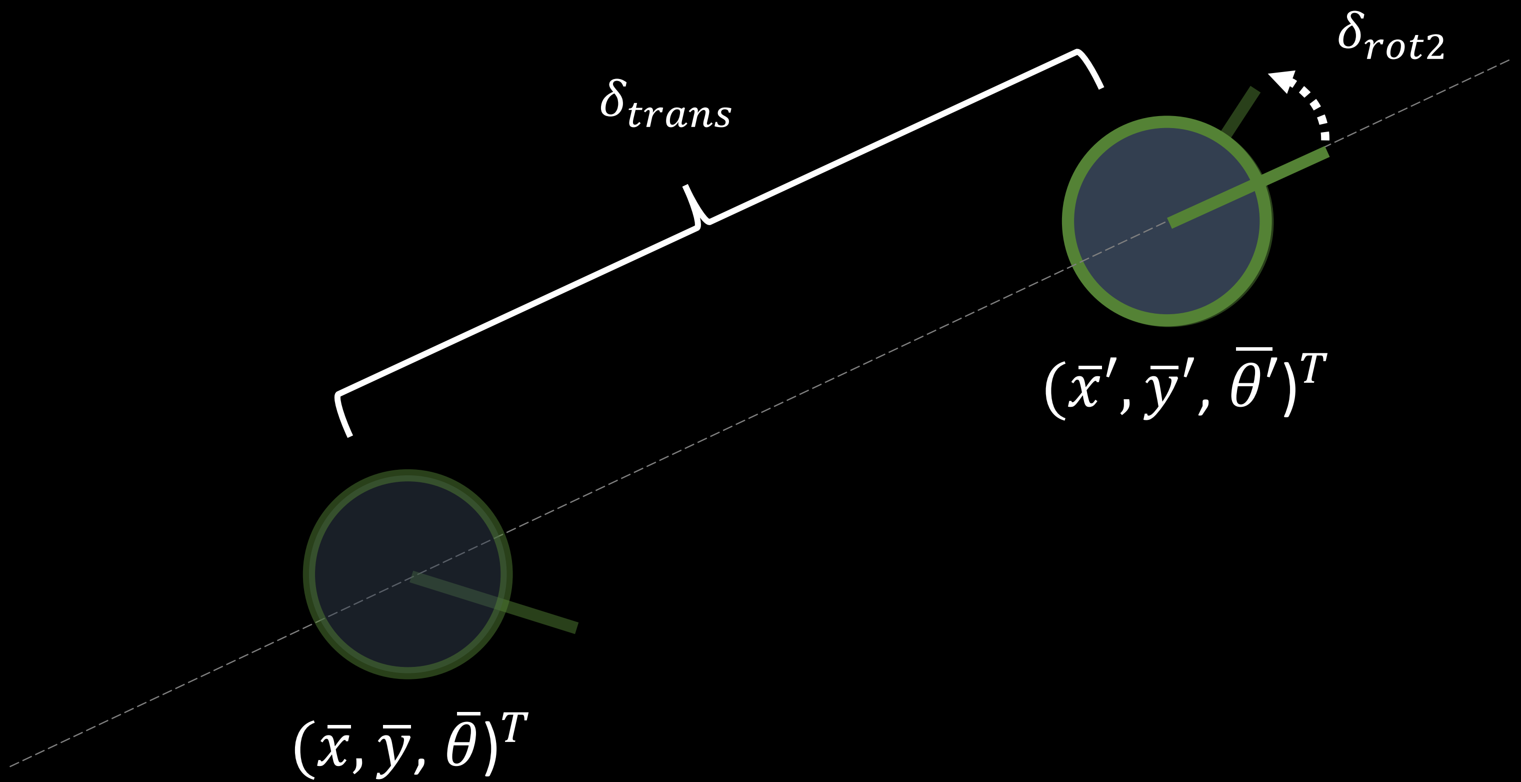
# Odometry Model Parameters



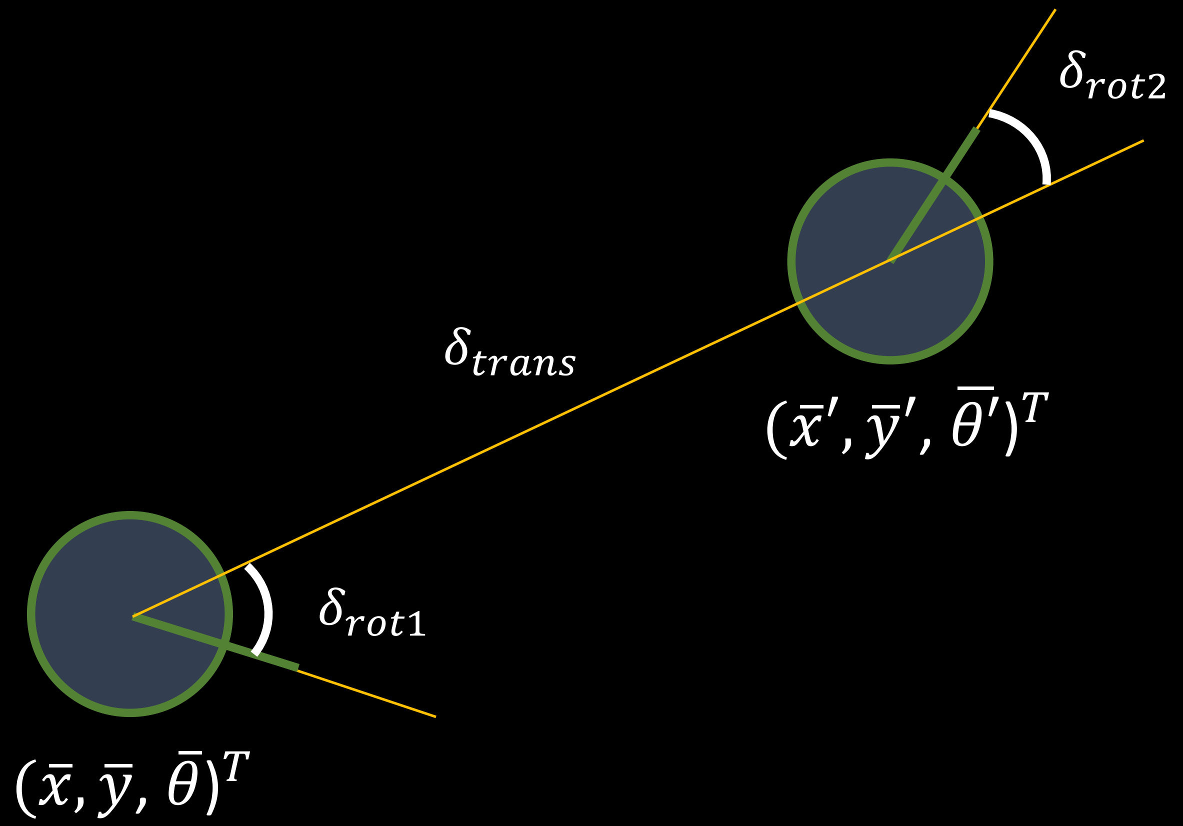
# Odometry Model Parameters



# Odometry Model Parameters



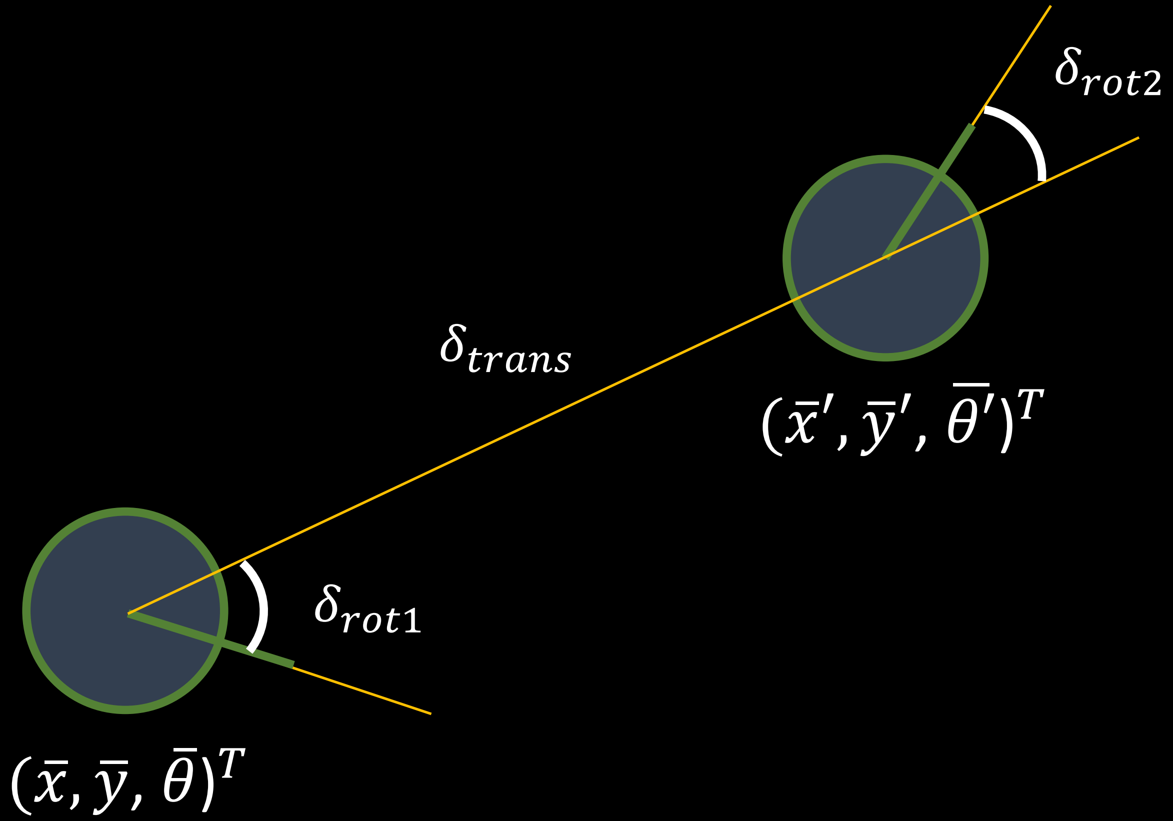
# Odometry Model Parameters



# Odometry Model Parameters

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

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

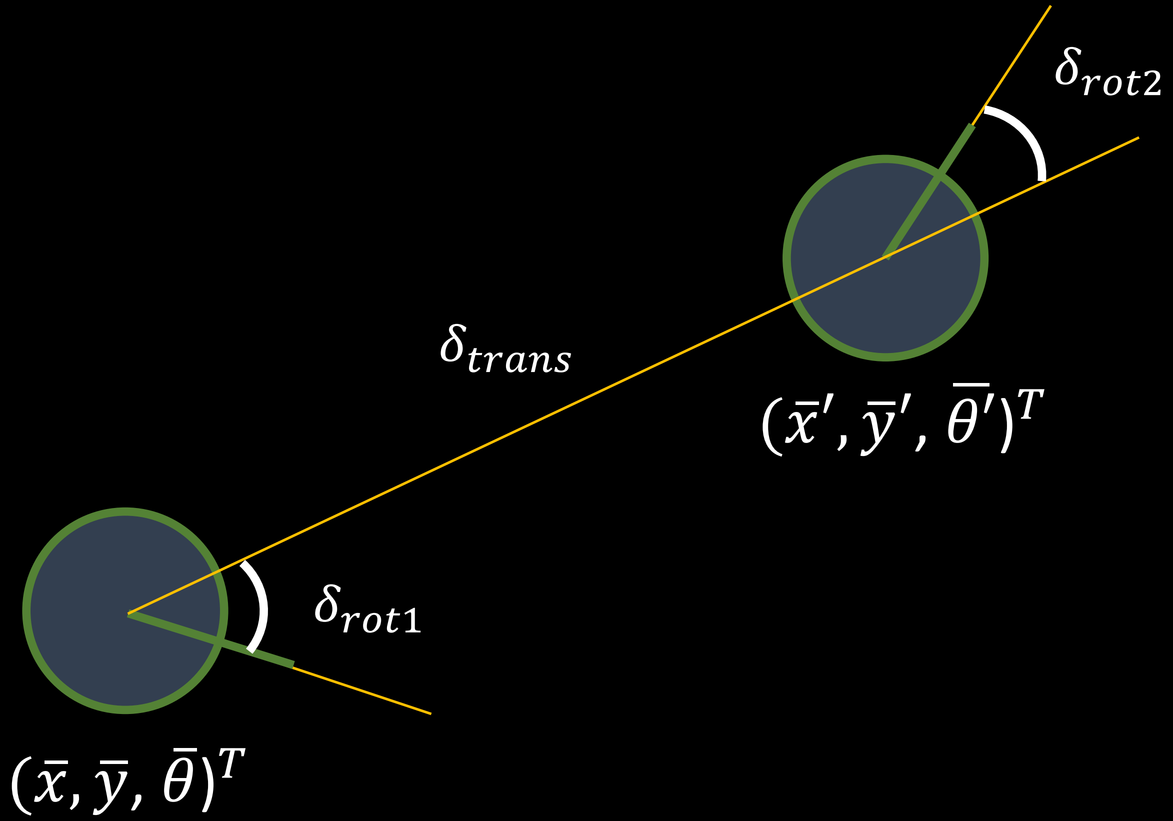


# Odometry Model Parameters

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

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

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



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

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

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

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

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

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

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

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

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

11. return $p_1 . p_2 . p_3$

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

Calculate the relative motion parameters from odometry readings

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

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

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

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

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

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

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

11. return $p_1 . p_2 . p_3$

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 $bel(x_t)$

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

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. $\mathtt{return}\ x_t = (x', y', \theta')^T$

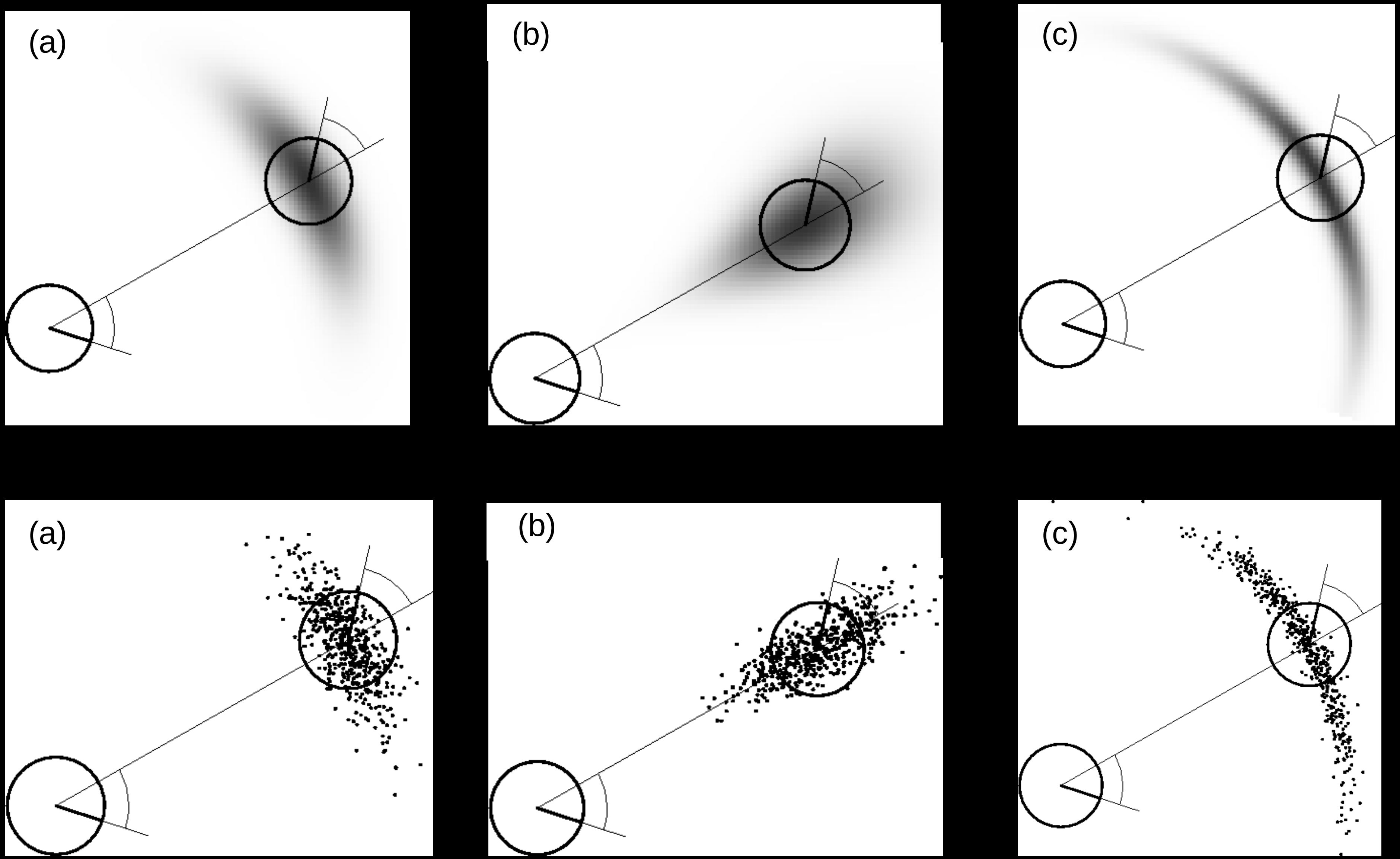
Calculate the relative motion parameters from odometry readings

Add noise to calculated motion parameters

Calculate the sample state

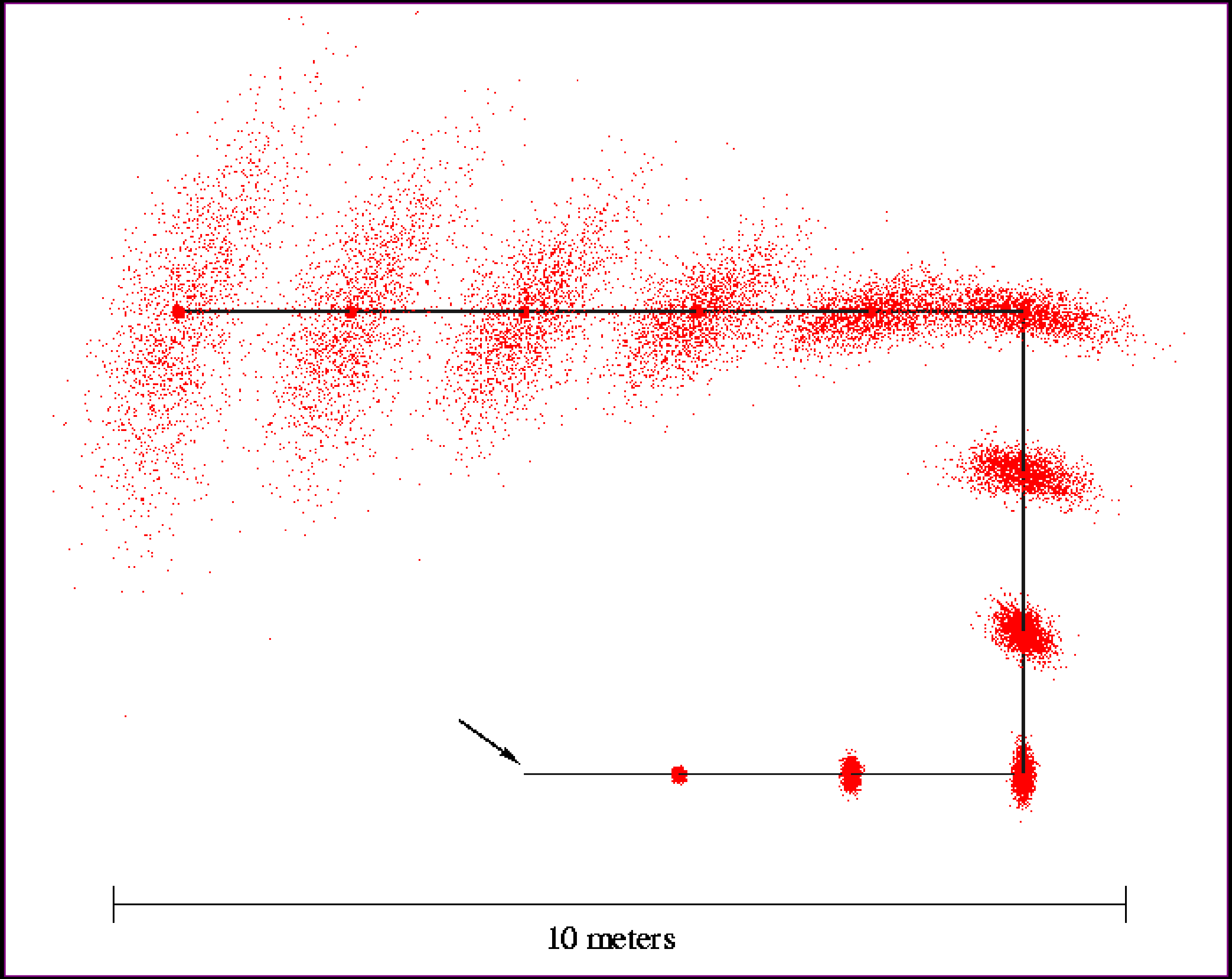
Algorithm for sampling from $p(x_t \mid u_t, x_{t-1})$ based on odometry information

# 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




Sampling approximation of the position belief for a non-sensing robot. Solid lines displays the robot's actual motion and the samples represent the robot's belief at different points in time

# 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 $p(x_t \mid u_t\ , x_{t-1})$
- Typically the calculations are done in fixed time intervals $\Delta t$
- 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