Motion Models (cont)

Odometry Model

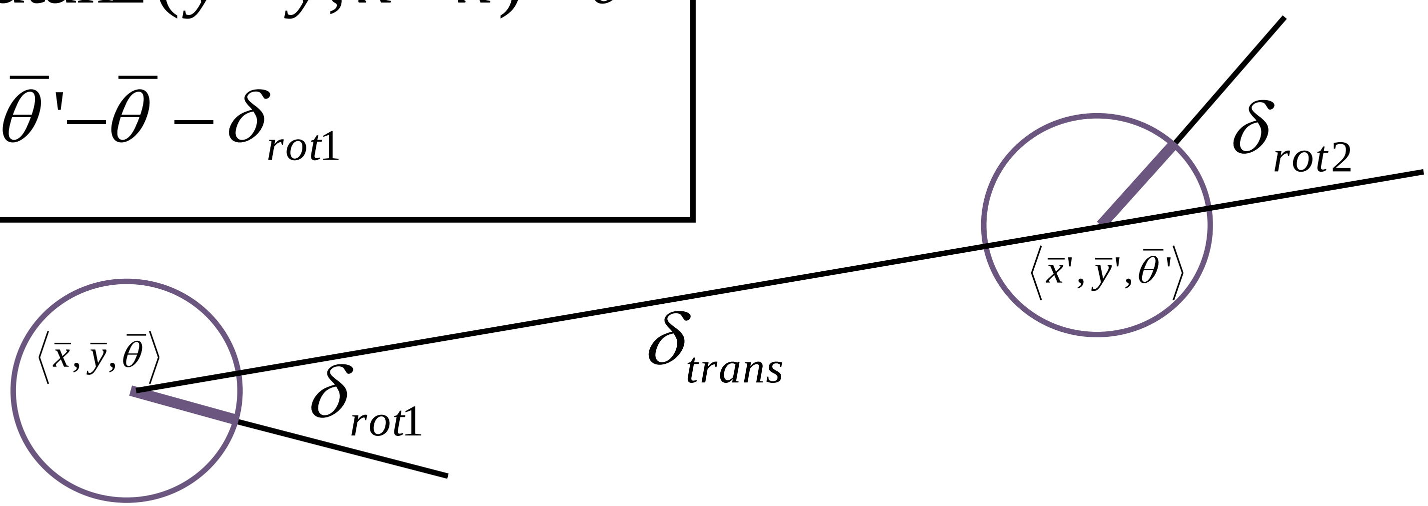
bar indicates odometer coordinates

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

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

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

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



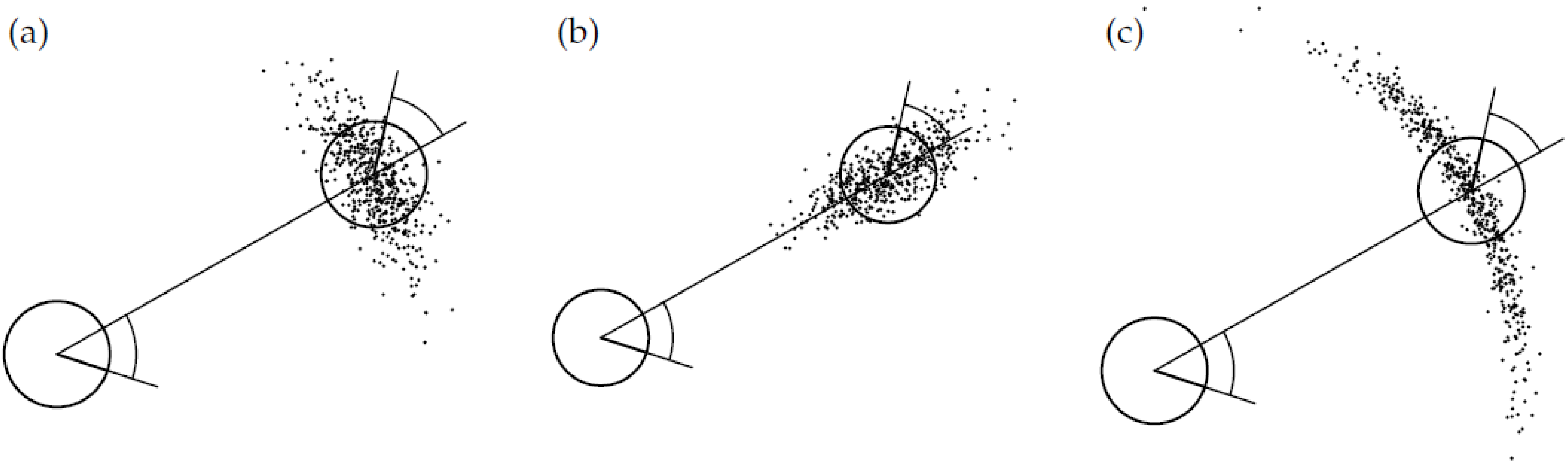
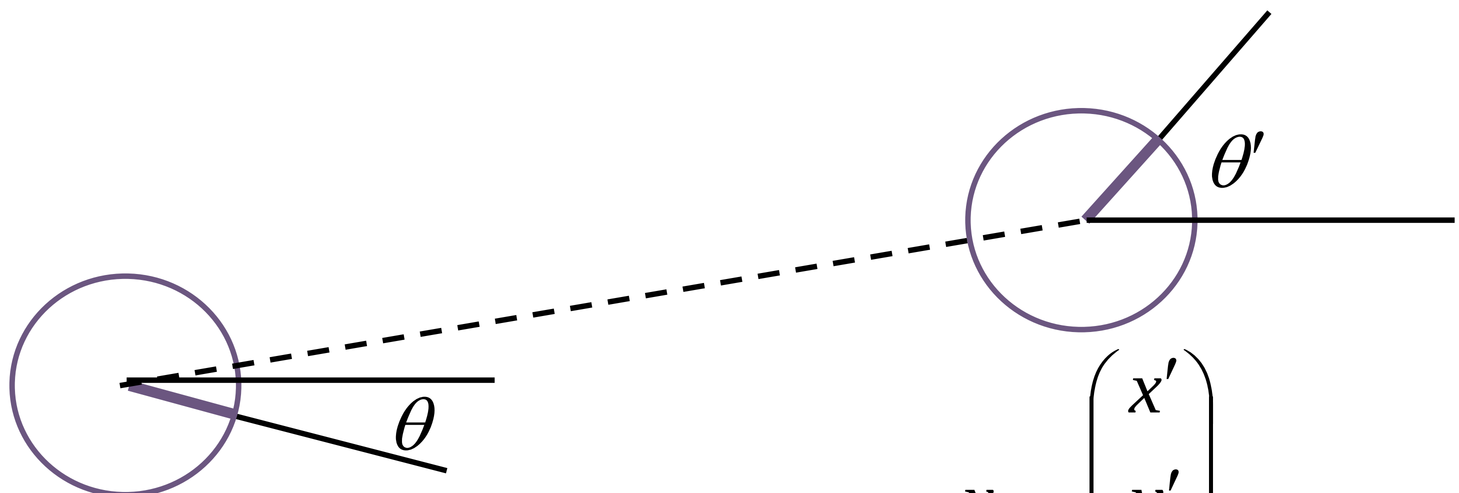


Figure 5.9 Sampling from the odometry motion model, using the same parameters as in Figure 5.8. Each diagram shows 500 samples.

Odometry Motion Model

- the key to computing $p(x_t \mid u_t, x_{t-1})$ for the odometry motion model is to remember that the robot has an internal estimate of its pose



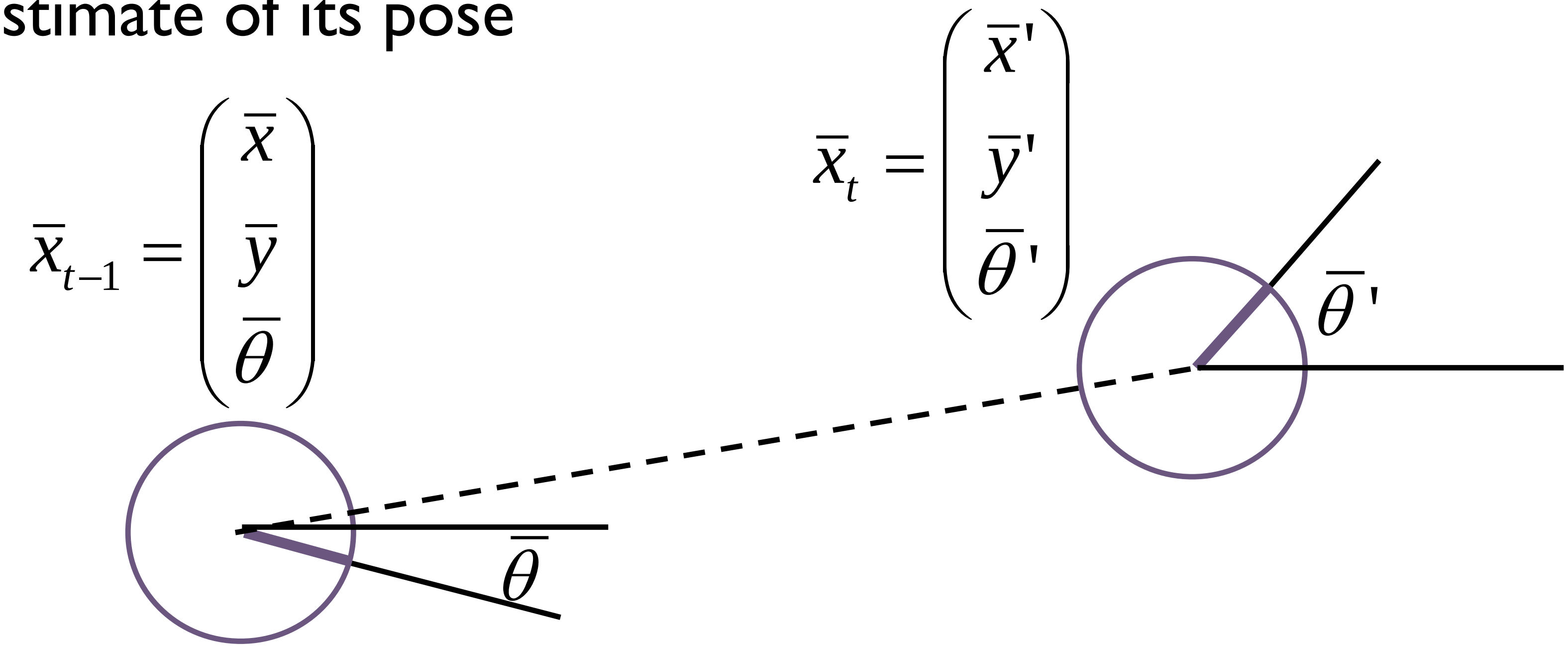
$$x_t = \begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix}$$

$$x_{t-1} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$$

true poses

Odometry Motion Model

- the key to computing $p(x_t \mid u_t, x_{t-1})$ for the odometry motion model is to remember that the robot has an internal estimate of its pose



robot's internal poses

Odometry Motion Model

- the control vector is made up of the three motions made by the robot

$$u_t = \begin{pmatrix} \delta_{trans} \\ \delta_{rot1} \\ \delta_{rot2} \end{pmatrix}$$

- use the robot's internal pose estimates to compute the δ

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

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

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

Odometry Motion Model

- use the true poses to compute the δ

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

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

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

- as with the velocity motion model, we have to solve the inverse kinematics problem here

Odometry Motion Model

- recall the noise model

$$\hat{\delta}_{trans} - \delta_{trans} = \varepsilon_{\alpha_3 \hat{\delta}^2_{trans} + \alpha_4 (\hat{\delta}^2_{rot1} + \hat{\delta}^2_{rot2})}$$
$$\hat{\delta}_{rot1} - \delta_{rot1} = \varepsilon_{\alpha_1 \hat{\delta}^2_{rot1} + \alpha_2 \hat{\delta}^2_{trans}}$$
$$\hat{\delta}_{rot2} - \delta_{rot2} = \varepsilon_{\alpha_1 \hat{\delta}^2_{rot2} + \alpha_2 \hat{\delta}^2_{trans}}$$

which makes it easy to compute the probabilities of observing the differences in the δ

$$p_1 = \text{prob}(\hat{\delta}_{trans} - \delta_{trans}, \alpha_3 \hat{\delta}^2_{trans} + \alpha_4 (\hat{\delta}^2_{rot1} + \hat{\delta}^2_{rot2}))$$
$$p_2 = \text{prob}(\hat{\delta}_{rot1} - \delta_{rot1}, \alpha_1 \hat{\delta}^2_{rot1} + \alpha_2 \hat{\delta}^2_{trans})$$
$$p_3 = \text{prob}(\hat{\delta}_{rot2} - \delta_{rot2}, \alpha_1 \hat{\delta}^2_{rot2} + \alpha_2 \hat{\delta}^2_{trans})$$

Calculating the Posterior Given x, x', and u

1. Algorithm **motion_model_odometry(x,x',u)**
2. $\delta_{trans} = \sqrt{(\bar{x}' - \bar{x})^2 + (\bar{y}' - \bar{y})^2}$
3. $\delta_{rot1} = \mathrm{atan2}(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta}$
4. $\delta_{rot2} = \bar{\theta}' - \bar{\theta} - \delta_{rot1}$
5. $\hat{\delta}_{trans} = \sqrt{(x' - x)^2 + (y' - y)^2}$
6. $\hat{\delta}_{rot1} = \mathrm{atan2}(y' - y, x' - x) - \bar{\theta}$
7. $\hat{\delta}_{rot2} = \theta' - \theta - \hat{\delta}_{rot1}$
8. $p_1 = \mathrm{prob}(\delta_{\mathrm{rot1}} - \hat{\delta}_{\mathrm{rot1}}, \alpha_1 \hat{\delta}^2_{\mathrm{rot1}} + \alpha_2 \hat{\delta}^2_{\mathrm{trans}})$
9. $p_2 = \mathrm{prob}(\delta_{\mathrm{trans}} - \hat{\delta}_{\mathrm{trans}}, \alpha_3 \hat{\delta}^2_{\mathrm{trans}} + \alpha_4 (\hat{\delta}^2_{\mathrm{rot1}} + \hat{\delta}^2_{\mathrm{rot2}}))$
10. $p_3 = \mathrm{prob}(\delta_{\mathrm{rot2}} - \hat{\delta}_{\mathrm{rot2}}, \alpha_1 \hat{\delta}^2_{\mathrm{rot2}} + \alpha_2 \hat{\delta}^2_{\mathrm{trans}})$
11. return $p_1 \cdot p_2 \cdot p_3$

odometry values (u)

values of interest (x,x')

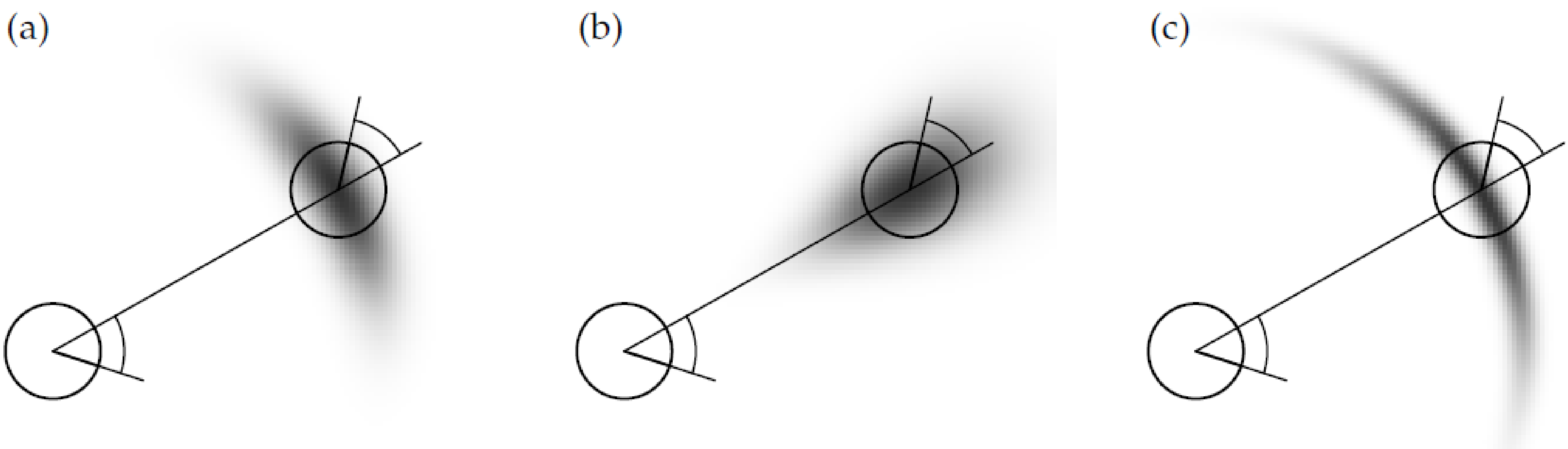
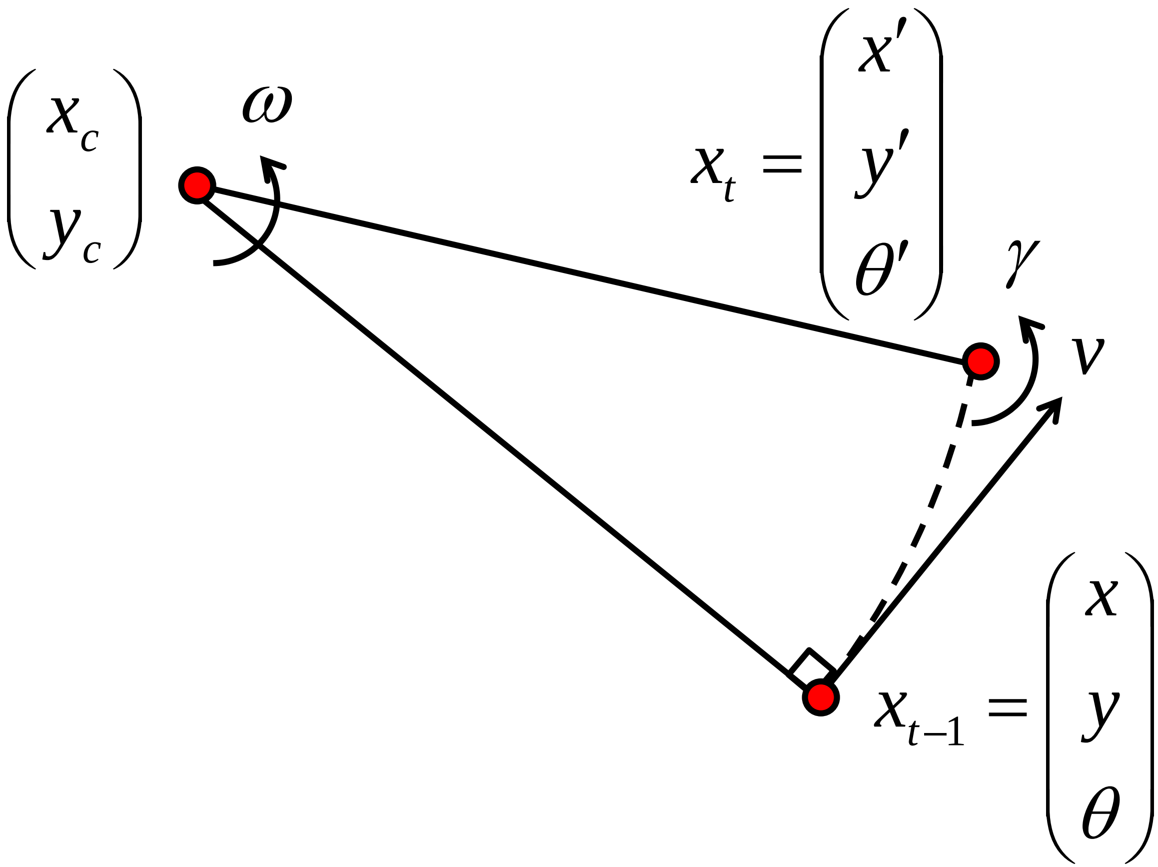


Figure 5.8 The odometry motion model, for different noise parameter settings.

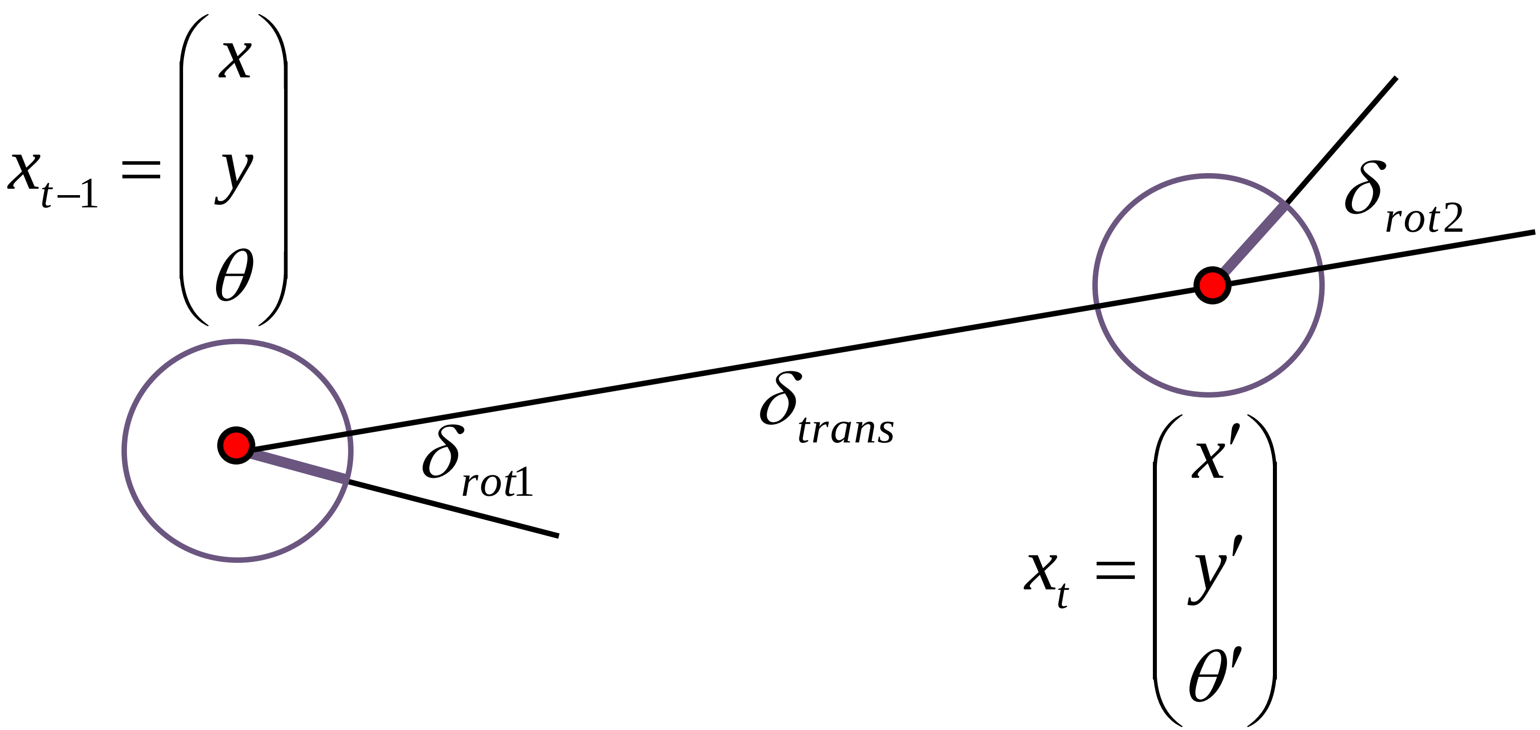
Recap

- velocity motion model
  - control variables were linear velocity, angular velocity about ICC, and final angular velocity about robot center



Recap

- odometric motion model
  - control variables were derived from odometry
    - initial rotation, translation, final rotation



Recap

- for both models we assumed the control inputs u_t were noisy
- the noise models were assumed to be zero-mean additive with a specified variance

$$\begin{pmatrix} \hat{v} \\ \hat{\omega} \end{pmatrix} = \begin{pmatrix} v \\ \omega \end{pmatrix} + \begin{pmatrix} v_{\text{noise}} \\ \omega_{\text{noise}} \end{pmatrix}$$

actual velocity	commanded velocity	noise

$$\mathrm{var}(v_{\text{noise}}) = \alpha_1 v^2 + \alpha_2 \omega^2$$
$$\mathrm{var}(\omega_{\text{noise}}) = \alpha_3 v^2 + \alpha_4 \omega^2$$

Recap

- for both models we assumed the control inputs u_t were noisy
- the noise models were assumed to be zero-mean additive with a specified variance

$$\begin{pmatrix} \hat{\delta}_{trans} \\ \hat{\delta}_{rot1} \\ \hat{\delta}_{rot2} \end{pmatrix} = \begin{pmatrix} \delta_{trans} \\ \delta_{rot1} \\ \delta_{rot2} \end{pmatrix} + \begin{pmatrix} \delta_{trans,noise} \\ \delta_{rot1,noise} \\ \delta_{rot2,noise} \end{pmatrix}$$

actual motion | commanded motion | noise

$$\operatorname{var}(\delta_{trans,noise}) = \alpha_3\, \hat{\delta}^2_{trans} + \alpha_4\, (\hat{\delta}^2_{rot1} + \hat{\delta}^2_{rot2})$$

$$\operatorname{var}(\delta_{rot1,noise}) = \alpha_1\, \hat{\delta}^2_{rot1} + \alpha_2\, \hat{\delta}^2_{trans}$$

$$\operatorname{var}(\delta_{rot2,noise}) = \alpha_1\, \hat{\delta}^2_{rot2} + \alpha_2\, \hat{\delta}^2_{trans}$$

Recap

- for both models we studied how to derive $p(x_t \mid u_t, x_{t-1})$
  - given
    - x_{t-1} current pose
    - u_t control input
    - x_t new pose

  find the probability density that the new pose is generated by the current pose and control input
- required inverting the motion model to compare the *actual* with the *commanded* control parameters

Recap

- for both models we studied how to sample from $p(x_t \mid u_t, x_{t-1})$
  - given
    - $x_{t\text{-}1}$ current pose
    - u_t control input

  generate a random new pose x_t consistent with the motion model
- sampling from $p(x_t \mid u_t, x_{t-1})$ is often easier than calculating $p(x_t \mid u_t, x_{t-1})$ directly because only the forward kinematics are required