CMPE 185 Autonomous Mobile Robots

Mobile Robot Odometry

Dr. Wencen Wu

Computer Engineering Department

Mobile Robot Kinematics in Practice – Wheel Encoders

- In practice, the forward velocities of the wheels v_1 and v_2/ angular velocities of wheels $\dot{\varphi}_1$ and $\dot{\varphi}_2$ are difficult to measure directly and accurately

- The rotation of each wheel can be measured by **wheel encoders**

- A **rotary encoder**, also called a **shaft encoder**, is an electromechanical device that converts the angular position or motion of a shaft or axle to analog or digital output signals.

Encoders

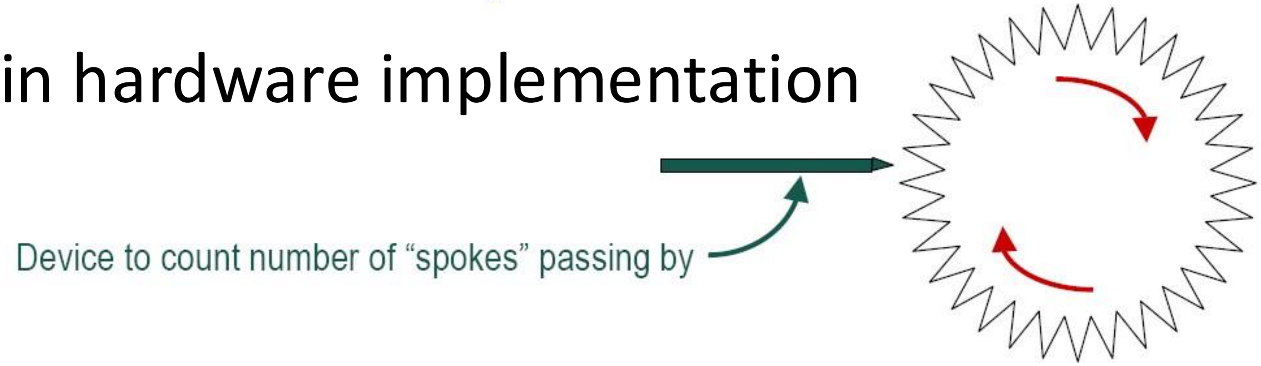
- Purpose
  - To measure turning distance of motors (in terms of rotations), which can be converted to robot translation/rotation distance
- A **digital optical encoder** is a device that converts motion into a sequence of digital pulses. By counting a single bit or by decoding a set of bits, the pulses can be converted to relative or absolute position measurements
- If wheel size is known, number of motor turns -> number of wheel turns -> estimation of distance robot has traveled

Encoders

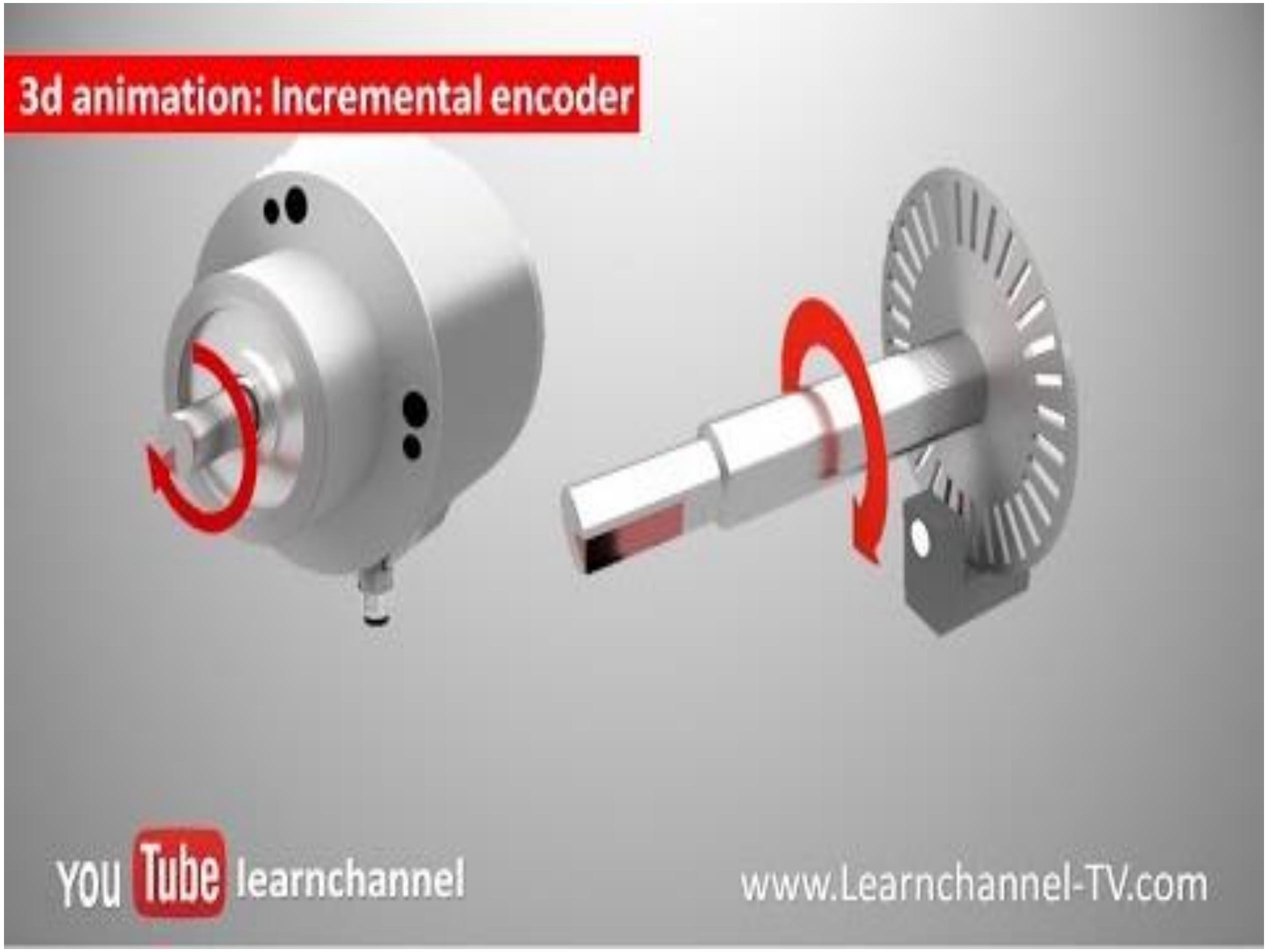
There are two types of encoders

- Absolute encoders
  - measure the current orientation of a wheel
- Incremental encoders
  - measure the change in orientation of a wheel

- Basic idea in hardware implementation



How an incremental encoder works?



Encoders and Dead Reckoning

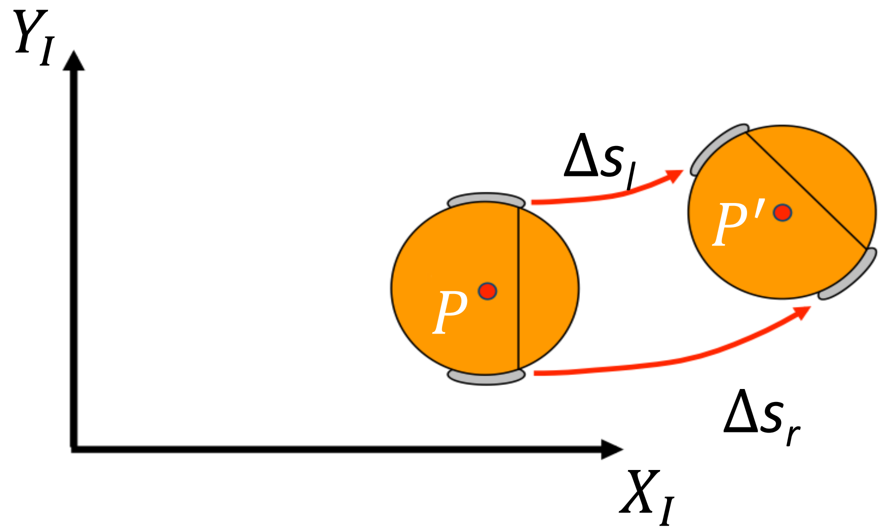
- Odometry
  - Use wheel encoders to update position

- Dead reckoning
  - The process of estimating one’s current position based upon a previously determined position and advancing that position based upon known speed, elapsed time, and course

- Straight forward to implement

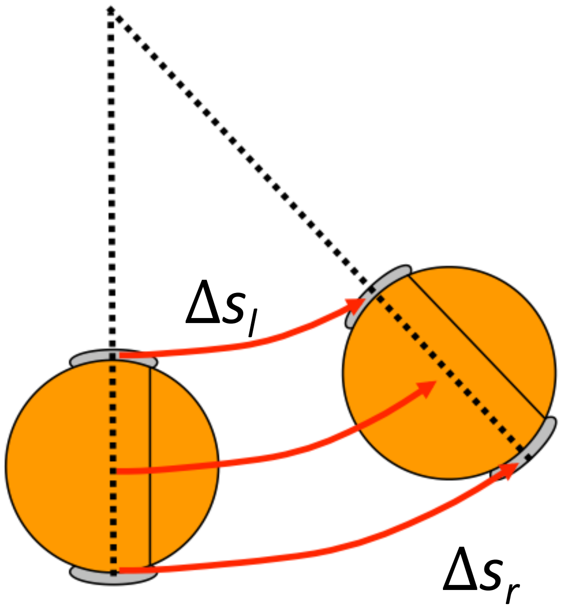
Modeling Motion

- Wheel encoders will give the distance moved by each wheel
- If a robot starts from a position *p*, and the right and left wheels move respective distances Δs_r and Δs_l, what is the resulting new position *p'*?



Modeling motion

- To start, let's model the change in angle $\Delta\theta$ and distance travelled Δs by the robot
- Assume the robot is travelling on a circular arc of constant radius



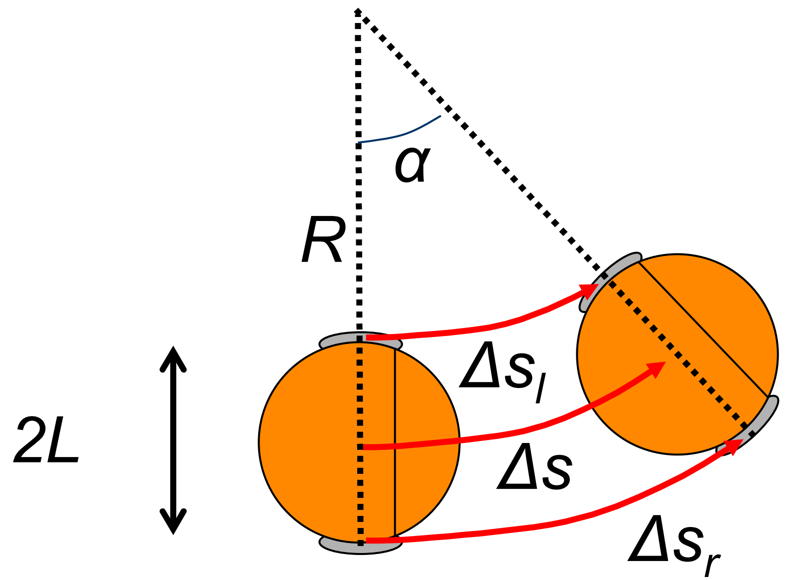
Modeling motion

- Begin by noting the following holds for circular arcs:

$$\Delta s_l = R\alpha$$

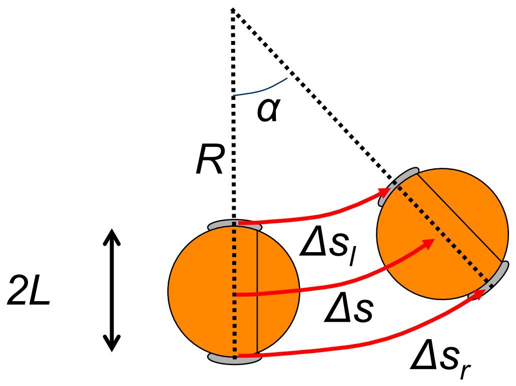
$$\Delta s_r = (R + 2L)\alpha$$

$$\Delta s = (R + L)\alpha$$



Modeling Motion

- Now manipulate first two equations



$$\Delta s_l = R\alpha$$

$$\Delta s_r = (R + 2L)\alpha$$

To

$$R\alpha = \Delta s_l$$

$$L\alpha = \frac{(\Delta s_r - R\alpha)}{2} = \frac{\Delta s_r}{2} - \frac{\Delta s_l}{2}$$

- Substitute this into last equation for delta for Δ*s*:

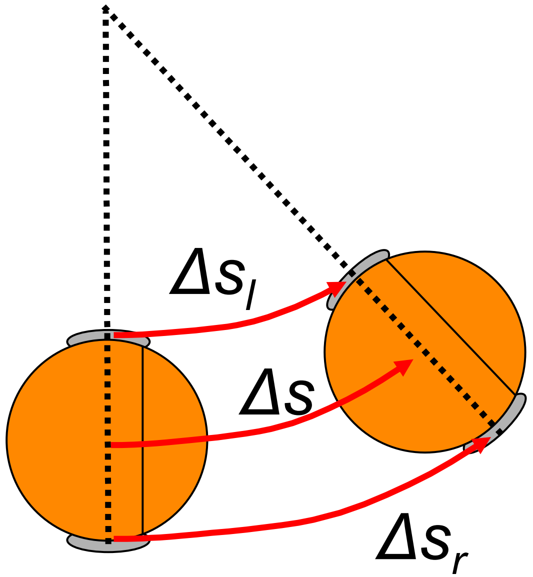
$$\Delta s = (R + L)\alpha = R\alpha + L\alpha = \Delta s_l + \frac{\Delta s_r}{2} - \frac{\Delta s_l}{2} = \frac{\Delta s_l}{2} + \frac{\Delta s_r}{2}$$

$$= \frac{\Delta s_l + \Delta s_r}{2}$$

Modeling Motion

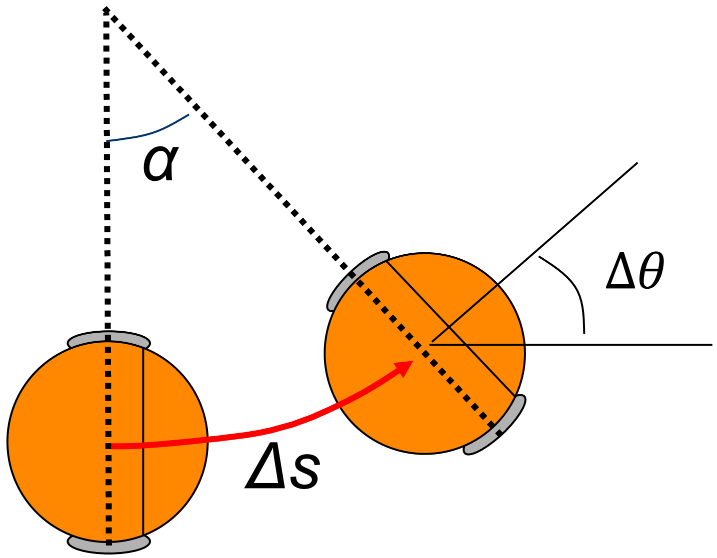
- Or, note the distance the center travelled is simply the average distance of each wheel

$$\Delta s = \frac{\Delta s_l + \Delta s_r}{2}$$



Modeling Motion

- To calculate the change in angle $\Delta\theta$, observe that it equals the rotation about the circular arc's center point



$$\Delta\theta = \alpha$$

Modeling Motion

- We solve for α by equating α from the first two equations:

$$\Delta s_l = R\alpha$$

$$\Delta s_r = (R + 2L)\alpha$$

$$\frac{\Delta s_l}{R} = \frac{\Delta s_r}{(R + 2L)}$$

$$(R + 2L)\,\Delta s_l = R\,\Delta s_r$$

$$2L\,\Delta s_l = R\,(\Delta s_r - \Delta s_l)$$

- This results in $\dfrac{2L\,\Delta s_l}{(\Delta s_r - \Delta s_l)} = R$

Modeling Motion

- Substitute *R* into

$$\alpha = \frac{\Delta s_l}{R}$$

$$= \frac{\Delta s_l(\Delta s_r - \Delta s_l)}{2L\,\Delta s_l}$$

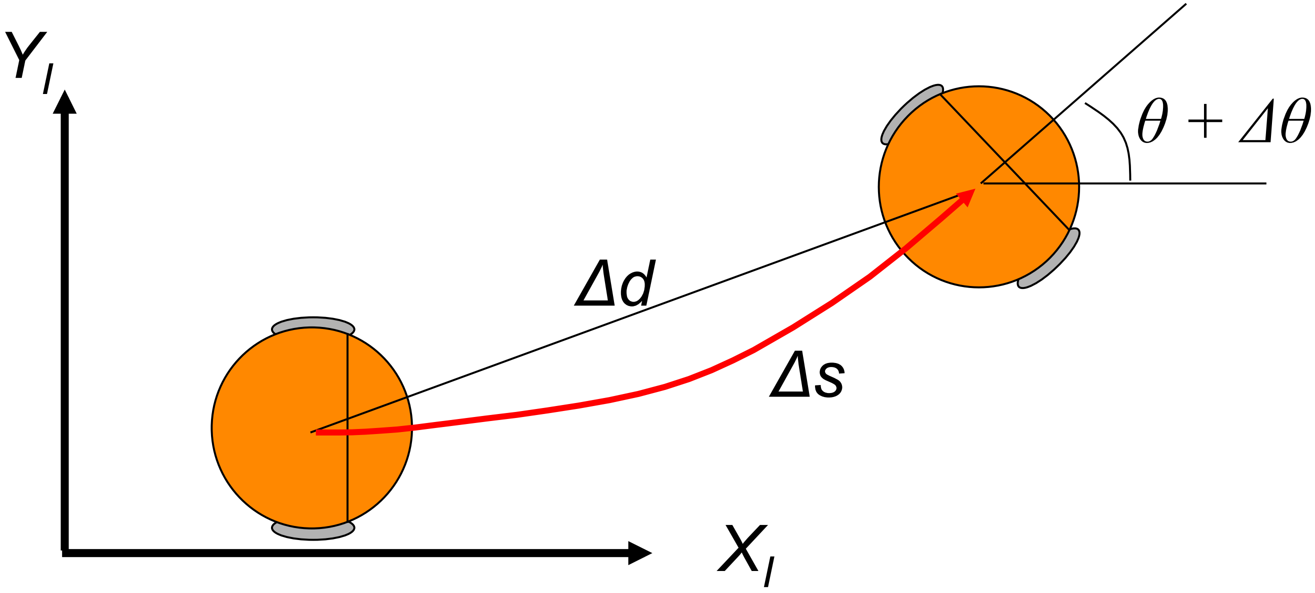
$$= \frac{(\Delta s_r - \Delta s_l)}{2L}$$

- So…

$$\Delta\theta = \frac{(\Delta s_r - \Delta s_l)}{2L}$$

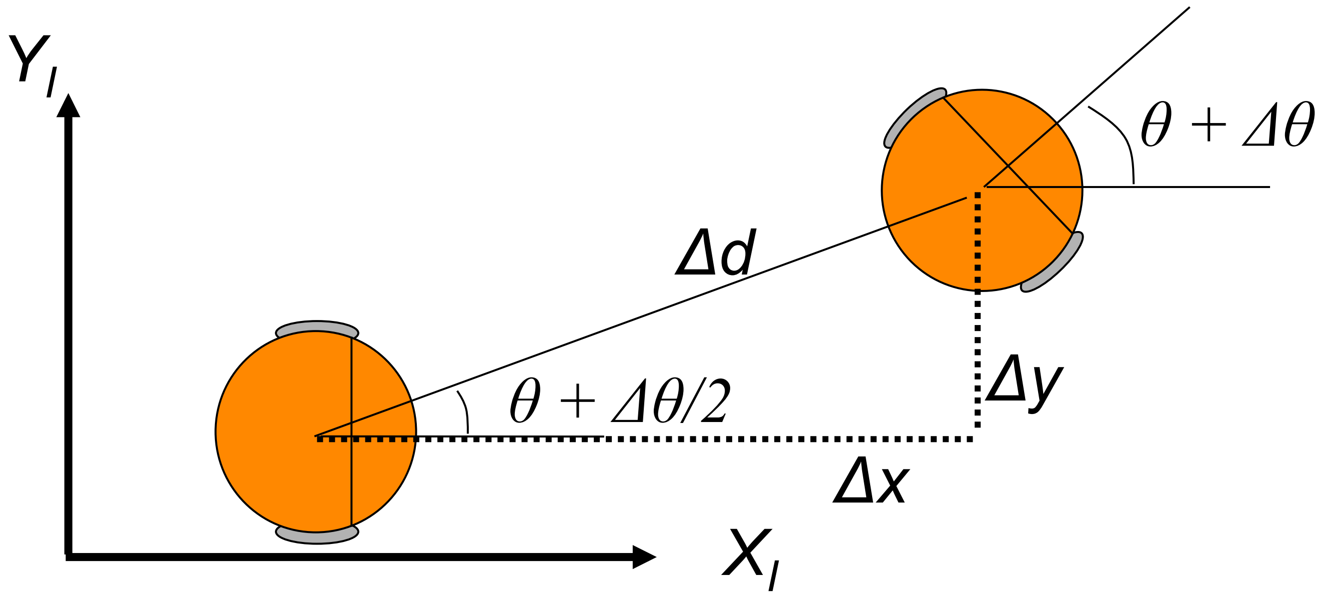
Modeling Motion

- Now that we have $\Delta\theta$ and Δs, we can calculate the position change in global coordinates
- We use a new segment of length Δd



Modeling Motion

- Assume $\Delta\theta$ is relatively small, so…



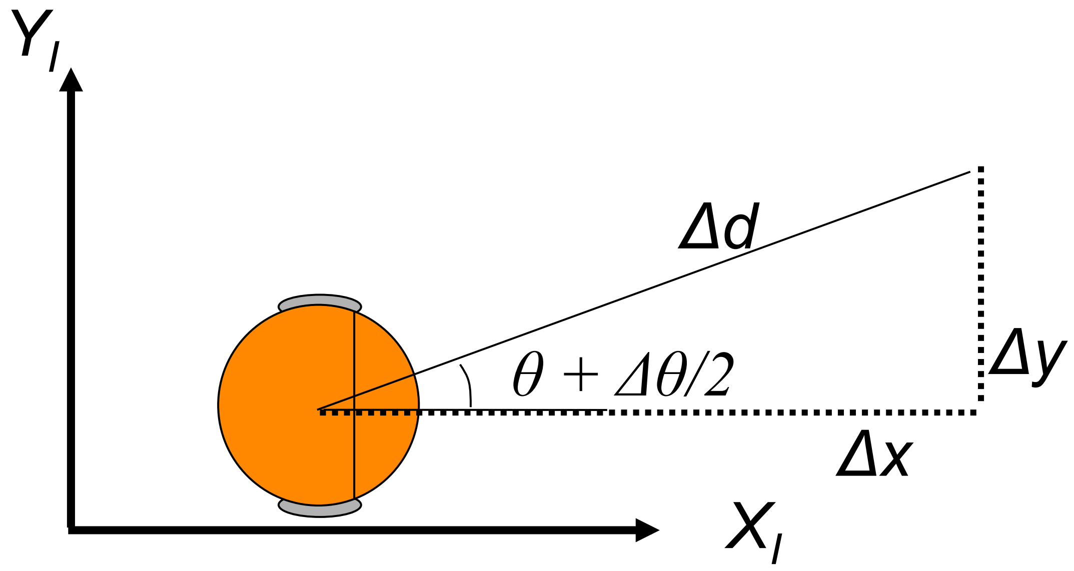
- Now calculate the change in position as a function of Δd

Modeling Motion

- Using Trigonometry

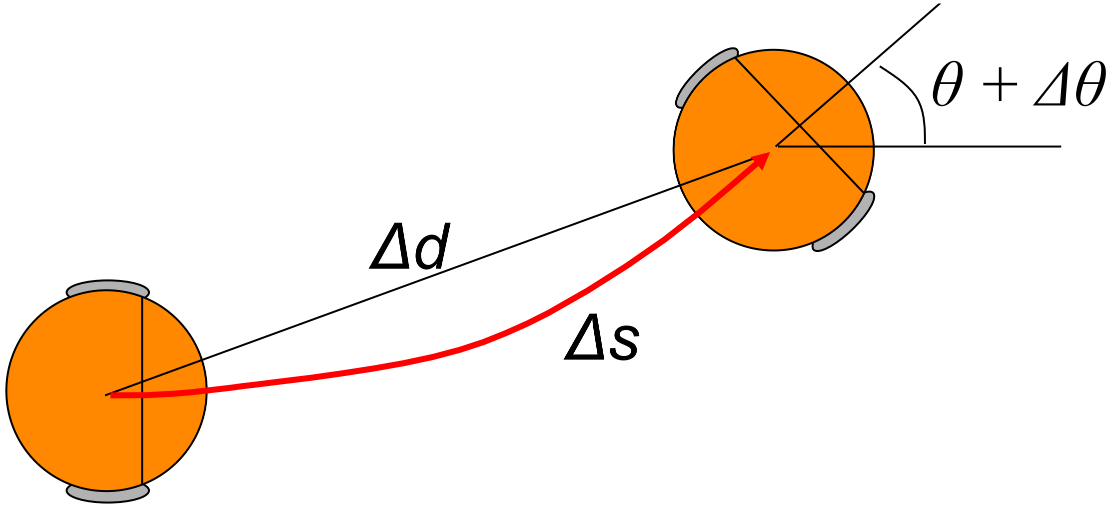
$$\Delta x = \Delta d \cos(\theta + \frac{\Delta\theta}{2})$$

$$\Delta y = \Delta d \sin(\theta + \frac{\Delta\theta}{2})$$



Modeling Motion

- Now if we assume that the motion is small, then we can assume that $\Delta d \approx \Delta s$



- So…

$$\Delta x = \Delta s \cos(\theta + \frac{\Delta\theta}{2})$$

$$\Delta y = \Delta s \sin(\theta + \frac{\Delta\theta}{2})$$

Modeling Motion

Summary

$$\Delta x = \Delta s \cos(\theta + \frac{\Delta\theta}{2})$$

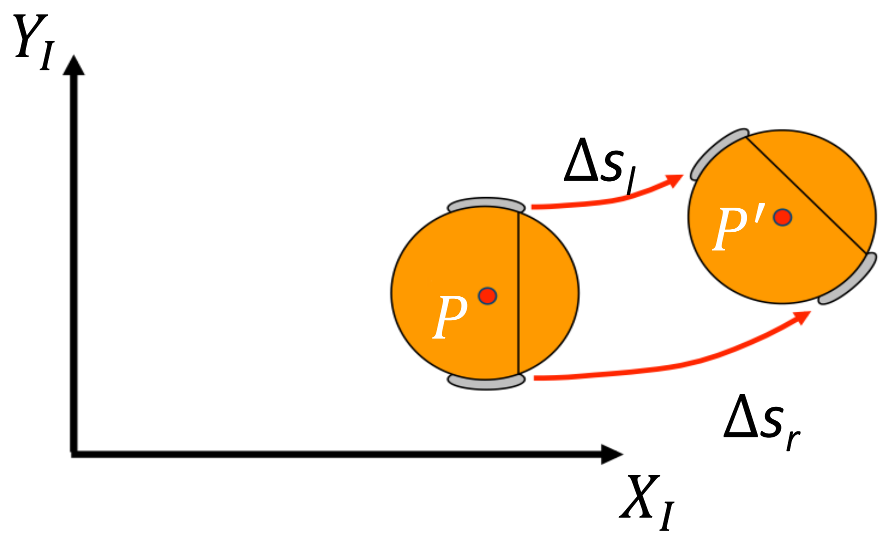
$$\Delta y = \Delta s \sin(\theta + \frac{\Delta\theta}{2})$$

$$\Delta\theta = \frac{\Delta s_r - \Delta s_l}{2L}$$

$$\Delta s = \frac{\Delta s_l + \Delta s_r}{2} \qquad \Delta\theta = \frac{(\Delta s_r - \Delta s_l)}{2L}$$

Recall: Modeling Motion

- Wheel encoders will give the distance moved by each wheel
- If a robot starts from a position *p*, and the right and left wheels move respective distances Δs_r and Δs_l, what is the resulting new position *p'*?



Modeling Motion

- So from…

$$p' = p + \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{pmatrix}$$

$$\Delta x = \Delta s \cos(\theta + \frac{\Delta\theta}{2})$$

$$\Delta y = \Delta s \sin(\theta + \frac{\Delta\theta}{2})$$

$$\Delta\theta = \frac{\Delta s_r - \Delta s_l}{2L}$$

$$\Delta s = \frac{\Delta s_l + \Delta s_r}{2}$$

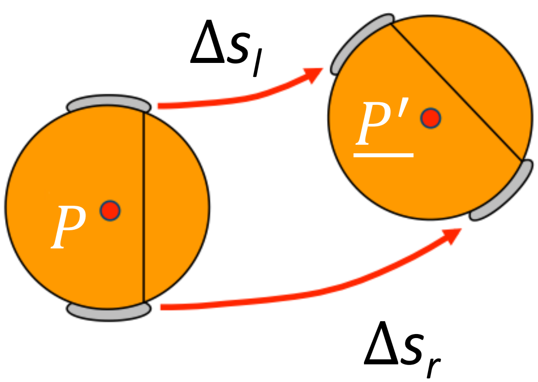
$$\Delta\theta = \frac{(\Delta s_r - \Delta s_l)}{2L}$$

- We can calculate the new position as

$$p' = f(x, y, \theta, \Delta s_r, \Delta s_l) = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{\Delta s_r + \Delta s_l}{2} \cos(\theta + \frac{\Delta s_r - \Delta s_l}{4L}) \\ \frac{\Delta s_r + \Delta s_l}{2} \sin(\theta + \frac{\Delta s_r - \Delta s_l}{4L}) \\ \frac{\Delta s_r - \Delta s_l}{2L} \end{bmatrix}$$

Mobile Robot Kinematics in Practice – Wheel Encoders

- Wheel encoders will give the distance moved by each wheel
- But how do we know how far each wheel has moved?
- Assume each wheel has *N* “ticks” per revolution
- Most wheel encoders give the total tick count since the beginning



For both wheels:

$$\Delta\text{tick} = \text{tick}' - \text{tick}$$

$$\Delta s = 2\pi r \frac{\Delta\text{tick}}{N}$$

For each wheel:

$$\Delta s_l = 2\pi r \frac{\Delta tick_l}{N}$$

$$\Delta s_r = 2\pi r \frac{\Delta tick_r}{N}$$

An Odometry Example

- If my robot starts at the origin (position = $[0, 0]_{T,}$ and orientation is 0), where is it located after 0.1s, given that 10 ticks were recorded for the right wheel and 6 ticks for the left wheel. The wheel radius is 2m, the total ticks per revolution is 100. The distance between wheels is 4m.

$$p' = f(x, y, \theta, \Delta s_r, \Delta s_l) = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{\Delta s_r + \Delta s_l}{2} \cos(\theta + \frac{\Delta s_r - \Delta s_l}{4L}) \\ \frac{\Delta s_r + \Delta s_l}{2} \sin(\theta + \frac{\Delta s_r - \Delta s_l}{4L}) \\ \frac{\Delta s_r - \Delta s_l}{2L} \end{bmatrix}$$

For each wheel:

$$\Delta s_l = 2\pi r \frac{\Delta tick_l}{N}$$

$$\Delta s_r = 2\pi r \frac{\Delta tick_r}{N}$$

Odometry & Dead Reckoning

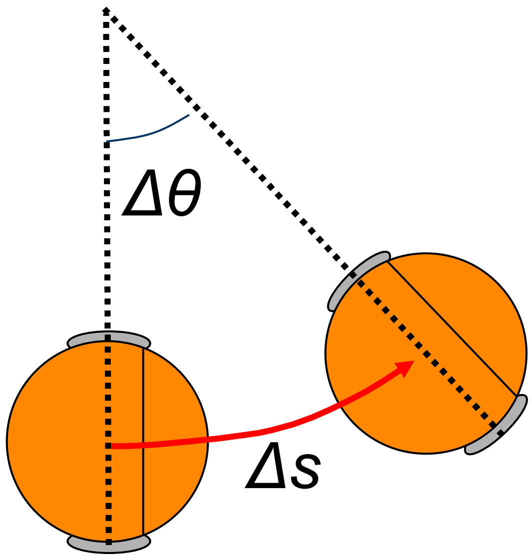
- Odometry error sources
  - Limited resolution during integration (time increments, measurement resolution)
  - Unequal wheel diameter (deterministic)
  - Variation in the contact point of the wheel (deterministic)
  - Unequal floor contact and variable friction can lead to slipping (non-deterministic)

- Odometry errors
  - Deterministic errors can be eliminated through proper calibration
  - Non-deterministic errors have to be described by error models and will always lead to uncertain position estimate

Modeling Uncertainty in Motion

- Let's look at delta terms as errors in wheel motion, and see how they propagate into positioning errors
- Example: the robot is trying to move forward *1* m on the *x* axis

where is the robot after the movement?



$$\Delta s = 1 + e_s$$

$$\Delta\theta = 0 + e_\theta$$

where e_s and e_θ are error terms

Modeling Uncertainty in Motion

- According to the following equations, the error e_s = 0.001m produces errors in the direction of motion

$$\Delta x = \Delta s \cos(\theta + \frac{\Delta\theta}{2})$$

$$\Delta y = \Delta s \sin(\theta + \frac{\Delta\theta}{2})$$

- However, the $\Delta\theta$ term affects each direction differently
- If e_θ = 2 deg and $e_s = 0$ meters, then

$$\cos\left(\theta + \frac{\Delta\theta}{2}\right) = 0.9998$$

$$\sin\left(\theta + \frac{\Delta\theta}{2}\right) = 0.0175$$

Modeling Uncertainty in Motion

- So

$$\Delta x = 0.9998$$

$$\Delta y = 0.0175$$

- But the robot is supposed to go to $x = 1$, $y = 0$, so the errors in each direction are

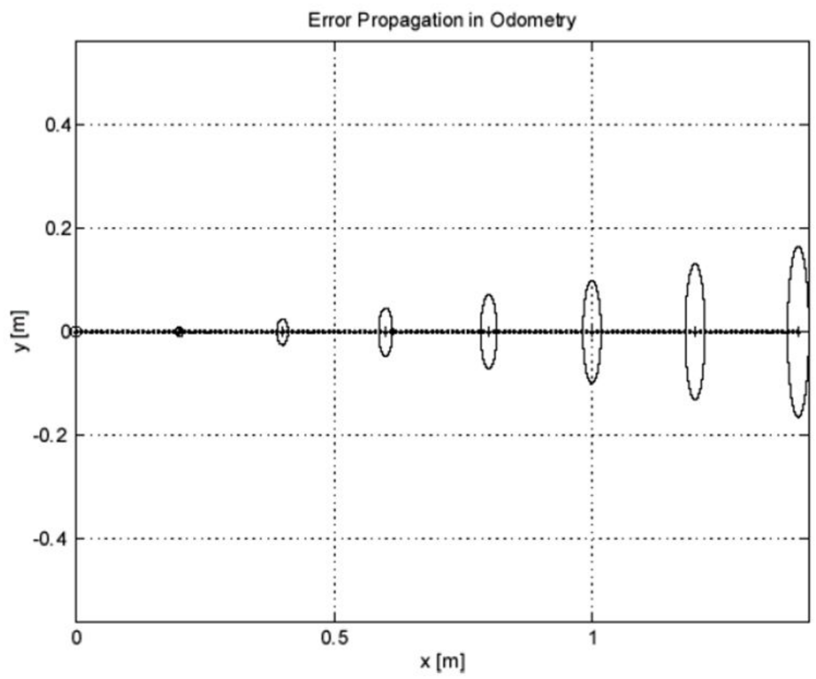
$$\Delta e_x = +0.0002$$
$$\Delta e_y = -0.0175$$

- THE ERROR IS BIGGER IN THE "Y" DIRECTION

Modeling Uncertainty in Motion

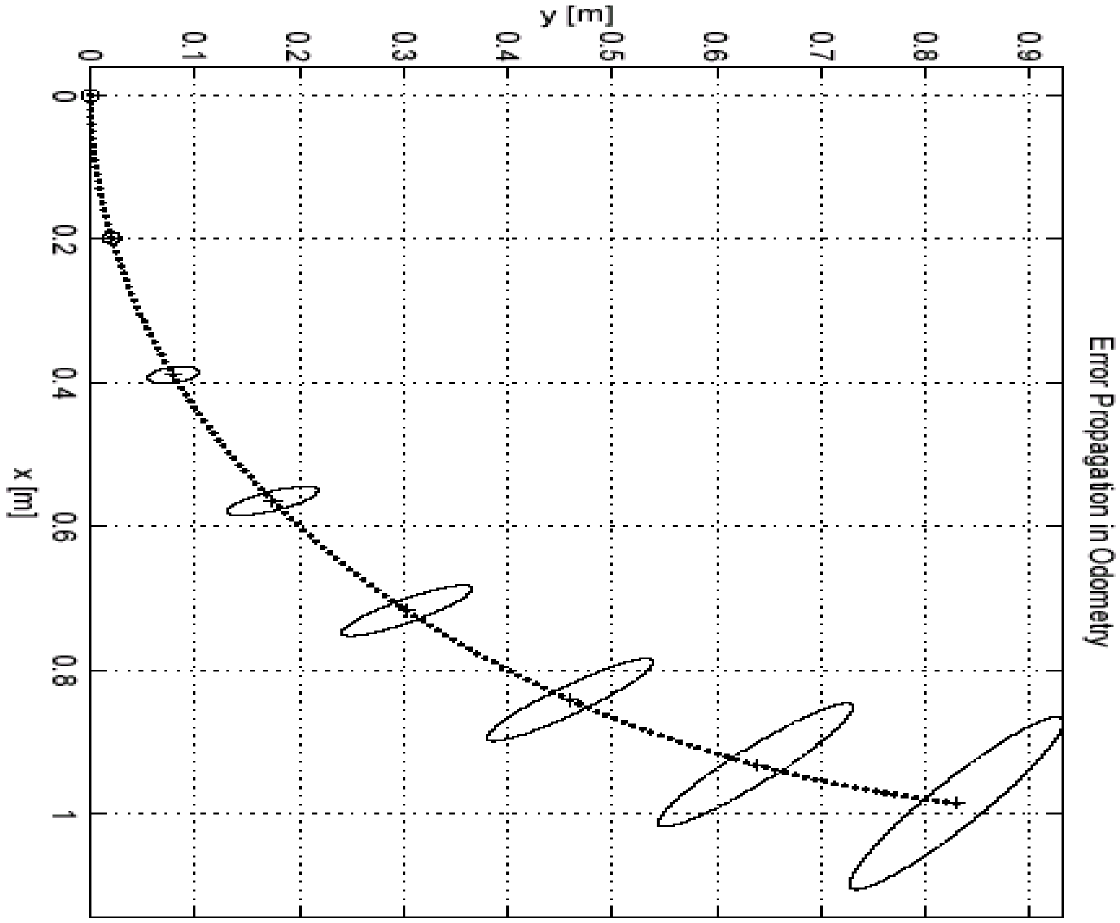
- Errors perpendicular to the direction grow much larger

  - Simple error model from kinematics



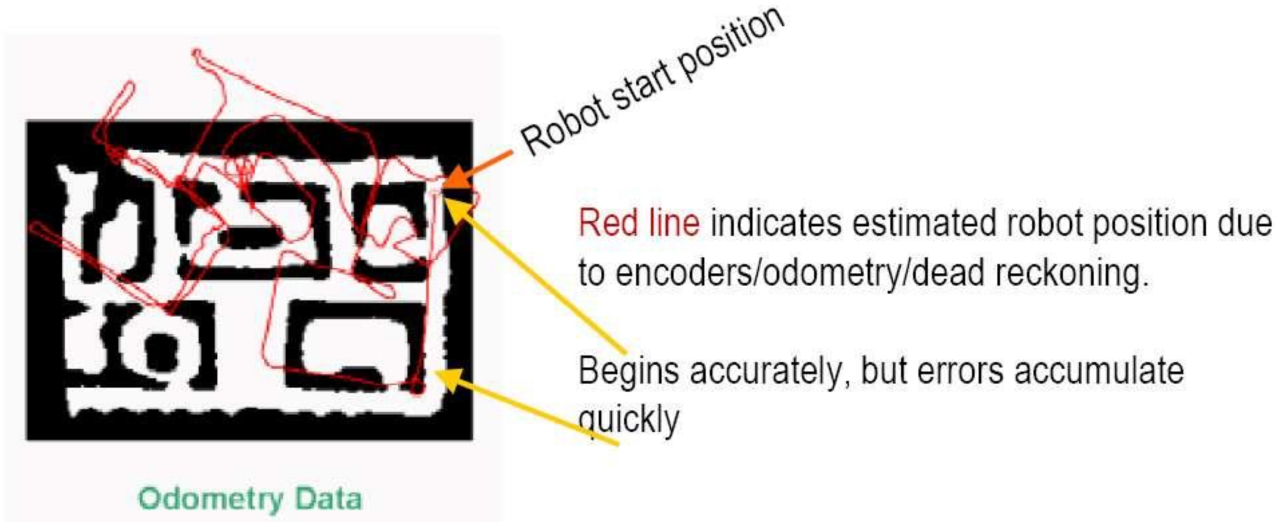
Modeling Uncertainty in Motion

- Error ellipse does not remain perpendicular to direction



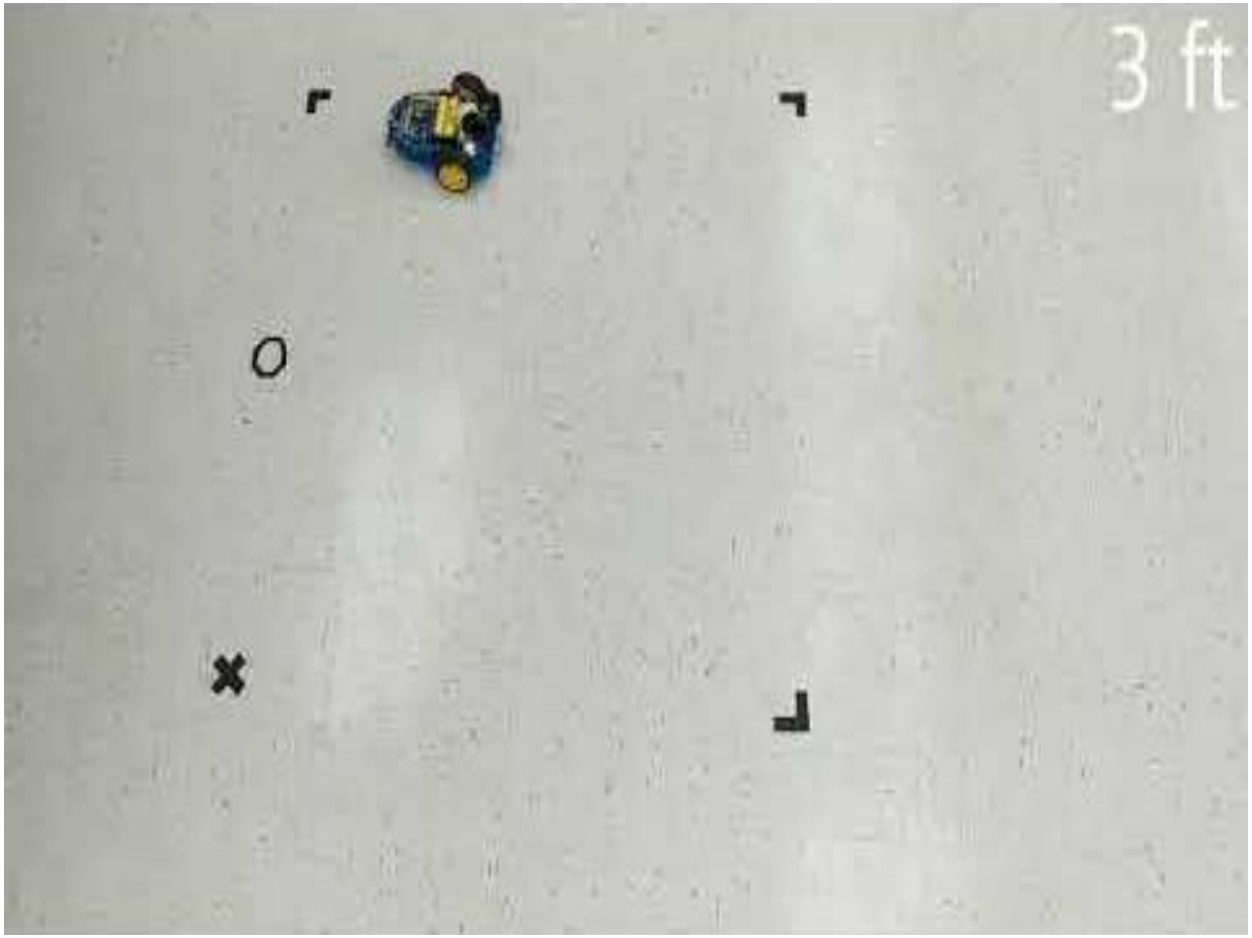
Dead Reckoning Errors

- Challenges/issues:
  - Errors are integrated, unbounded
  - Motion of wheels not corresponding to robot motion, e.g., due to wheel spinning
  - Wheels don't move but robot does, e.g., due to robot sliding



Robot Navigation: Dead-reckoning Error

- Error accumulates quickly, especially due to turning



- Thank you!