اصول علم ربات – اسلاید نهم

Fundamentals of Robotics – Slide 09

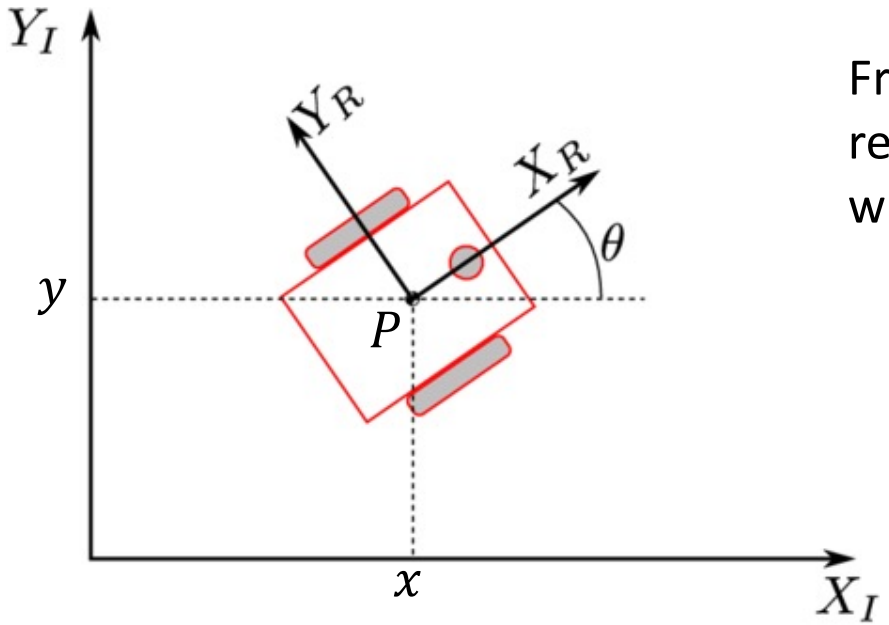
Kinematics 3

دکتر مهدی جوانمردی

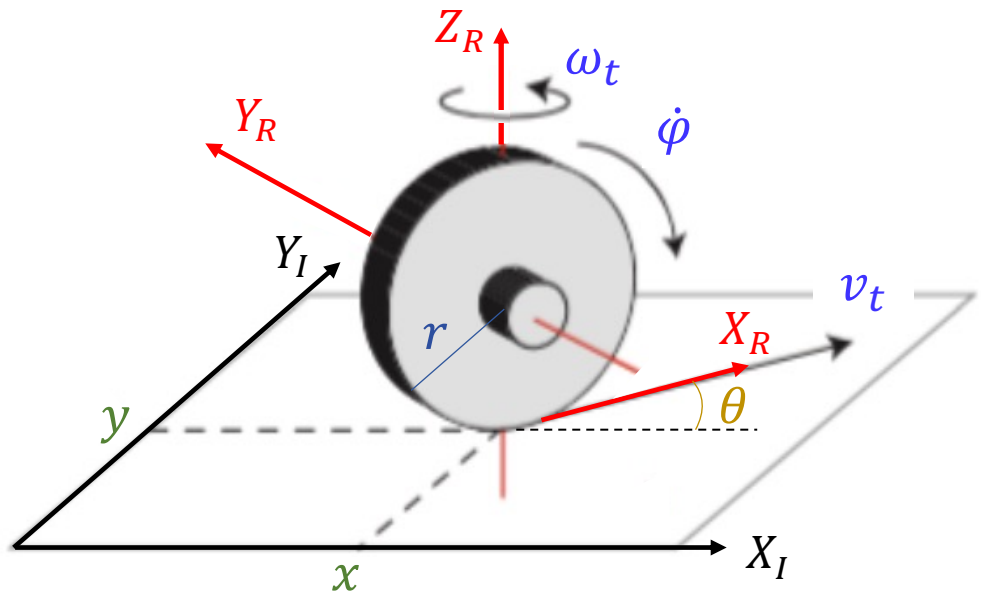
زمستان ۱۴۰۰ – بهار ۱۴۰۱

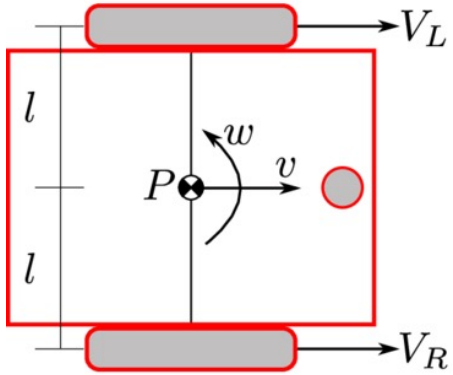
[slides adapted from Gianni Di Caro, @CMU with permission]

Differential kinematics for differential robot: unicycle model



From a purely kinematic perspective, the motion of the robot's reference frame R centered in P is equivalent to a **unicycle** model, which abstracts the (ideal) motion of a **standard steering wheel**





Summary of differential kinematics equations

$$\dot{\xi}_I = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v(t)\cos\theta \\ v(t)\sin\theta \\ \omega(t) \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} v(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega(t) = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix}$$

$$\dot{\xi}_I = R(\theta)\dot{\xi}_R = R(\theta) \begin{bmatrix} \dot{x}_R \\ \dot{y}_R \\ \dot{\theta}_R \end{bmatrix} = R(\theta) \begin{bmatrix} v(t) \\ 0 \\ \omega(t) \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v(t) \\ 0 \\ \omega(t) \end{bmatrix}$$

$$\dot{\xi}_R = R^{-1}(\theta)\dot{\xi}_I = R^{-1}(\theta) \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}$$

Summary of differential kinematics eqs. in configuration variables

$$v(t) = \frac{r\dot{\varphi}_R(t) + r\dot{\varphi}_L(t)}{2} = \frac{v_R(t) + v_L(t)}{2}$$

$$\omega(t) = \frac{r\dot{\varphi}_R(t) - r\dot{\varphi}_L(t)}{2\ell} = \frac{v_R(t) - v_L(t)}{2\ell}$$

$$\dot{\xi}_I = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = R(\theta) \begin{bmatrix} v(t) \\ 0 \\ \omega(t) \end{bmatrix} = \begin{cases} R(\theta) \begin{bmatrix} \frac{r\dot{\varphi}_R(t) + r\dot{\varphi}_L(t)}{2} \\ 0 \\ \frac{r\dot{\varphi}_R(t) - r\dot{\varphi}_L(t)}{2\ell} \end{bmatrix} = R(\theta) \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ 0 & 0 \\ \frac{r}{2\ell} & -\frac{r}{2\ell} \end{bmatrix} \begin{bmatrix} \dot{\varphi}_R(t) \\ \dot{\varphi}_L(t) \end{bmatrix} \\ R(\theta) \begin{bmatrix} \frac{v_R(t) + v_L(t)}{2} \\ 0 \\ \frac{v_R(t) - v_L(t)}{2\ell} \end{bmatrix} \end{cases}$$

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

✓ In R robot's frame, $R(\theta)$ disappears

Compute pose velocity for a small robot

A robot is positioned at an angle of 60 degrees with respect to the global reference frame and has wheels with a radius of 1 cm. The wheels are 2 cm from the center of the chassis. If the speeds of wheels 1 and 2, are 4 cm/s and 2 cm/s, respectively, what is the robot velocity with respect to the global reference frame?

$$\theta = \pi / 3$$
$$r = 1$$
$$l = 2$$
$$\dot{\varphi}_1 = 4$$
$$\dot{\varphi}_2 = 2$$

$$\dot{x}_{r1} = \frac{r\dot{\varphi}_1}{2}$$
$$\dot{x}_{r2} = \frac{r\dot{\varphi}_2}{2}$$
$$\omega_1 = \frac{r\dot{\varphi}_1}{2l}$$
$$\omega_2 = -\frac{r\dot{\varphi}_2}{2l}$$

$$\dot{\xi}_I = R(\theta)^{-1}\begin{bmatrix} \dot{x}_{r1} + \dot{x}_{r2} \\ 0 \\ \omega_1 + \omega_2 \end{bmatrix} = \begin{bmatrix} \cos\pi/3 & -\sin\pi/3 & 0 \\ \sin\pi/3 & \cos\pi/3 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 3.0 \\ 0 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 1.5 \\ 2.5981 \\ 0.5 \end{bmatrix}$$

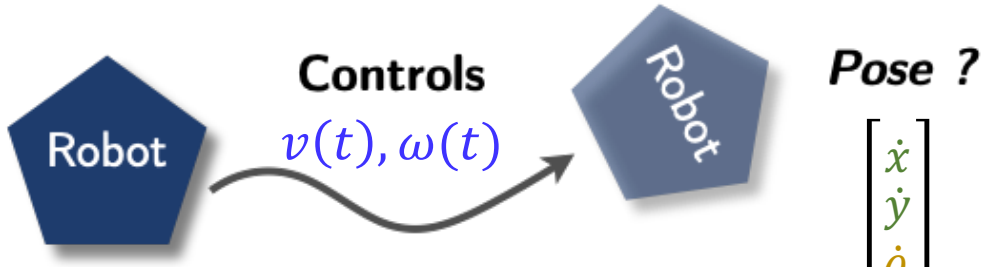
This robot will move instantaneously along the global reference frame x-axis with a speed of 1.5 cm/s and along the y-axis at 2.5981 cm/s while rotating with a speed of 0.5 radians/second.

Pose prediction? → Compute integrals of differential kinematic eqs.

$$\dot{\xi}_I = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v(t)\cos\theta \\ v(t)\sin\theta \\ \omega(t) \end{bmatrix}$$

Note: These equations **hold in general for any robot** that can be assimilated to a *unicycle model*, since they only refer to linear and angular velocities, v and ω, of the origin of the robot's reference frame in I, and to its orientation θ

Posture prediction:



$$x(t) = \int_0^t v(t)\cos(\theta(t))dt$$

$$y(t) = \int_0^t v(t)\sin(\theta(t))dt$$

$$\theta(t) = \int_0^t \omega(t)dt$$

Pose prediction? → Compute integrals of differential kinematic eqs.

$$\dot{\xi}_I = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v(t)\cos\theta \\ v(t)\sin\theta \\ \omega(t) \end{bmatrix}$$

$$x(t) = \int_0^t v(t)\cos(\theta(t))dt$$

$$y(t) = \int_0^t v(t)\sin(\theta(t))dt$$

$$\theta(t) = \int_0^t \omega(t)dt$$

Specifically for a two-wheeled differential robot:

$$x(t) = \frac{1}{2}\int_0^t (v_R(t) + v_L(t))\cos(\theta(t))dt$$

$$x(t) = \frac{1}{2}\int_0^t (v_R(t) + v_L(t))\sin(\theta(t))dt$$

$$\theta(t) = \frac{1}{2\ell}\int_0^t (v_R(t) - v_L(t))dt$$

Pose prediction? → Easy but useful cases

$$x(t) = \frac{1}{2}\int_0^t (v_R(t) + v_L(t))\cos(\theta(t))dt$$

$$x(t) = \frac{1}{2}\int_0^t (v_R(t) + v_L(t))\sin(\theta(t))dt$$

$$\theta(t) = \frac{1}{2\ell}\int_0^t (v_R(t) - v_L(t))dt$$

- Equal and constant forward speed for both wheels
- E.g., useful also in interval-wise changes in common velocity

$$v_L = v_R = v$$

$$x(t) = x_0 + vt\cos(\theta)$$
$$y(t) = y_0 + vt\sin(\theta)$$
$$\theta(t) = \theta_0$$

Robot moves in a straight line

- Equal but opposite wheel speeds

$$v_L = -v_R$$

$$x(t) = x_0$$
$$y(t) = y_0$$
$$\theta(t) = \theta_0 + \frac{2vt}{2\ell}$$

Robot rotates in place

- Constant and different wheel speeds

$$v_L(t) = v_L,\ v_R(t) = v_R,\ v_L \neq v_R$$

$$x(t) = x_0 + \frac{\ell}{2}\frac{v_R + v_L}{v_R - v_L}\sin\left(\tfrac{t}{\ell}(v_R - v_L)\right)$$
$$y(t) = y_0 - \frac{\ell}{2}\frac{v_R + v_L}{v_R - v_L}\cos\left(\tfrac{t}{\ell}(v_R - v_L)\right)$$
$$\theta(t) = \theta_0 + \frac{t}{\ell}(v_R - v_L)$$

Robots moves long a circular trajectory of radius:

$$R = \ell\frac{v_R + v_L}{v_R - v_L}$$

Compute pose for a small robot

A robot is positioned at an angle of 60 degrees with respect to the global reference frame and has wheels with a radius of 1 cm. The wheels are 2 cm from the center of the chassis. If the speeds of wheels 1 and 2, are 4 cm/s and 2 cm/s, respectively, what is the robot velocity with respect to the global reference frame?

+

Velocity is kept constant for 20 seconds, and the robot started at position ${}^{I}(0,1)$. What is the pose of the robot in I at time $t = 20$?

$$\theta = \pi / 3$$
$$r = 1$$
$$l = 2$$
$$\dot{\varphi}_1 = 4$$
$$\dot{\varphi}_2 = 2$$

$$\dot{x}_{r1} = \frac{r\dot{\varphi}_1}{2} = v_L$$
$$\dot{x}_{r2} = \frac{r\dot{\varphi}_2}{2} = v_R$$
$$\omega_1 = \frac{r\dot{\varphi}_1}{2l}$$
$$\omega_2 = -\frac{r\dot{\varphi}_2}{2l}$$

$$v_L(t) = v_L,\ v_R(t) = v_R,\ v_L \neq v_R$$

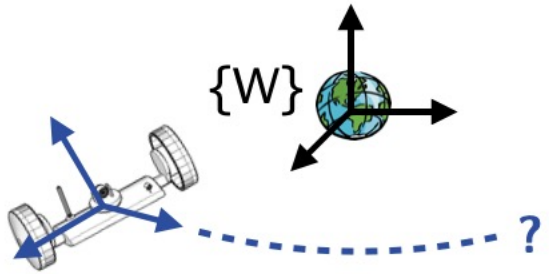
$$x(t) = x_0 + \frac{\ell}{2}\frac{v_R + v_L}{v_R - v_L}\sin\left(\tfrac{t}{\ell}(v_R - v_L)\right)$$

$$y(t) = y_0 - \frac{\ell}{2}\frac{v_R + v_L}{v_R - v_L}\cos\left(\tfrac{t}{\ell}(v_R - v_L)\right)$$

$$\theta(t) = \theta_0 + \frac{t}{\ell}(v_R - v_L)$$

Inverse kinematics?

Given an initial and a goal pose, what are the velocity profiles to provide to the wheels to achieve the desired pose transition?



In the general case, a very hard problem, the presence of the non-holonomic sliding constraints makes computations difficult

Even in the simplest and a bit general case of constant but different velocities the problem looks difficult!

$$v_L(t) = v_L,\ v_R(t) = v_R,\ v_L \neq v_R$$

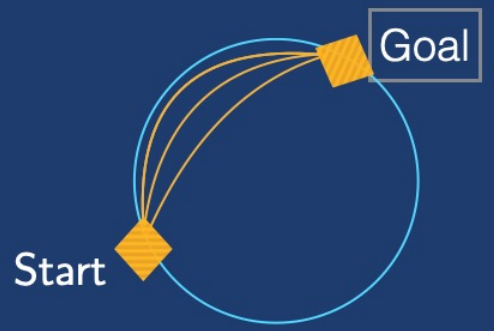
$$x(t) = x_0 + \ell\frac{v_R + v_L}{v_R - v_L}\sin\left(\frac{t}{2\ell}(v_R - v_L)\right)$$

$$y(t) = y_0 - \ell\frac{v_R + v_L}{v_R - v_L}\cos\left(\frac{t}{2\ell}(v_R - v_L)\right)$$

$$\theta(t) = \theta_0 + \frac{t}{2\ell}(v_R - v_L)$$

Given a time t and a goal pose, the equations solve for v_L and v_R but do not provide an independent control for θ

The same final pose(t)
can be achieved in many/infinite ways



Different radii, and/or multiple iterations over the same circular path to meet time requirements

Inverse kinematics? A simplified, piece-wise approach

Easy forward kinematics cases

Rotate in place

$v_L = -v_R$

$$x(t) = x_0$$
$$y(t) = y_0$$
$$\theta(t) = \theta_0 + \frac{2vt}{2\ell}$$

Move straight

$v_L = v_R = v$

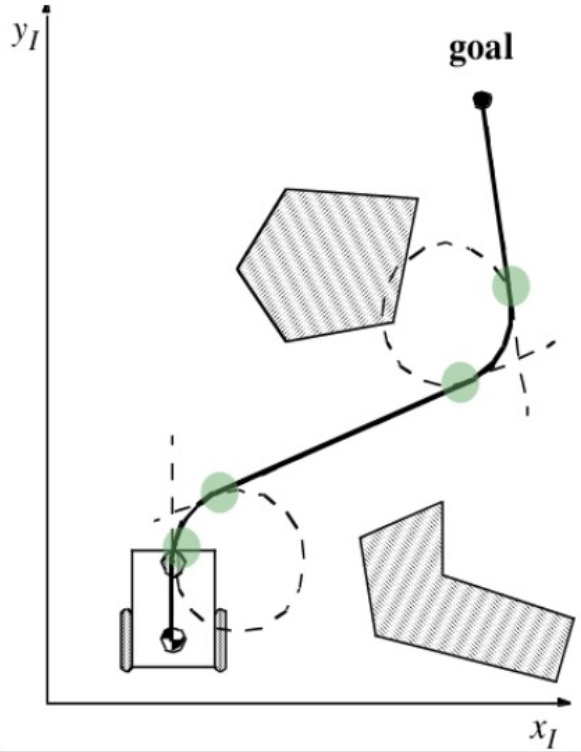
$$x(t) = x_0 + vt\cos(\theta)$$
$$y(t) = y_0 + vt\sin(\theta)$$
$$\theta(t) = \theta_0$$

Solve the problem by ***decomposing the trajectory in primitive motion segments*** (very easy in open space):

- Straight lines
- Segments of a circle or rotation in place

Easier but not easy: a lot of issues to guarantee smoothness (and other quality constraints) and to deal with robot and environments' constraints

Other (better) ways to do it, later on ....



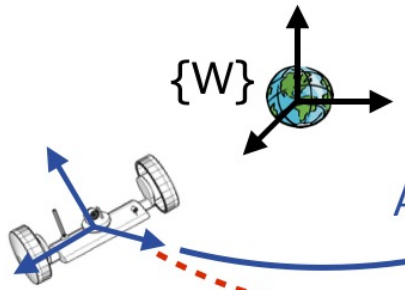
So, should we give up computing the pose? Is kinematics useless?

No!

- Knowing the pose is knowing the state of the robotic system, it's an often a NECESSARY piece of information for any decision-making

What is robot's *pose* in {W} after moving at a velocity **(v,ω)**, for 1 minute?

$\Delta\xi(t)$?

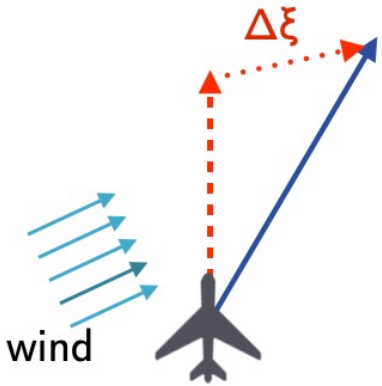


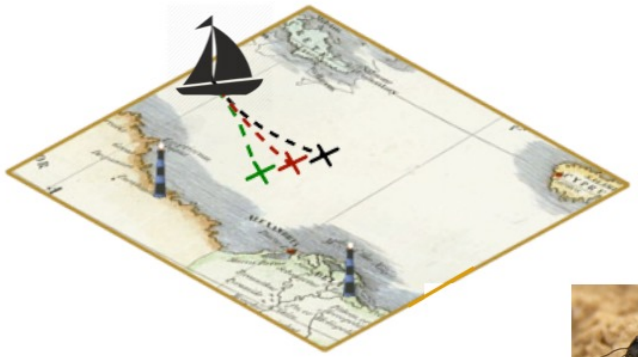
Actual trajectory $\xi(t) = [x(t)\ y(t)\ \theta(t)]$

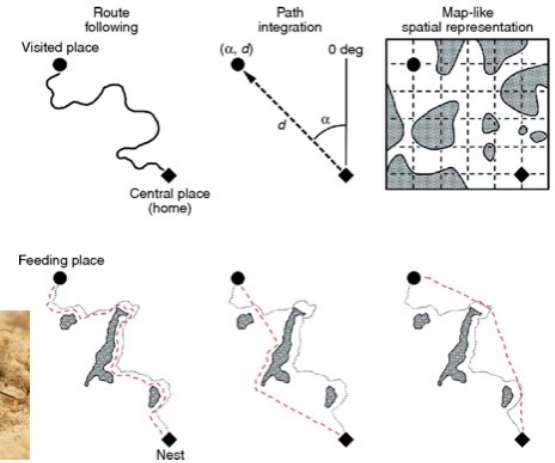
Target trajectory $\xi_d(t) = [x_d(t)\ y_d(t)\ \theta_d(t)]$

Where am I? / What is my pose?

With respect to an initial reference point, a coordinate system, a map ....







Dead (Deduced) Reckoning

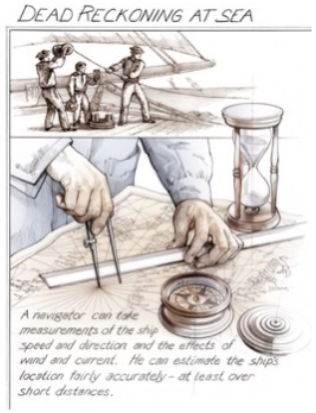
- In absence of an external infrastructure (e.g., GPS + Filters + Cameras) able to **track** the pose of the robot, **numerical integration** can be used, based on:
  - **Kinematic model** of the robot
  - Knowledge of the issued velocity commands, $v(t), \omega(t)$

→ **Incrementally** build the state estimate using on-board information

Dead reckoning: The process of (incrementally, at discrete time steps) determining its own pose based on the knowledge of some reference frame (a *fix*) and the knowledge or the estimate of the velocities (speeds and headings) actuated over time. Data related to exogenous and endogenous disturbances can be included.

Time-integration of velocity vectors estimated through on-board data

Dead Reckoning & Odometry

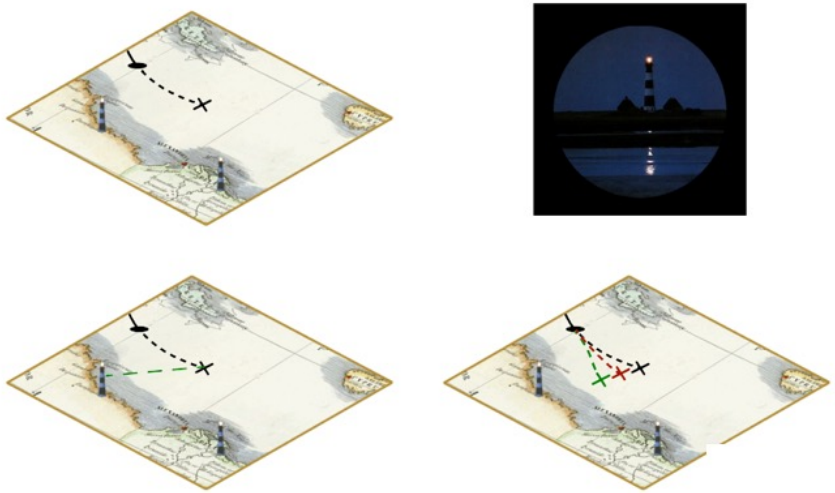


- ***Odometry*** is "basically" another way to say the same thing, that has different roots and etymology ...

- In the animal world, this is called ***path integration***

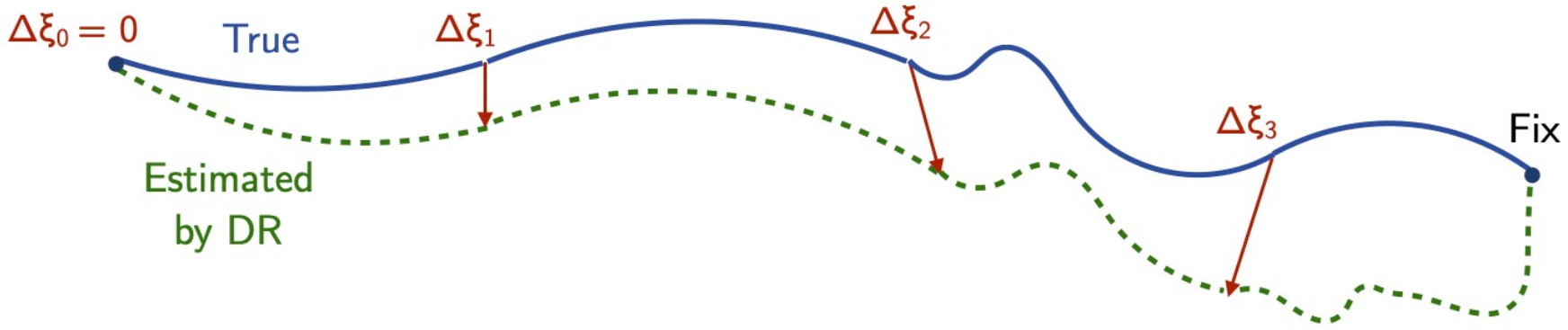
- The uncertainty of dead reckoning / odometry increases over time and maybe over distance ➔ A new fix is intermittently needed to determine a more reliable position from which a new dead reckoning process (i.e., integration) can be restarted

- In navigation, from where the terms comes from, lighthouses and/or celestial observations were used to get a new fix



Dead Reckoning / Odometry: error growing and fixes

1. Robot started in the initial configuration
2. Moved N steps in direction ***x***, and then M steps in direction *y* ...
3. In general, will the new position be determined with high accuracy?



Small/Large discrepancies between issued commands and actual motion, due to friction, imprecision, approximations, ... computations and real world intrinsically bring errors!

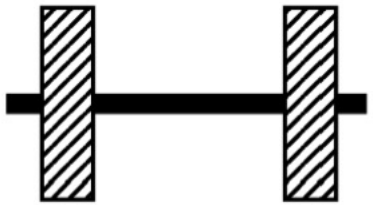
Dead Reckoning / Odometry: causes of error

Odometry drift (error growth) in pose estimate is due to:

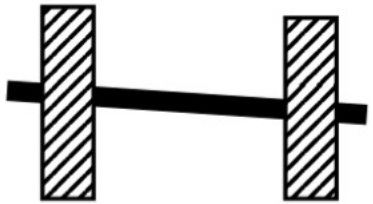
- *Numerical integration* errors (approximations, floating point rounding,....)
- Error readings from the *wheel encoders*
- Wheel *slippage* (friction issues, uneven ground, ...)
- Inaccurate measurement or calibration of *wheel and chassis parameters*, that are reflected in inaccuracies in the results from the kinematic model

- Rotations usually determine more errors than translations (more slippage)
- Systematic / **Deterministic** errors can be corrected by a calibration process
- **Random** / Environment-related errors have to be explicitly modeled, but will inevitably determine uncertain pose estimates
- The additional use of inertial measures (heading, acceleration) can greatly improve accuracy of the whole dead reckoning process

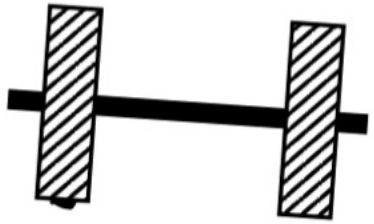
Dead Reckoning / Odometry: issues with wheels



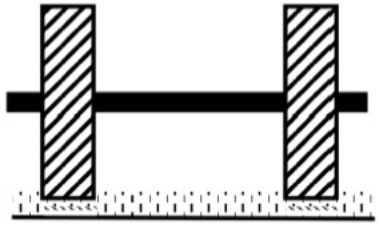
ideal case



different wheel diameters



bump

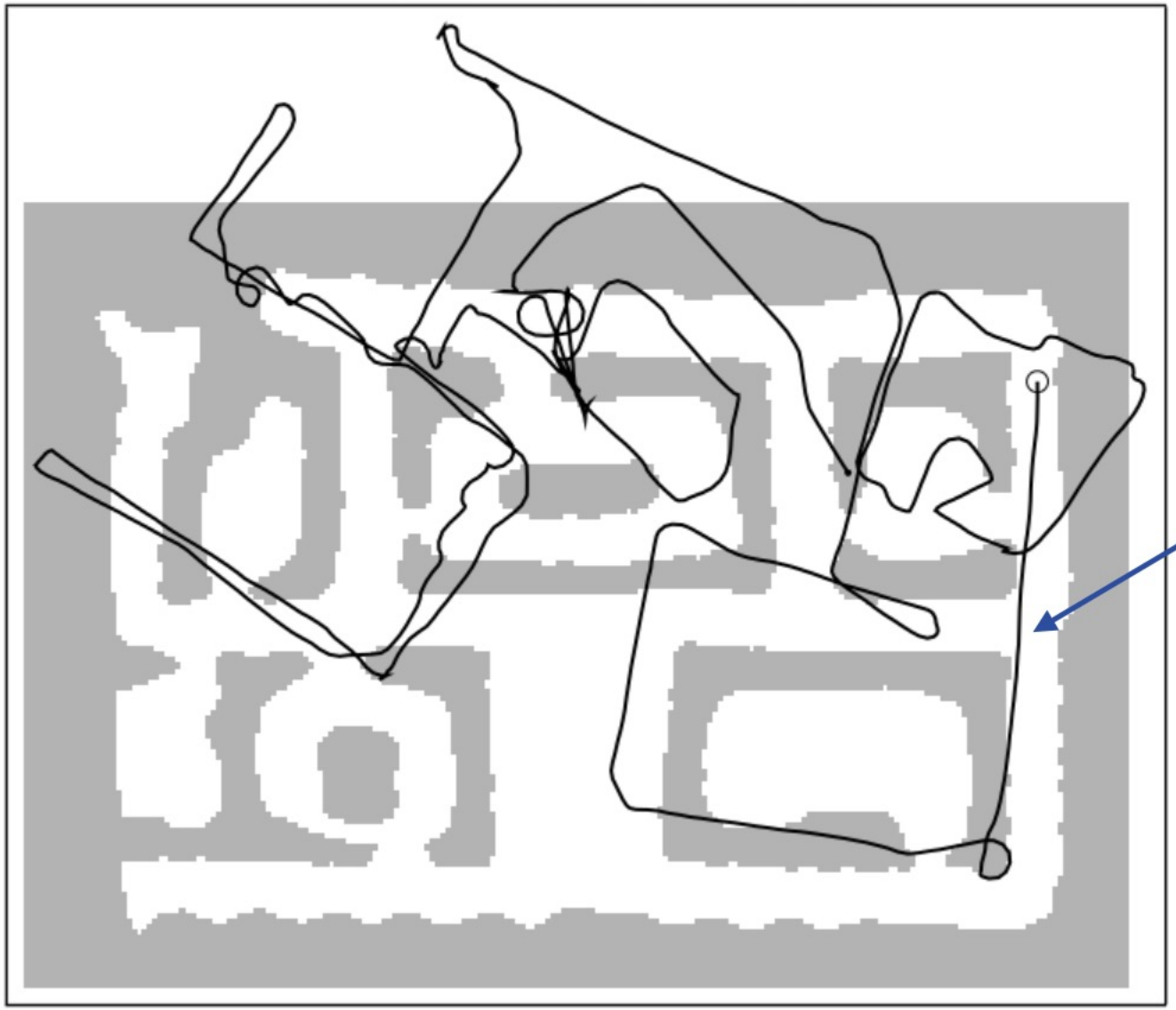


carpet

and many more ...

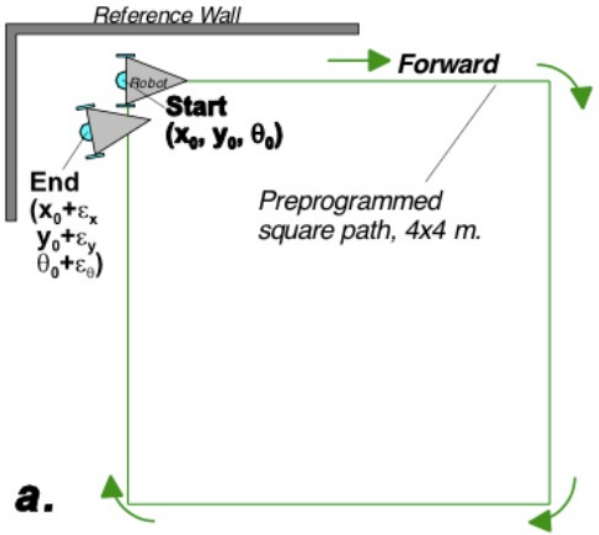
Using wheels' measures to compute odometry is not perfect at all, it comes with a number of *uncertainties*!

A typical odometry-estimated path

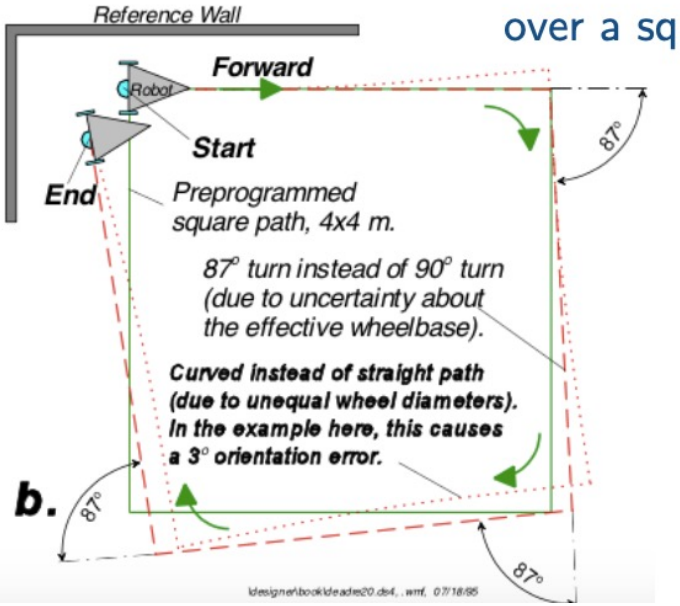


This the path the robot has estimated using odometry measures

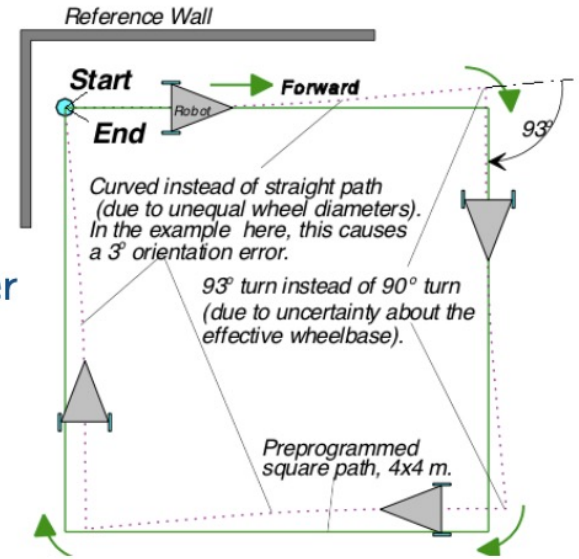
Interesting cases for odometry-based errors



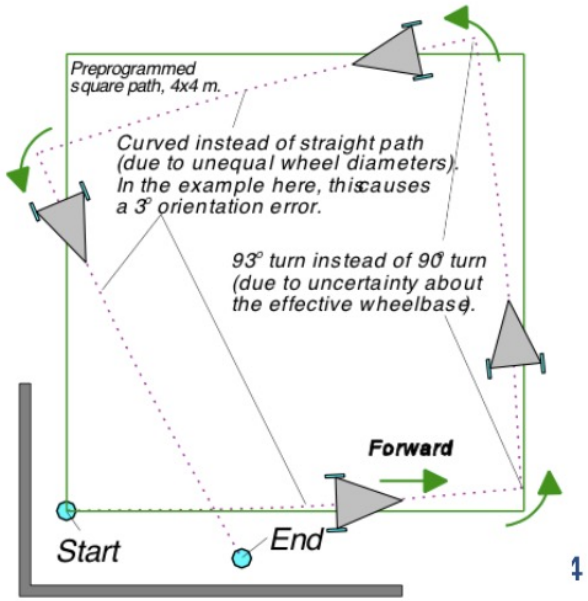
Cumulative error over a squared path



Different errors canceling with each other



Different errors summing up



From continuous-time to discrete-time dynamical system

Dynamics of the robot system

$$\dot{x} = v(t)\cos(\theta(t))$$

$$\dot{y} = v(t)\sin(\theta(t))$$

$$\dot{\theta} = \omega(t)$$

Motion from geometric considerations

$$W\begin{bmatrix} x(t+\delta t) \\ y(t+\delta t) \\ \theta(t+\delta t) \end{bmatrix} = \begin{bmatrix} x(t) + \frac{v(t)}{\omega(t)}\left(\sin(\theta(t) + \Delta\theta(t+\delta t)) - \sin(\theta(t))\right) \\ y(t) - \frac{v(t)}{\omega(t)}\left(\cos(\theta(t) + \Delta\theta(t+\delta t)) - \cos(\theta(t))\right) \\ \theta(t) + \omega(t)\delta t \end{bmatrix}$$

- Transform the continuous kinematic equations into a ***discrete-time system***
- Numeric integration is therefore a viable option can be also done online
- Useful to numerically predict future poses
- Useful to keep updating current pose / state: *Where am I? (*In the environment, on the trajectory ...)

Assumptions for numerical integration

- ***Time is discretized*** in short intervals of length Δt
- When used online, a numeric integration is performed at each time step
- During a time interval $[t_k, t_{k+1}]$, the **velocity inputs v_k and ω_k are assumed to be constant** and the robot moves along a circle of radius v_k / ω_k centered at the ICR(t), or moves along a segment if $\omega_k = 0$
- At step *k*, robot's pose ξ_k and its velocities are assumed to be *known* and are used to compute ξ_{k+1} by integration of the kinematic model over $\Delta t_k = [t_k, t_{k+1}]$

Three main approaches

Euler | Runge-Kutta | Exact

Euler approach: use initial orientation

$$\dot{x} = v(t)\cos(\theta(t))$$

$$\dot{y} = v(t)\sin(\theta(t))$$

$$\dot{\theta} = \omega(t)$$

$$\begin{cases} x_{k+1} = x_k + \Delta t_k v_k \cos(\theta_k) \\ y_{k+1} = y_k + \Delta t_k v_k \sin(\theta_k) \\ \theta_{k+1} = \theta_k + \Delta t_k \omega_k \end{cases}$$

x_{k+1} and y_{k+1} are **approximate**, θ_{k+1} is **exact**

E.g., for x component

$$\dot{x} = v(t)\cos(\theta(t))$$

$$\dot{x} \approx \frac{x(t+\Delta t) - x(t)}{\Delta t}$$

$$x(t+\Delta t) \approx x(t) + \Delta t\, v(t)\cos(\theta(t)) = x(t) + \Delta t\, \dot{x}$$

Limitation: During the time interval Δt_k, the orientation changes because of the issued velocity commands, but this is not considered in the calculation of sin and cos, since θ_k is kept as at the beginning of the interval

Euler approach: use initial orientation

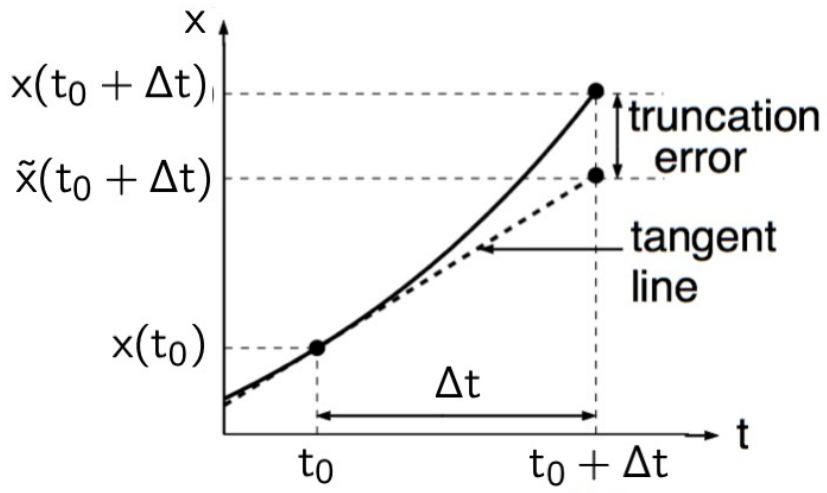
$$\dot{x} = v(t)\cos(\theta(t))$$
$$\dot{y} = v(t)\sin(\theta(t))$$
$$\dot{\theta} = \omega(t)$$

$$\dot{x} = v(t)\cos(\theta(t))$$
$$\dot{x} \approx \frac{x(t+\Delta t) - x(t)}{\Delta t}$$
$$x(t+\Delta t) \approx x(t) + \Delta t\, v(t)\cos(\theta(t)) = x(t) + \Delta t\, \dot{x}$$

Euler method: first term of Taylor series, 1st order approximation

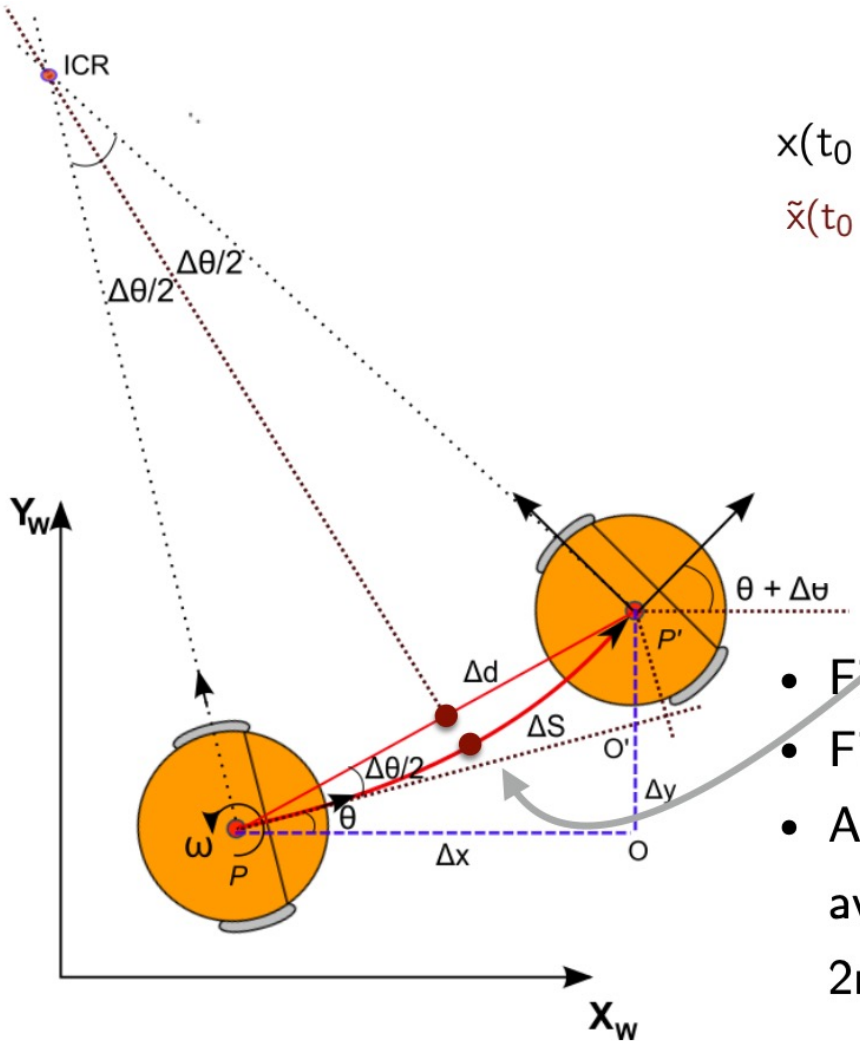
$$x(t_0+\Delta t) = x(t_0) + \Delta t\, \dot{x}(t_0) + \frac{(\Delta t)^2}{2!}\ddot{x}(t_0) + \frac{(\Delta t)^3}{3!}\dddot{x}(t_0) + \ldots$$

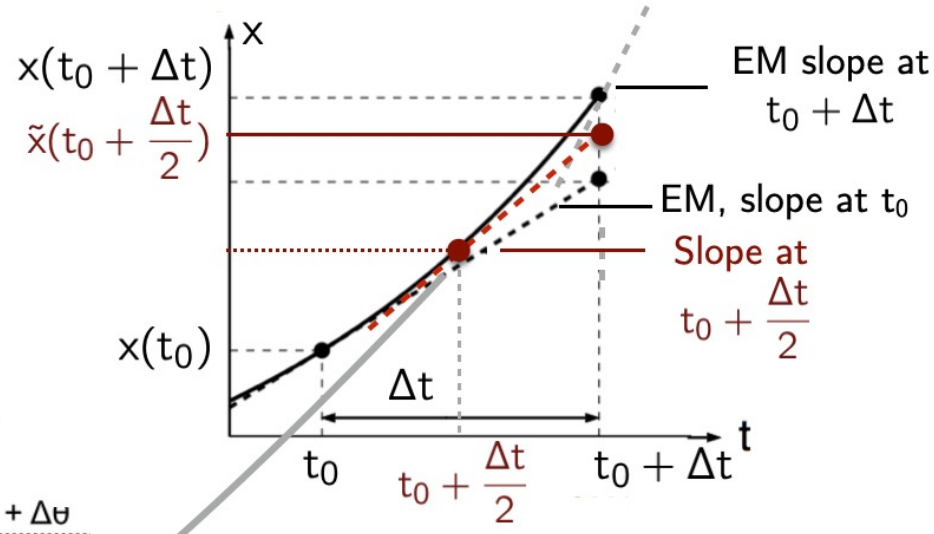
$$\tilde{x}(t_0+\Delta t) = x(t_0) + \Delta t\, \dot{x}(t_0)$$



- Approximate the arc trajectory with the tangent in the initial point

Runge-Kutta: Average orientation





- Find slope s_1 (1st order approx) in t_0
- Find slope s_2 (1st order approx) in $t_0 + \Delta t$
- Approximate the value in $t_0 + \Delta t$ with the average slope between t_0 and $t_0 + \Delta t$ → 2nd order approx, Taylor in $t_0 + \Delta t/2$

$$\tilde{x}(t_0 + \Delta t) = x(t_0) + \Delta t \frac{s_1 + s_2}{2} = x(t_0) + \Delta t \left(\frac{\dot{x}(t_0) + \dot{x}(t_0 + \Delta t)}{2} \right)$$

- Slope (derivative) at mid-point of the linearly approximated arc → **Average slope**

$$\theta_k + \frac{(\theta_{k+1} - \theta_k)}{2} \rightarrow \theta_k + \frac{\Delta\theta_k}{2}$$

Runge-Kutta: Average orientation

$$\begin{cases} x_{k+1} = x_k + v_k \Delta t_k \cos(\theta_k + \frac{\Delta\theta_k}{2}) \\ y_{k+1} = y_k + v_k \Delta t_k \sin(\theta_k + \frac{\Delta\theta_k}{2}) \\ \theta_{k+1} = \theta_k + \omega_k \Delta t_k \end{cases}$$

- The average orientation over the integration interval is considered, therefore the x_{k+1} and y_{k+1} values are better than in the Euler case, when only the initial orientation is taken, but still they are **approximations**, while θ_{k+1} is exact
- The value of $\Delta\theta_k$ can be computed directly from the issued velocity commands (v, ω), or can be estimated from measures (from wheels' encoders)

Exact approach: Variation of the orientation

$$W\begin{bmatrix} x(t+\delta t) \\ y(t+\delta t) \\ \theta(t+\delta t) \end{bmatrix} = \begin{bmatrix} x(t) + \frac{v(t)}{\omega(t)}\Big(\sin(\theta(t) + \Delta\theta(t+\delta t)) - \sin(\theta(t))\Big) \\ y(t) - \frac{v(t)}{\omega(t)}\Big(\cos(\theta(t) + \Delta\theta(t+\delta t)) - \cos(\theta(t))\Big) \\ \theta(t) + \omega(t)\delta t \end{bmatrix}$$

$$x_{k+1} = x_k + \frac{v_k}{\omega_k}(\sin\theta_{k+1} - \sin\theta_k)$$

$$y_{k+1} = y_k - \frac{v_k}{\omega_k}(\cos\theta_{k+1} - \cos\theta_k)$$

$$\theta_{k+1} = \theta_k + \omega_k \Delta t_k$$

$\omega = 0$ is a condition that must be monitored, also when ω is close to 0 some practical precaution is necessary in the code

Summary + Comparison among the three methods

$$\xi_{k+1} = \begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_k + \frac{v_k}{\omega_k}(\sin\theta_{k+1} - \sin\theta_k) \\ y_k - \frac{v_k}{\omega_k}(\cos\theta_{k+1} - \cos\theta_k) \\ \theta_k + \omega_k \Delta t_k \end{bmatrix}$$

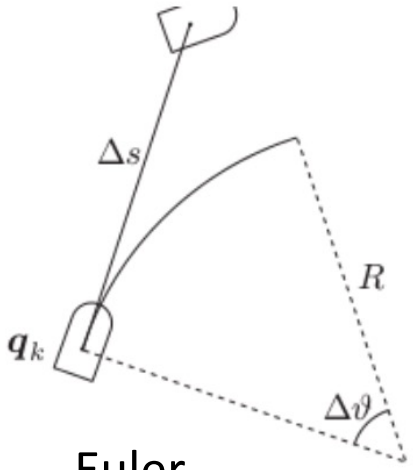
Exact numeric integration

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_k + v_k \Delta t_k \cos\theta_k \\ y_k + v_k \Delta t_k \sin\theta_k \\ \theta_k + \omega_k \Delta t_k \end{bmatrix}$$

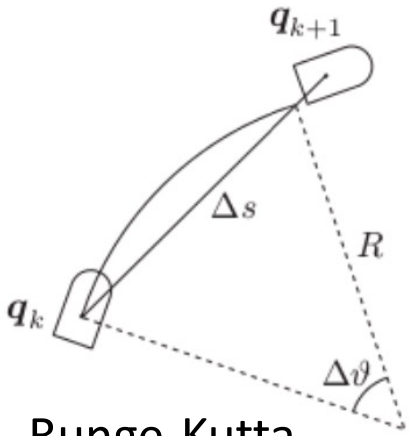
Euler numeric integration

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_k + v_k \Delta t_k \cos\left(\theta_k + \frac{\Delta\theta_k}{2}\right) \\ y_k + v_k \Delta t_k \sin\left(\theta_k + \frac{\Delta\theta_k}{2}\right) \\ \theta_k + \omega_k \Delta t_k \end{bmatrix}$$

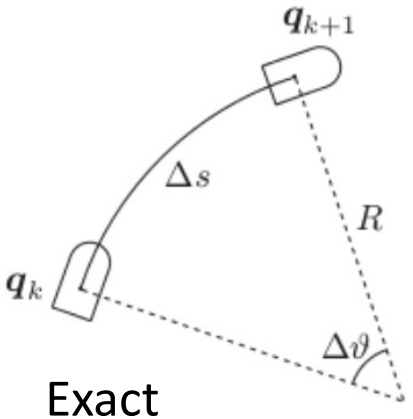
Runge-Kutta numeric integration



Euler



Runge-Kutta



Exact