COLUMBIA UNIVERSITY EEME E6911 FALL ‘25

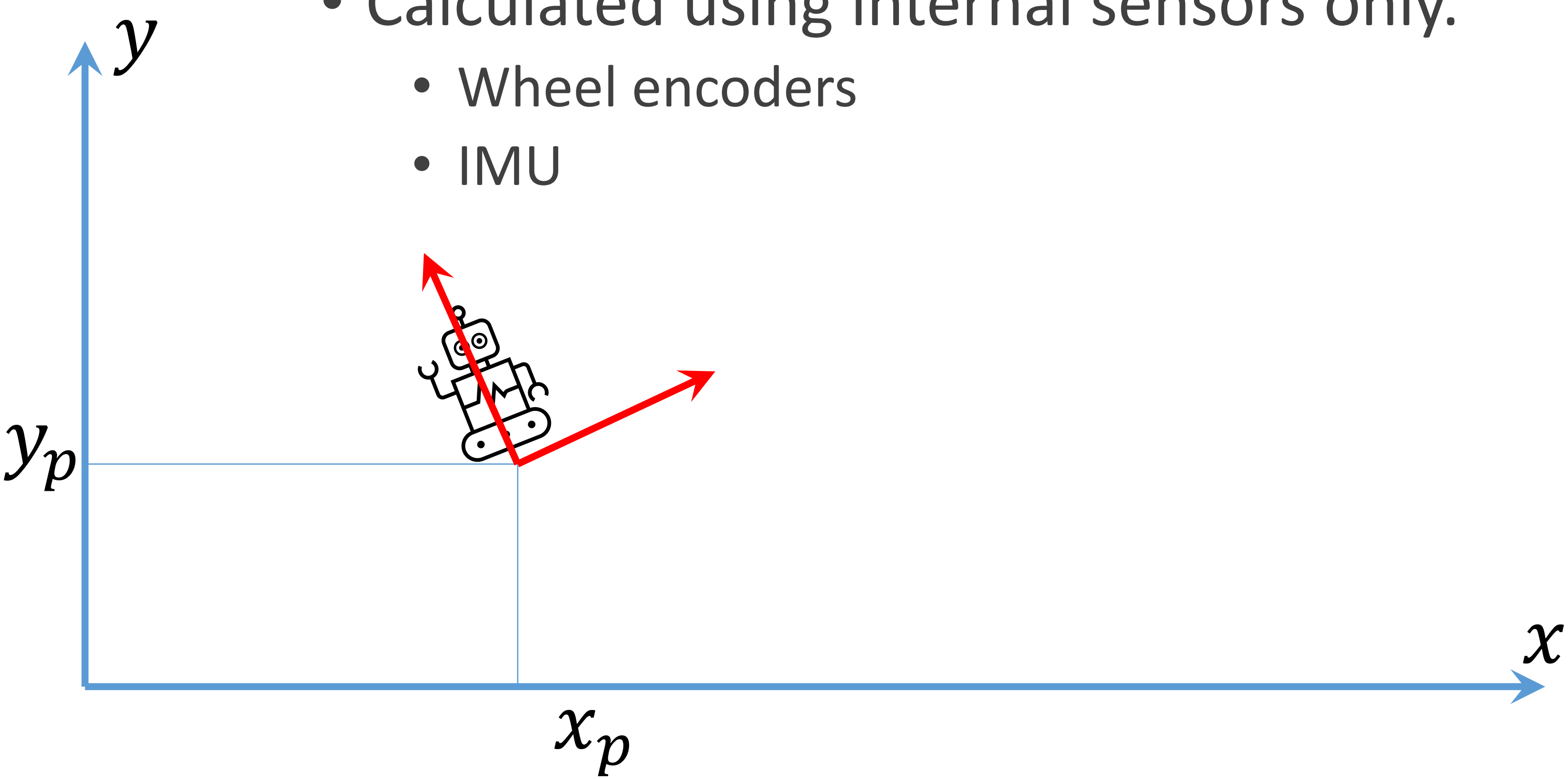
TOPICS IN CONTROL : PROBABILISTIC ROBOTICS

ODOMETRY

Instructor: Ilija Hadzic

Odometry Pose

- Relative to an arbitrary frame of reference.
  - Origin set at boot-up time.
- Calculated using internal sensors only.
  - Wheel encoders
  - IMU



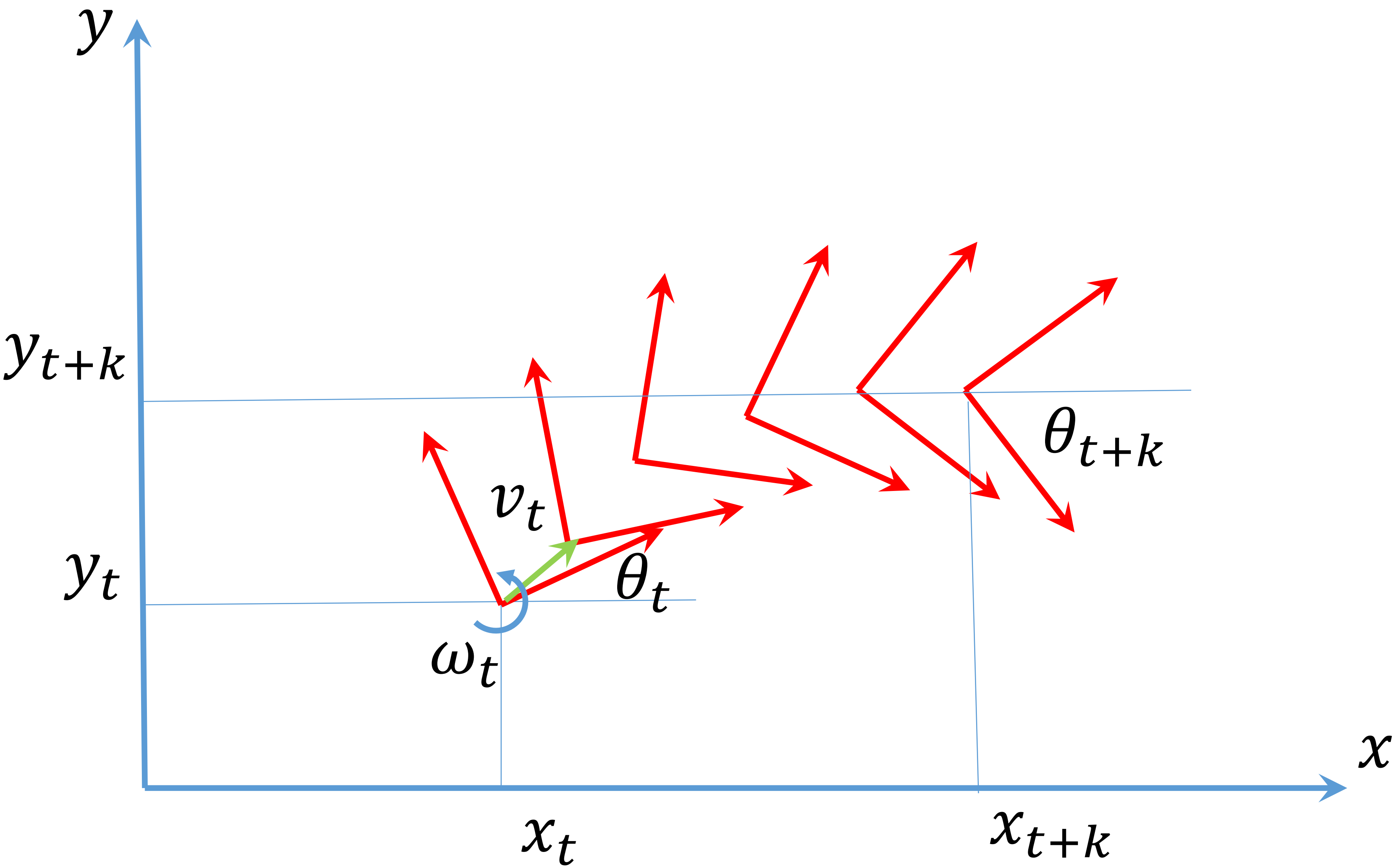
Odometry Properties

- Simple to calculate, always available
- Useful for motion prediction
- Continuous (and therefore differentiable).
- Drifts over time.
- Covariance grows without bound (therefore useless)
- We can still track covariance over a finite time interval
- Analogy:
    - How far can you walk with your eyes closed?
- Origin ambiguity

Constructing Odometry

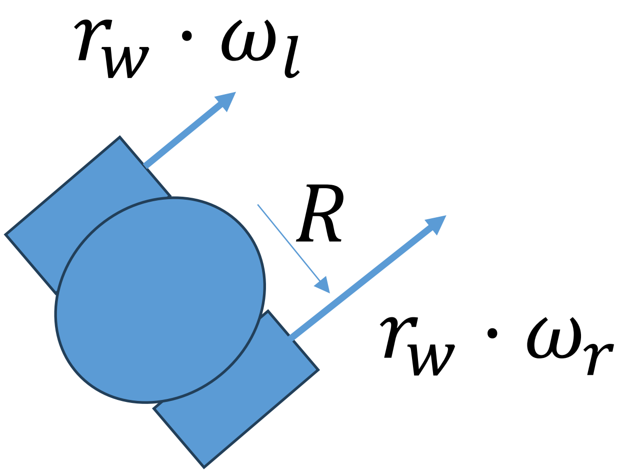
- Initialize the pose $\boldsymbol{T}_0 = \boldsymbol{I}$
- Estimate velocity $\boldsymbol{V}_t$ using internal sensors.
  - Encoders → Kinematic solver → Body velocity
  - IMU → Acceleration + Angular velocity → Body velocity
  - Combine the two
  - Careful: Account for the gravity vector!!!
- Calculate the motion transform $\boldsymbol{M}_t(\boldsymbol{V}_t, \Delta t)$
- Produce the pose at present time $\boldsymbol{T}_t = \boldsymbol{T}_{t-1}\boldsymbol{M}_t$
- Track the covariance: $\Sigma_t^{(T)} \sim f\left(\Sigma_{t-1}^{(T)}, \Sigma_t^{(V)}\right)$

Constructing Odometry



Estimating Velocity - Example

- Differential drive, encoders only
- Measure wheel angular velocities
- Calculate body twist



$$\begin{bmatrix} \omega \\ v_x \\ v_y \end{bmatrix} = \begin{bmatrix} \frac{r_w}{2R}(\omega_r - \omega_l) \\ \frac{r_w}{2}(\omega_r + \omega_l) \\ 0 \end{bmatrix}$$

Estimating Velocity – Covariance

- Start form actuator covariance (wheel)
- How? Characterize, measure, analyze, guess

$$\mathbf{\Sigma}_a = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_l^2 \end{bmatrix} = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix}$$

Assume both wheels have the same statistics!

Estimating Velocity – Covariance

- Find the Jacobian of the motion model

$$\boldsymbol{J}_m = \begin{bmatrix} \frac{r_w}{2R} & -\frac{r_w}{2R} \\ \frac{r_w}{2} & \frac{r_w}{2} \\ 0 & 0 \end{bmatrix} = \frac{r_w}{2}\begin{bmatrix} \frac{1}{R} & -\frac{1}{R} \\ 1 & 1 \\ 0 & 0 \end{bmatrix}$$

$$\boldsymbol{J}_m^T = \frac{r_w}{2}\begin{bmatrix} \frac{1}{R} & 1 & 0 \\ -\frac{1}{R} & 1 & 0 \end{bmatrix}$$

Estimating Velocity – Covariance

- Propagate the covariance

$$\boldsymbol{\Sigma}_v = \boldsymbol{J}_m \boldsymbol{\Sigma}_a \boldsymbol{J}_m^T = \begin{bmatrix} \frac{\sigma^2 r_w^2}{2R^2} & 0 & 0 \\ 0 & \frac{\sigma^2 r_w^2}{2} & 0 \\ 0 & 0 & \epsilon \end{bmatrix}$$

Actuator covariance

Twist covariance

Theoretically zero, but fudge it with something small to avoid singular matrix (it also models random skidding).

Estimating velocity – other options

- Encoders → Linear, Gyroscope → Angular
- Encoders → Linear, Gyro + Encoders → Angular
- Visual/Inertial Odometry (VIO)
- Kalman filter:
  - Control input → Prediction
  - Sensors → Measurement
- Dynamic vs. Kinematic model
  - Must formulate the motion model
- Data-driven (ML) techniques
  - Capture hard-to-model effects
  - Must collect the data → experiments and measurements.
- Different for different drivetrain/locomotion system!

Odometry Pose Tracking (2D)

- Ground robot twist:

$$\boldsymbol{v} = \begin{bmatrix} \omega \\ v_x \\ v_y \end{bmatrix}, \qquad \boldsymbol{\Sigma}_v = \cdots 3 \times 3$$

- Ground robot pose:

$$\boldsymbol{T}_{OB} = \begin{bmatrix} \cos\theta & -\sin\theta & x \\ \sin\theta & \cos\theta & y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}(\theta) & & \boldsymbol{p} \\ 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{\Sigma}_T = \cdots \ 3 \times 3$$

Odometry Pose Tracking (2D)

- Integrate the pose (local frame of reference):

$$\boldsymbol{T}_{OB}[n+1] = \begin{bmatrix} \cos\theta & -\sin\theta & x \\ \sin\theta & \cos\theta & y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\omega\Delta t & -\sin\omega\Delta t & v_x\Delta t \\ \sin\omega\Delta t & \cos\omega\Delta t & v_y\Delta t \\ 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{T}_{OB}[n+1] = \begin{bmatrix} \boldsymbol{R}(\theta) & \boldsymbol{p} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{R}(\omega\Delta t) & \boldsymbol{v}\Delta t \\ 0 & 1 \end{bmatrix}$$

Exponential mapping of 2D twist!

$$\boldsymbol{T}_{OB}[n+1] = \begin{bmatrix} \boldsymbol{R}(\theta+\omega\Delta t) & \boldsymbol{R}(\theta)\boldsymbol{v}\Delta t + \boldsymbol{p} \\ 0 & 1 \end{bmatrix}$$

Read off new orientation!

Read off new position!

Odometry Pose Tracking (2D)

- Extract the pose from the transformation matrix:

$$\begin{bmatrix} \theta[n+1] \\ x[n+1] \\ y[n+1] \end{bmatrix} = \begin{bmatrix} \theta[n] + \omega\Delta t \\ v_{\mathrm{x}}[\mathrm{n}]\Delta\mathrm{t} \cdot \cos\theta[n] - v_y[n]\Delta\mathrm{t} \cdot \sin\theta[n] + x[n] \\ v_{\mathrm{x}}[\mathrm{n}]\Delta\mathrm{t} \cdot \sin\theta[n] + v_y[n]\Delta\mathrm{t} \cdot \cos\theta[n] + y[n] \end{bmatrix}$$

$$\boldsymbol{g}(\underbrace{v_x, v_y, \omega}_{\text{Inputs}}, \underbrace{x, y, \theta}_{\text{State}})$$

Remember this?

$$\overline{\boldsymbol{x}}[n] = \boldsymbol{g}(\boldsymbol{x}[n-1], \boldsymbol{u}[n])$$

$$\overline{\boldsymbol{\Sigma}}_x[n] = \boldsymbol{G}_x \boldsymbol{\Sigma}_x[n-1] \boldsymbol{G}_x^T + \boldsymbol{G}_u \boldsymbol{\Sigma}_u[n] \boldsymbol{G}_u^{\mathrm{T}}$$

$$\boldsymbol{G}_x = \left[\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}\right] \qquad \boldsymbol{G}_u = \left[\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{u}}\right]$$

Jacobians

- State:

$$\boldsymbol{G}_x = \begin{bmatrix} 1 & 0 & 0 \\ -v_x \Delta t \sin\theta - v_y \Delta t \cos\theta & 1 & 0 \\ v_x \Delta t \cos\theta - v_y \Delta t \sin\theta & 0 & 1 \end{bmatrix}$$

- Input:

$$\boldsymbol{G}_u = \begin{bmatrix} \Delta t & 0 & 0 \\ 0 & \Delta t \cos\theta & -\Delta t \sin\theta \\ 0 & \Delta t \sin\theta & \Delta t \cos\theta \end{bmatrix}$$

- Note: For differential drive, $v_y = 0$

Covariance Accumulation

$$\boxed{\boldsymbol{\Sigma}_T[n] = \boldsymbol{G}_x \boldsymbol{\Sigma}_T[n-1] \boldsymbol{G}_x^T + \boldsymbol{G}_u \boldsymbol{\Sigma}_v[n] \boldsymbol{G}_u^{\mathrm{T}}}$$

$$\boldsymbol{\Sigma}_T[0] = \mathbf{0}$$

$$\boldsymbol{\Sigma}_T[1] = \boldsymbol{G}_u \boldsymbol{\Sigma}_v[0] \boldsymbol{G}_u^{\mathrm{T}}$$

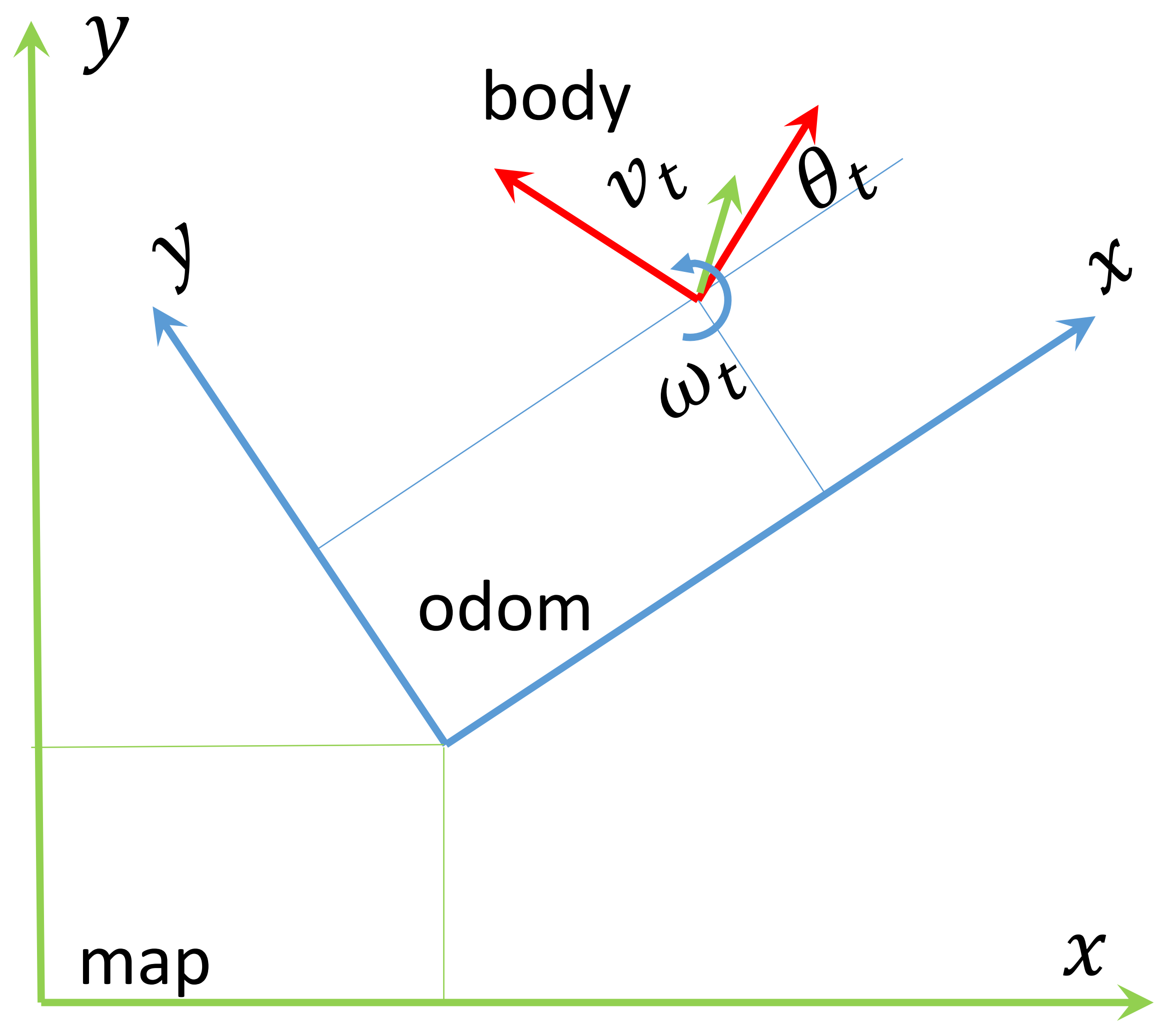
$$\boldsymbol{\Sigma}_T[2] = \boldsymbol{G}_x \boldsymbol{G}_u \boldsymbol{\Sigma}_v[0] \boldsymbol{G}_u^{\mathrm{T}} \boldsymbol{G}_x^T + \boldsymbol{G}_u \boldsymbol{\Sigma}_v[1] \boldsymbol{G}_u^{\mathrm{T}}$$

$$\boldsymbol{\Sigma}_T[3] = \boldsymbol{G}_x^2 \boldsymbol{G}_u \boldsymbol{\Sigma}_v[0] \boldsymbol{G}_u^{\mathrm{T}} (\boldsymbol{G}_x^T)^2 + \boldsymbol{G}_x \boldsymbol{G}_u \boldsymbol{\Sigma}_v[1] \boldsymbol{G}_u^{\mathrm{T}} \boldsymbol{G}_x^T + \boldsymbol{G}_u \boldsymbol{\Sigma}_v[2] \boldsymbol{G}_u^{\mathrm{T}}$$

- Covariance grows without bounds
- Good for estimating uncertainty accumulation over finite time window!

Practical uses of odometry

- Dead Reckoning
- Extrapolation
- Prediction
- Time alignment
- Localization



Tracking Odometry (3D)

- Pose is in SE(3) space.
- Covariance is in exponential coordinates se(3) space.

$$\widetilde{\mathbf{T}}[n+1] = \widetilde{\mathbf{T}}[n]e^{[\boldsymbol{s}]\omega\Delta t}$$

Twist, uncertainty in local frame!

$$\boldsymbol{S} = \begin{bmatrix} \boldsymbol{k}_\omega \\ \boldsymbol{k}_v \end{bmatrix} = \begin{bmatrix} \dfrac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|} \\ \dfrac{\boldsymbol{v}}{\|\boldsymbol{\omega}\|} \end{bmatrix}$$

Tracking Covariance (3D)

- Recall covariance propagation for local frame:

$$\boldsymbol{\Sigma} = Adj\boldsymbol{T}_2^{-1}\boldsymbol{\Sigma}_1(Adj\boldsymbol{T}_2^{-1})^T + \boldsymbol{\Sigma}_2$$

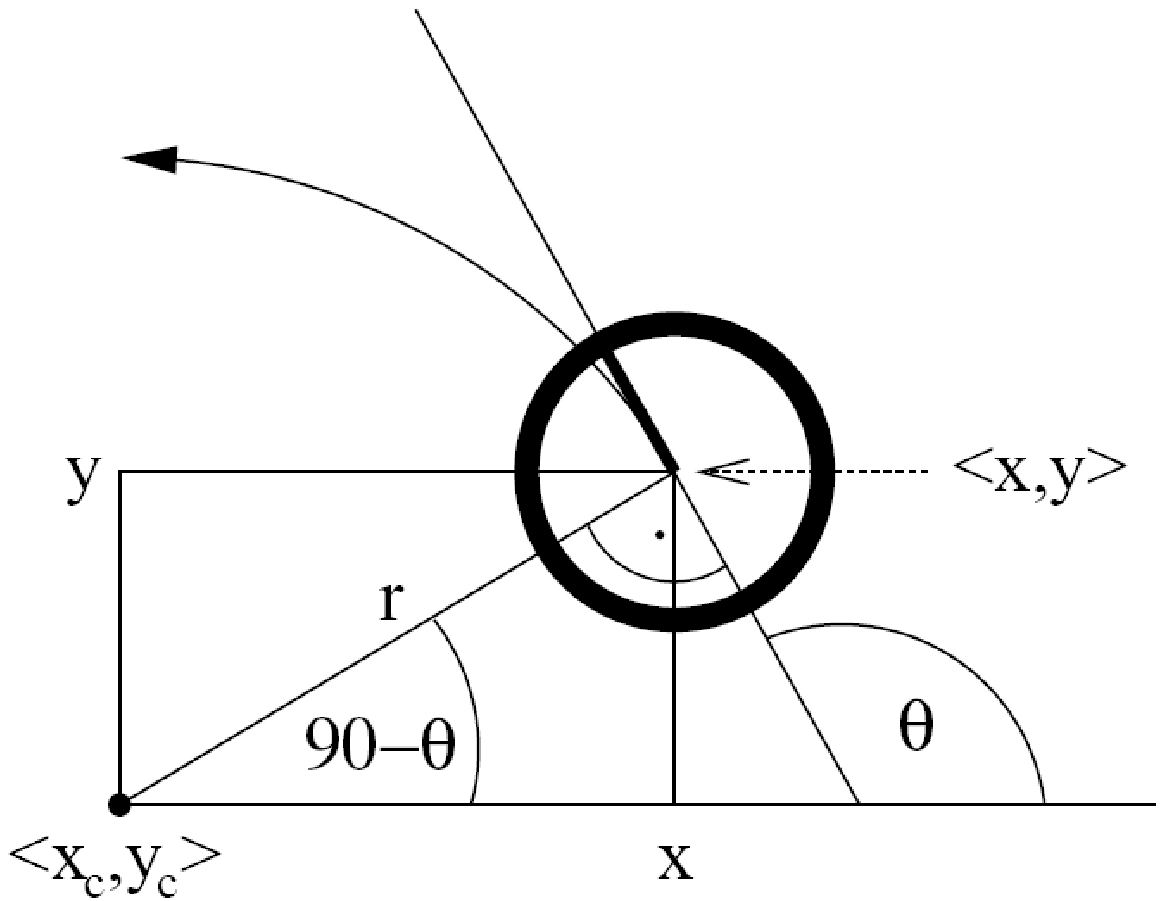
- Second transform comes from twist estimate:

$$\boldsymbol{\Sigma}[\mathrm{n}+1] = Adj\ e^{-[\boldsymbol{s}]\omega\Delta t}\ \boldsymbol{\Sigma}[n]\left(Adj\ e^{[\boldsymbol{s}]\omega\Delta t}\ \right)^T + \Delta\mathrm{t}^2\boldsymbol{\Sigma}_v$$

- Remember, for ROS representation:
  - Transform $\boldsymbol{\Sigma}[n]$ to local frame.
  - Multiply with adjoints.
  - Transform back to ROS representation.

Arc-motion tracking

- Chapter 5 in Thrun



Source: probabilistic-robotics.org

Arc-motion tracking

$$x_c = x[n-1] - r\sin\theta$$

$$y_c = y[n-1] + r\cos\theta$$

$$r = \frac{v}{\omega}$$

$$x[n] = x_c + r\sin(\theta + \omega\Delta t)$$

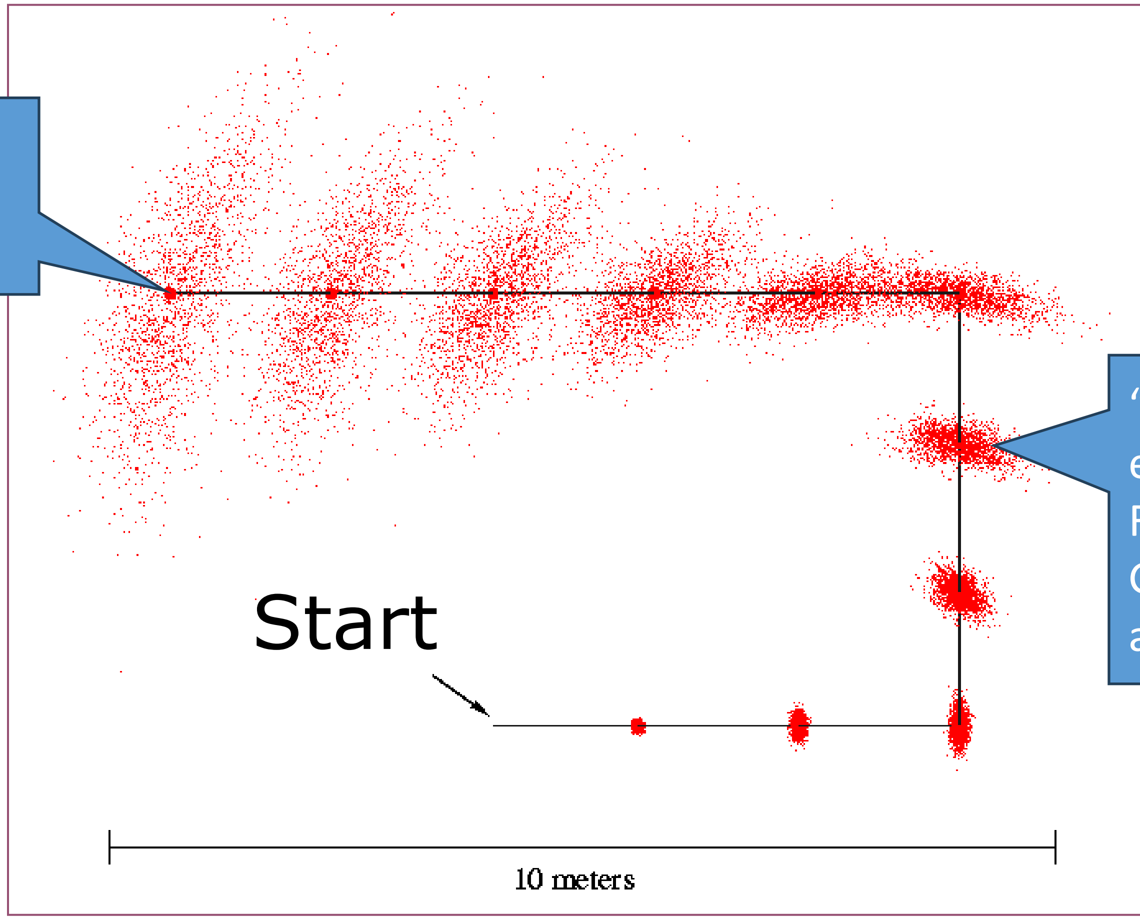
$$y[n] = y_c - r\cos(\theta + \omega\Delta t)$$

Non-Gaussian models (2D)

- Chapter 5 in Thrun
- Two models: PDF and Monte-Carlo
- Sometimes called “Banana Distribution”

Deviations from Gaussian Model

Distribution deformed into a bent shape.

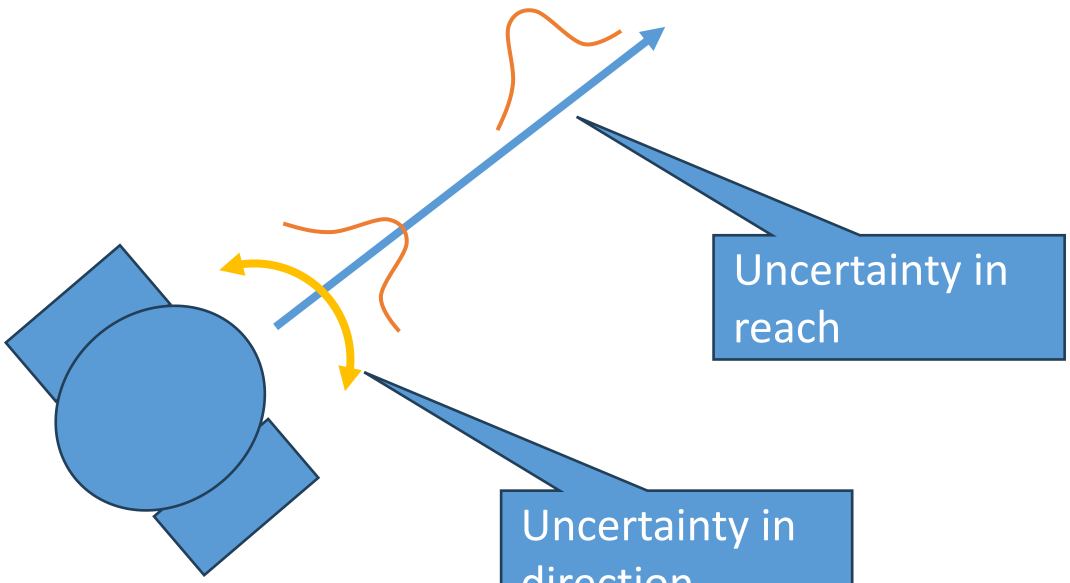


"Good looking" ellipse – Reasonable Gaussian approximation.

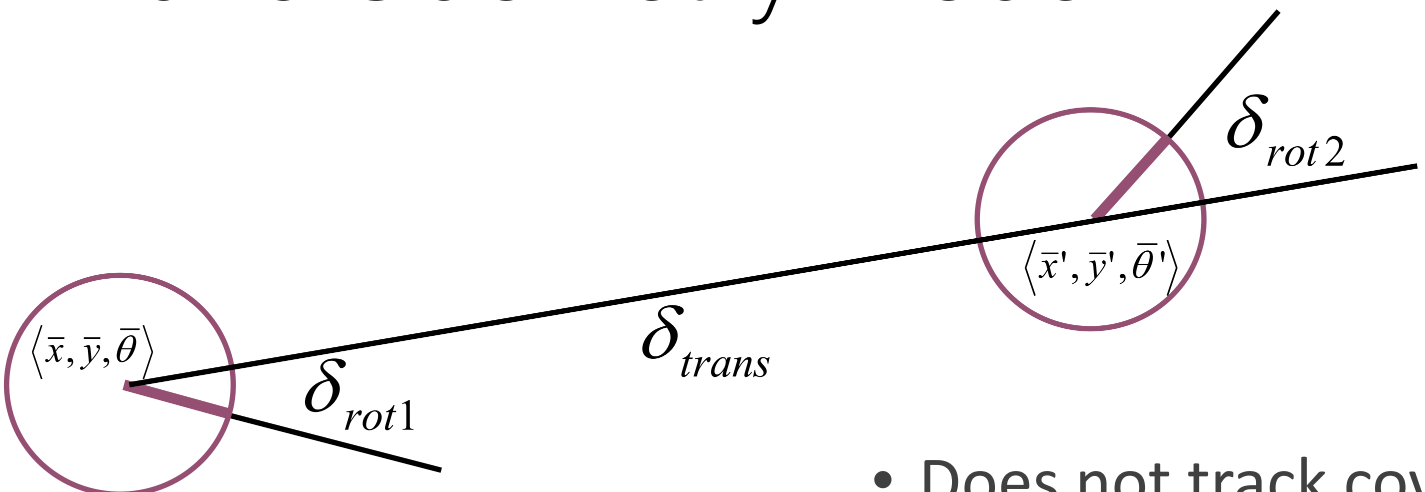
Source: probabilistic-robotics.org

Cause of Deviations

- Steering (non-holonomic) drivetrain:



Thrun's Odometry Model



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

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

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

- Does not track covariance
- Looks at two end-poses in elapsed odometry
- Constructs the distribution (or draws samples)
- Empirical parameters characterize the robot

Source: probabilistic-robotics.org

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

odometry values (u)

values of interest (x,x′)

Source: probabilistic-robotics.org

Random sources

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

Gaussian, or triangular, or any single-mode distribution that fades away with distance form the mean

Rationale: more the robot has moved, more standard deviation we need to add to the random source

- Alpha parameters are empirically tuned for the robot

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

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

sample_normal_distribution

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

Examples