Motion Models (cont)

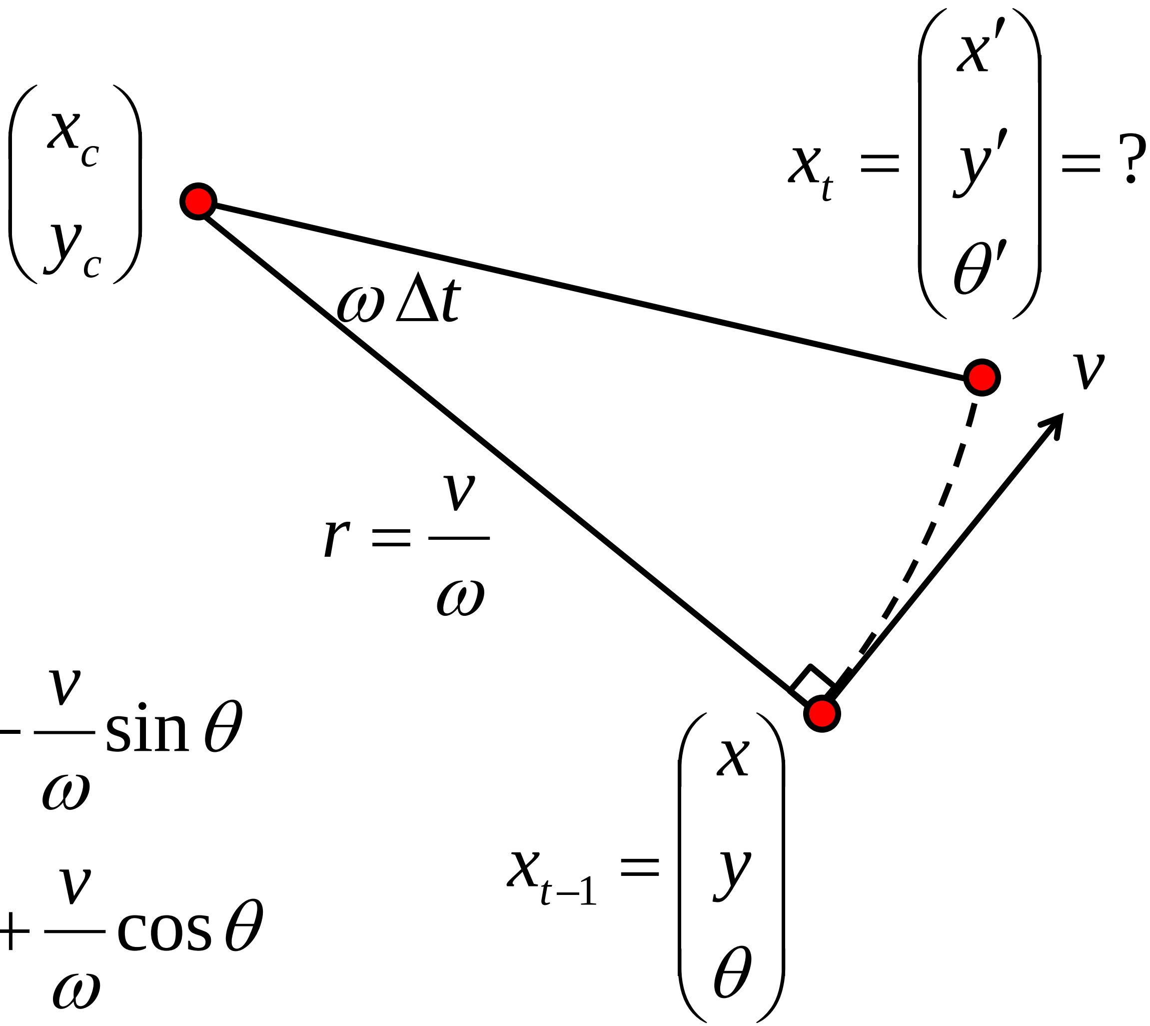
Sampling from the Velocity Motion Model

- suppose that a robot has a map of its environment and it needs to find its pose in the environment
  - this is the robot localization problem (Chapter 7)
  - several variants of the problem
    - the robot knows where it is initially
    - the robot does not know where it is initially
    - kidnapped robot: at any time, the robot can be teleported to another location in the environment
- a popular solution to the localization problem is the particle filter (Chapter 4)
  - uses simulation to sample the state density $p(x_t \mid u_t, x_{t-1})$

Sampling from the Velocity Motion Model

- sampling the conditional density is easier than computing the density because we only require the forward kinematics model
  - given the control u_t and the previous pose $x_{t\text{-}1}$ find the new pose x_t

Sampling from the Velocity Motion Model



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

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

Eqs 5.7, 5.8

Sampling from the Velocity Motion Model

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

$$= \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} -\frac{v}{\omega}\sin\theta + \frac{v}{\omega}\sin(\theta + \omega\Delta t) \\ \frac{v}{\omega}\cos\theta - \frac{v}{\omega}\cos(\theta + \omega\Delta t) \\ \omega\Delta t \end{pmatrix} \quad \text{Eqs 5.9}$$

*we already derived this for the differential drive!

Sampling from the Velocity Motion Model

- as with the original motion model, we will assume that given noisy velocities the robot can also make a small rotation in place to determine the final orientation of the robot

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

Sampling from the Velocity Motion Model

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

$\hat{v} = v + \textbf{sample}(\alpha_1\, v^2 + \alpha_2\, \omega^2)$

$\hat{\omega} = \omega + \textbf{sample}(\alpha_3\, v^2 + \alpha_4\, \omega^2)$

$\hat{\gamma} = \textbf{sample}(\alpha_5\, v^2 + \alpha_6\, \omega^2)$

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

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

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

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

Sampling from the Velocity Motion Model

- the function $\mathrm{sample}(b^2)$ generates a random sample from a zero-mean distribution with variance b^2
  - Table 5.4 has two algorithms you could use
  - Matlab is able to generate random numbers from many different distributions
    - help randn
    - help stats

How to Sample from Normal or Triangular Distributions?

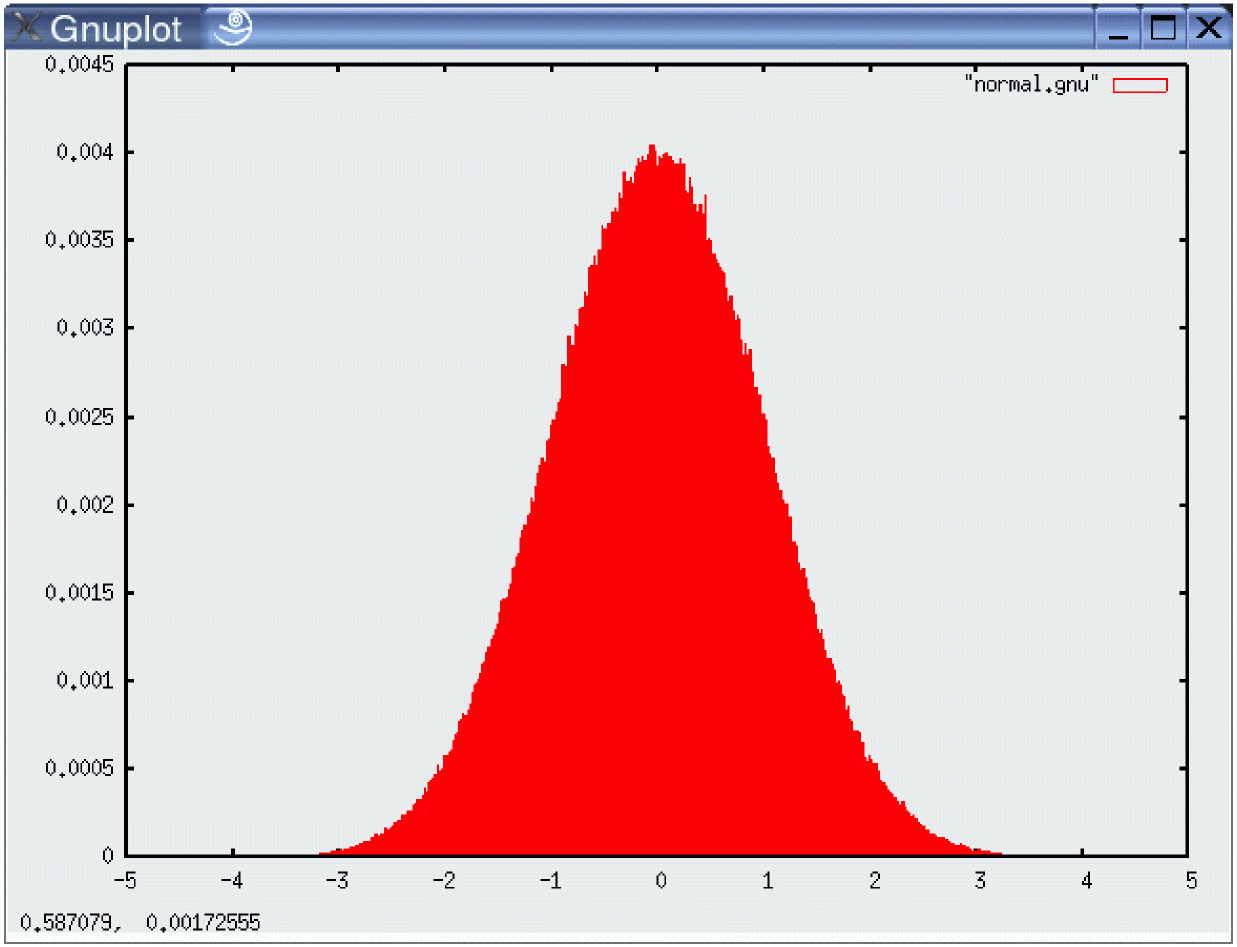
- Sampling from a normal distribution

1. Algorithm **sample_normal_distribution**(b):
2. return $$\frac{1}{2}\sum_{i=1}^{12} rand(-b, b)$$

- Sampling from a triangular distribution

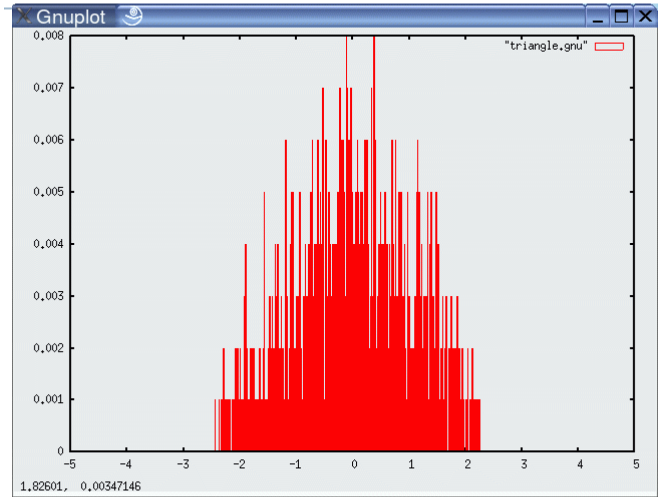
1. Algorithm **sample_triangular_distribution**(b):
2. return $$\frac{\sqrt{6}}{2}\left[\mathbf{rand}(-b, b) + \mathbf{rand}(-b, b)\right]$$

Normally Distributed Samples

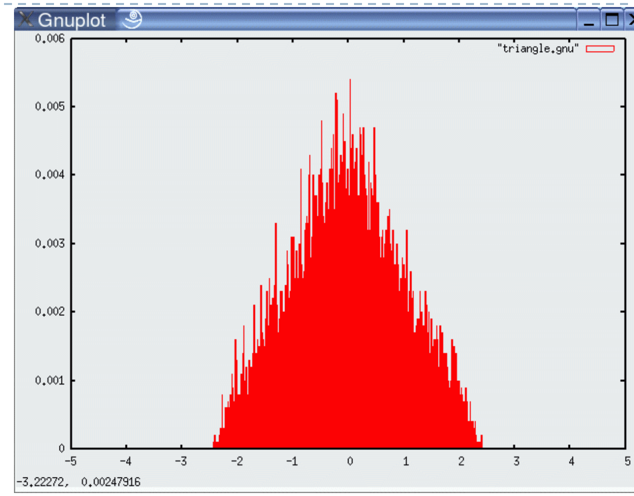


10^6 samples

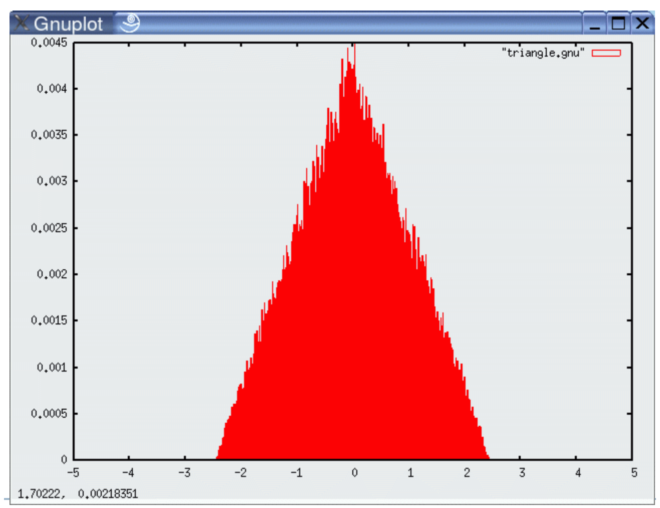
For Triangular Distribution



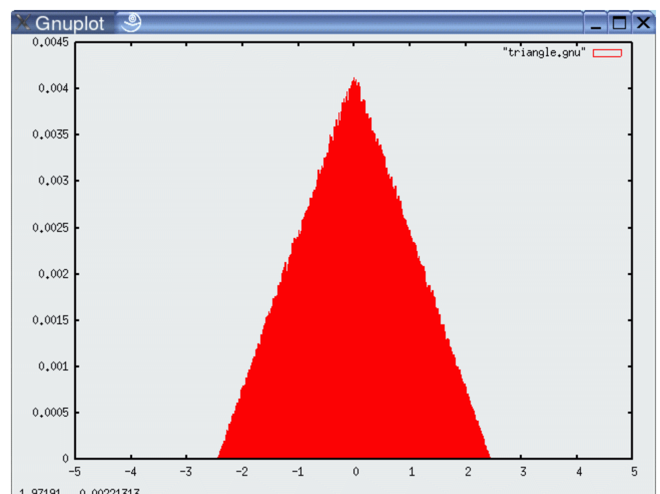
10^3 samples



10^4 samples



10^5 samples



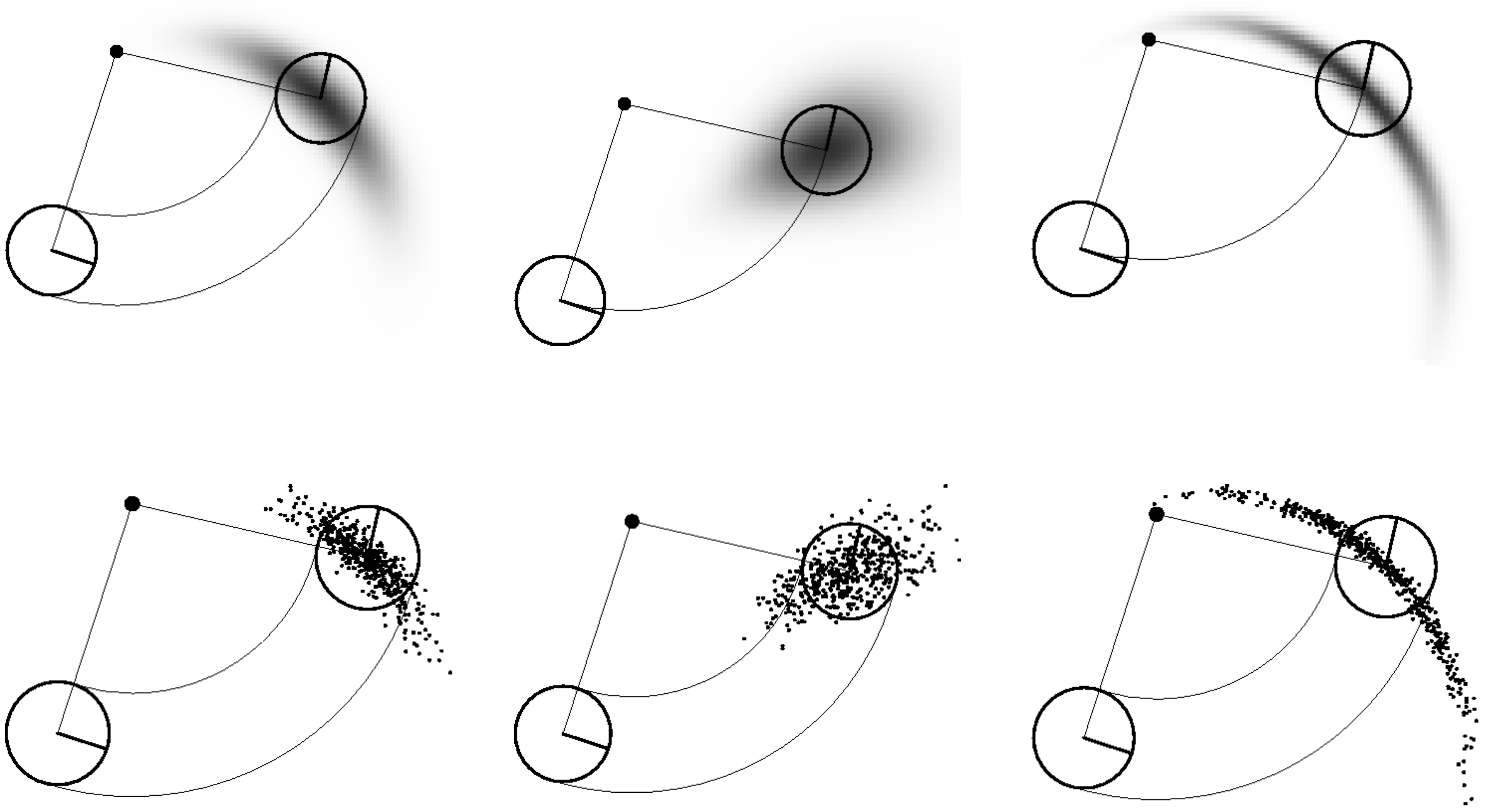
10^6 samples

Rejection Sampling

- Sampling from arbitrary distributions

1. Algorithm **sample_distribution**(*f*,*b*):
2. repeat
3. $x = \mathbf{rand}(-b, b)$
4. $y = \mathbf{rand}(0, \max\{f(x) \mid x \in (-b, b)\})$
5. until ($y \leq f(x)$)
6. return x

Examples

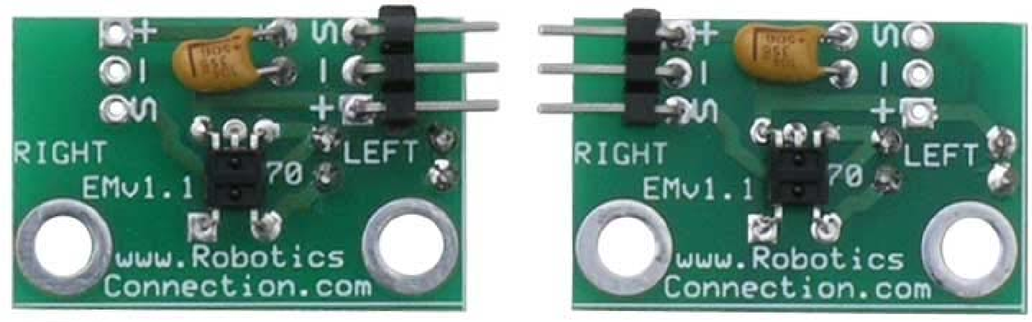


Odometry Motion Model

- many robots make use of odometry rather than velocity
- odometry uses a sensor or sensors to measure motion to estimate changes in position over time
- typically more accurate than velocity motion model, but measurements are available only after the motion has been completed
- technically a measurement rather than a control
  - but usually treated as control to simplify the modeling
- odometry allows a robot to estimate its pose
  - but no fixed mapping from odometer coordinates and world coordinates
- in wheeled robots the sensor is often a rotary encoder

Example Wheel Encoders

These modules require +5V and GND to power them, and provide a 0 to 5V output. They provide +5V output when they "see" white, and a 0V output when they "see" black.



These disks are manufactured out of high quality laminated color plastic to offer a very crisp black to white transition. This enables a wheel encoder sensor to easily see the transitions.

Source: http://www.active-robots.com/

Odometry Model

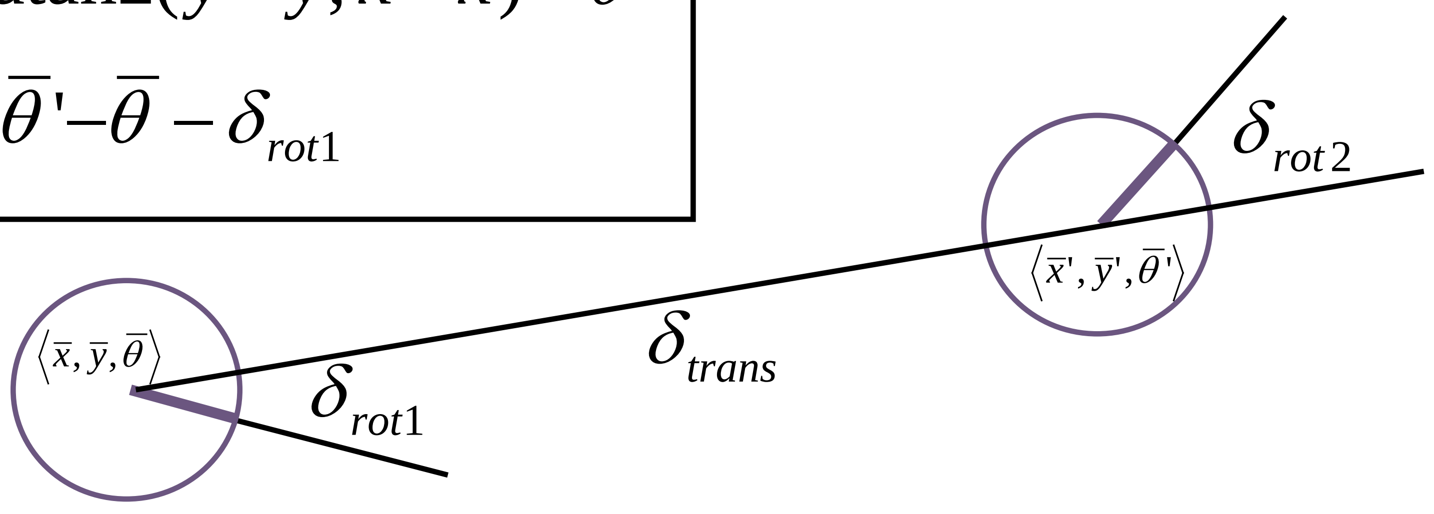
bar indicates odometer coordinates

- Robot moves from $\langle \bar{x}, \bar{y}, \bar{\theta} \rangle$ to $\langle \bar{x}', \bar{y}', \bar{\theta}' \rangle$.
- Odometry information $u = \langle \delta_{rot1}, \delta_{rot2}, \delta_{trans} \rangle$

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

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

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



Noise Model for Odometry

- The measured motion is given by the true motion corrupted with noise.

$$\hat{\delta}_{rot1} = \delta_{rot1} - \varepsilon_{\alpha_1 \delta^2_{rot1} + \alpha_2 \delta^2_{trans}}$$

$$\hat{\delta}_{trans} = \delta_{trans} - \varepsilon_{\alpha_3 \delta^2_{trans} + \alpha_4 \delta^2_{rot1} + \delta^2_{rot2}}$$

$$\hat{\delta}_{rot2} = \delta_{rot2} - \varepsilon_{\alpha_1 \delta^2_{rot2} + \alpha_2 \delta^2_{trans}}$$

Sample Odometry Motion Model

1. Algorithm **sample_motion_model**(u, x):

   $u = \langle \delta_{rot1}, \delta_{rot2}, \delta_{trans} \rangle, x = \langle x, y, \theta \rangle$

1. $\hat{\delta}_{rot1} = \delta_{rot1} + \text{sample}(\alpha_1\ \delta_{rot1}^2 + \alpha_2\ \delta_{trans}^2)$
2. $\hat{\delta}_{trans} = \delta_{trans} + \text{sample}(\alpha_3\ \delta_{trans}^2 + \alpha_4\ (\delta_{rot1}^2 + \delta_{trans}^2))$
3. $\hat{\delta}_{rot2} = \delta_{rot2} + \text{sample}(\alpha_1\ \delta_{rot2}^2 + \alpha_2\ \delta_{trans}^2)$

4. $x' = x + \hat{\delta}_{trans} \cos(\theta + \hat{\delta}_{rot1})$
5. $y' = y + \hat{\delta}_{trans} \sin(\theta + \hat{\delta}_{rot1})$
6. $\theta' = \theta + \hat{\delta}_{rot1} + \hat{\delta}_{rot2}$

7. Return $\langle x', y', \theta' \rangle$

sample_normal_distribution

Sampling from Our Motion Model