16-311-Q Introduction to Robotics

Lecture 8: Kinematics Equations Odometry, Dead Reckoning

Instructor:
Gianni A. Di Caro

FORWARD KINEMATICS EQUATIONS

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

Function of the ICR

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

Function of the issued velocities

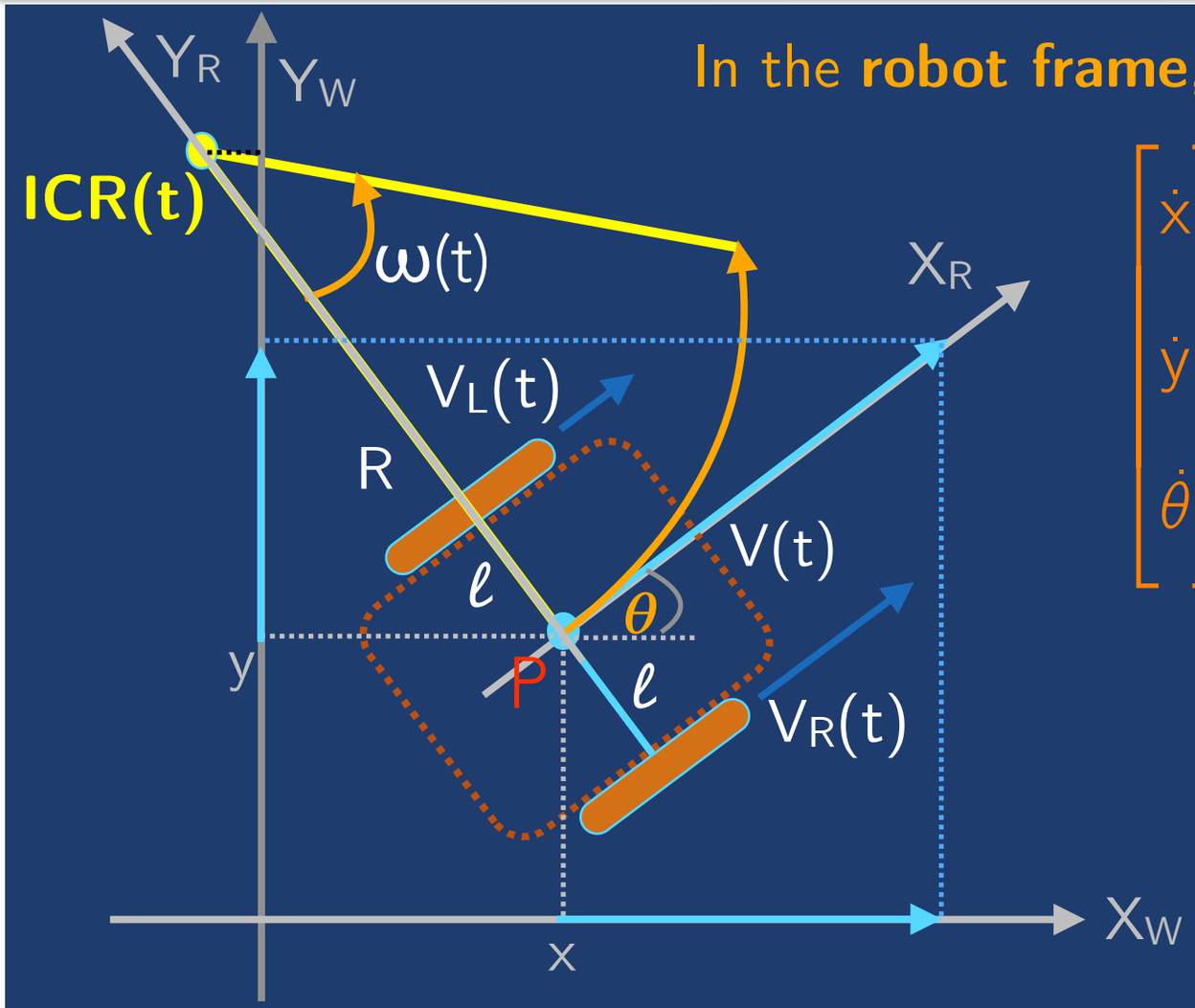
To obtain ***future poses over time-extended intervals,*** it is necessary to provide initial conditions, specify geometry parameters, assign the linear and angular velocity profiles v(t) and ω(t), and ***integrate over time*** (which might not be obvious/easy)

In the specific case of a ***two-wheeled differential robot***, v(t) and ω(t) at the reference point P on the chassis are functions of the Left and Right speeds issued to the Left and Right wheel, respectively:

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

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

FORWARD KINEMATICS FOR DIFFERENTIAL DRIVE



In the **robot frame**, given the spinning velocity controls for wheels

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}_R = \begin{bmatrix} \frac{r\dot{\varphi}_R(t) + r\dot{\varphi}_L(t)}{2} \\ 0 \\ \frac{r\dot{\varphi}_R(t) - r\dot{\varphi}_L(t)}{2\ell} \end{bmatrix} = \begin{bmatrix} \frac{v_R(t) + v_L(t)}{2} \\ 0 \\ \frac{v_R(t) - v_L(t)}{2\ell} \end{bmatrix}$$

Two control inputs, linear and angular velocity

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}_R = \begin{bmatrix} v(t) \\ 0 \\ \omega(t) \end{bmatrix}$$

In the **world reference frame**, given the local inputs v(t), ω(t)

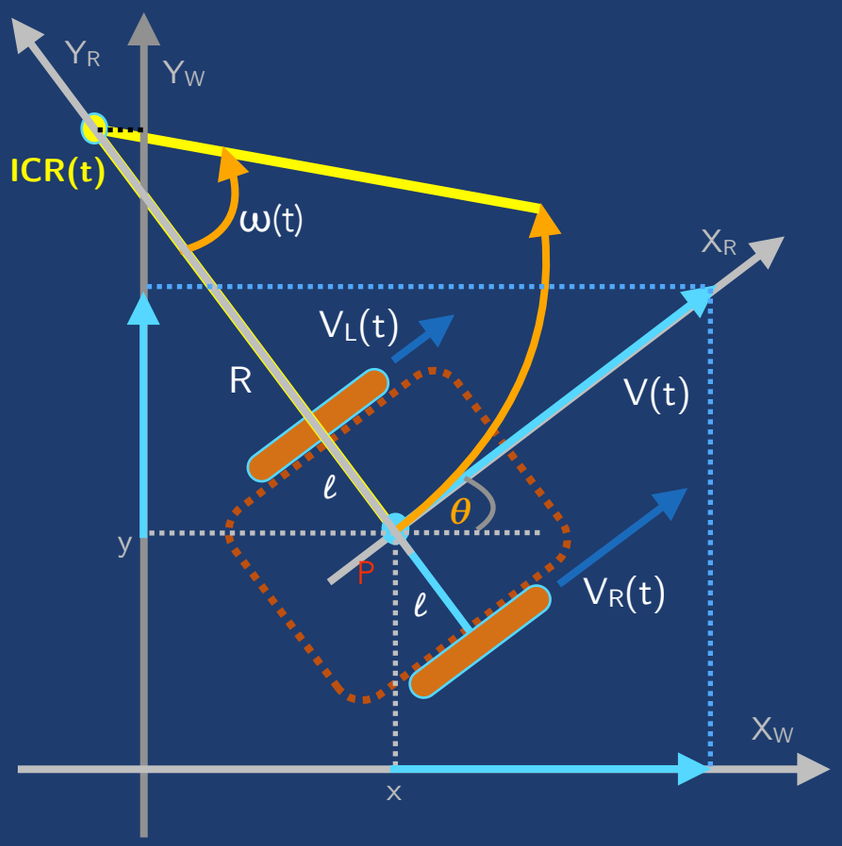
$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}_W = \begin{bmatrix} \cos(\theta(t)) & -\sin(\theta(t)) & 0 \\ \sin(\theta(t)) & \cos(\theta(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v(t) \\ 0 \\ \omega(t) \end{bmatrix}_R$$

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

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

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

A GENERAL MODEL: THE UNICYCLE MODEL



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

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

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

$$\dot{x} = \frac{v_R(t) + v_L(t)}{2}\cos(\theta(t))$$

$$\dot{y} = \frac{v_R(t) + v_L(t)}{2}\sin(\theta(t))$$

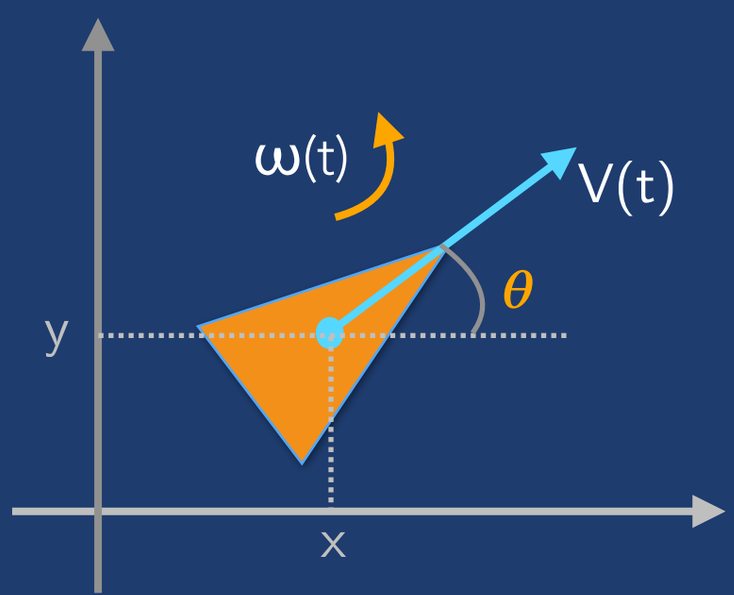
$$\dot{\theta} = \frac{v_R(t) - v_L(t)}{2\ell}$$

$$\dot{x} = \frac{r\dot{\varphi}_R(t) + r\dot{\varphi}_L(t)}{2}\cos(\theta(t))$$

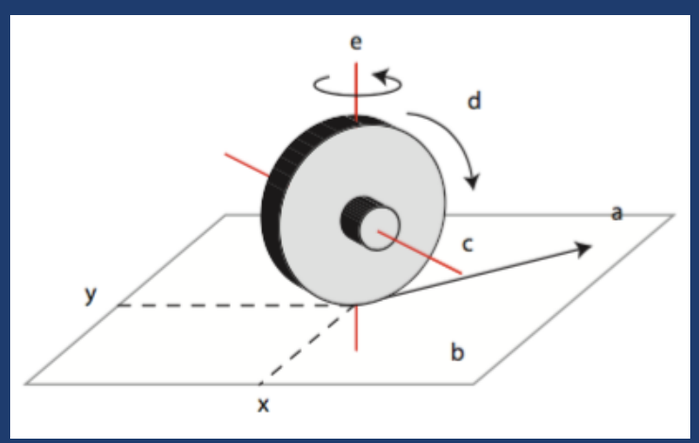
$$\dot{y} = \frac{r\dot{\varphi}_R(t) + r\dot{\varphi}_L(t)}{2}\sin(\theta(t))$$

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

Unicyle model (non holonomic)



Abstract model!



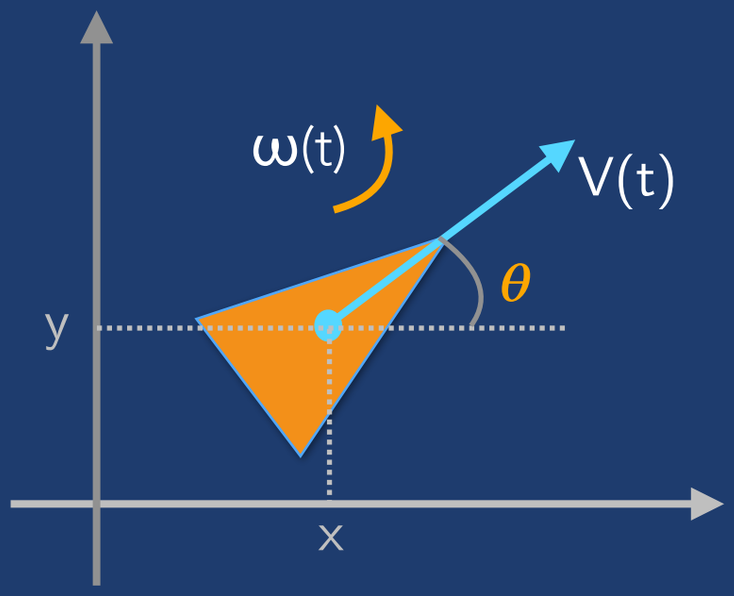
FULL FORWARD KINEMATICS: TIME TRAJECTORY

For a generic robot, given v(t), ω(t) as local inputs, the velocity of pose change in the world reference frame:

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

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

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



If the time-profiles of the velocities are known, the equations can be integrated over time to predict the ***time trajectory:***

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

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

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

For a 2-wheeled differential robot

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

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

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

COMPUTATION OF RATE OF CHANGE IN THE POSE VECTOR

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.

FORWARD KINEMATICS: EASY (BUT USEFUL) CASES

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

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

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

Equal (constant) forward speed for both wheels

$$v_L = v_R = v$$

$$x(t) = x_0 + vt\cos(\theta)$$

$$y(t) = y_0 + vt\sin(\theta)$$

$$\theta(t) = \theta_0$$

Also for interval-wise changes in the common velocity

The robot moves along a straight trajectory

Equal but opposite wheel speeds

$$v_L = -v_R$$

$$x(t) = x_0$$

$$y(t) = y_0$$

$$\theta(t) = \theta_0 + \frac{2vt}{2\ell}$$

The robot rotates in place

Constant (different) speeds for both wheels

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

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

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

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

The robot moves along a circular trajectory of constant radius R

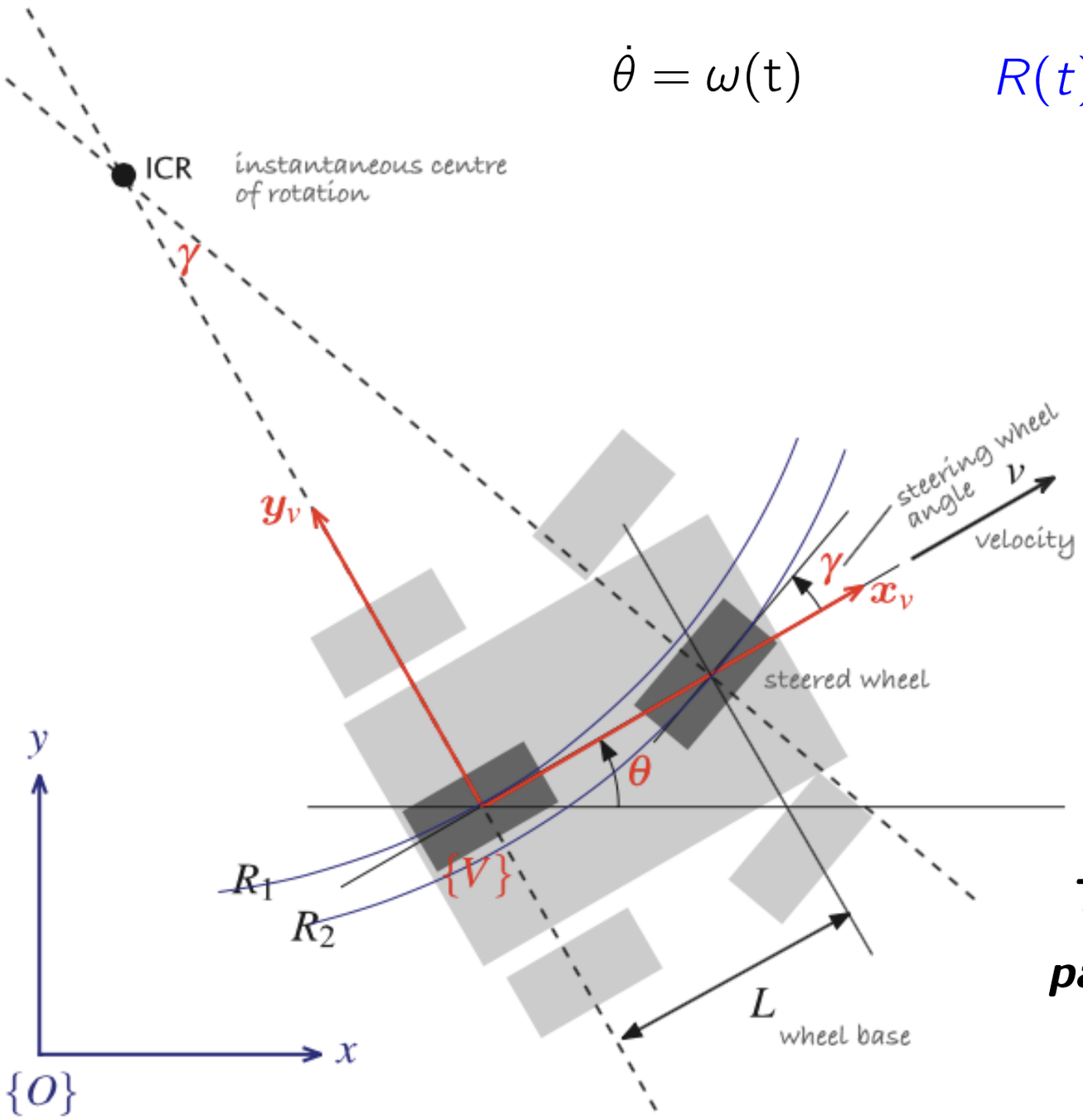
$$R = \ell\frac{v_R + v_L}{v_R - v_L}$$

KINEMATICS EQUATIONS FOR THE BICYCLE MODEL

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

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

$$\dot{\theta} = \omega(t) \qquad R(t) = R_1(t) = \frac{L}{\tan(\gamma(t))}, \quad \omega(t) = \frac{v(t)}{R(t)}$$



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

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

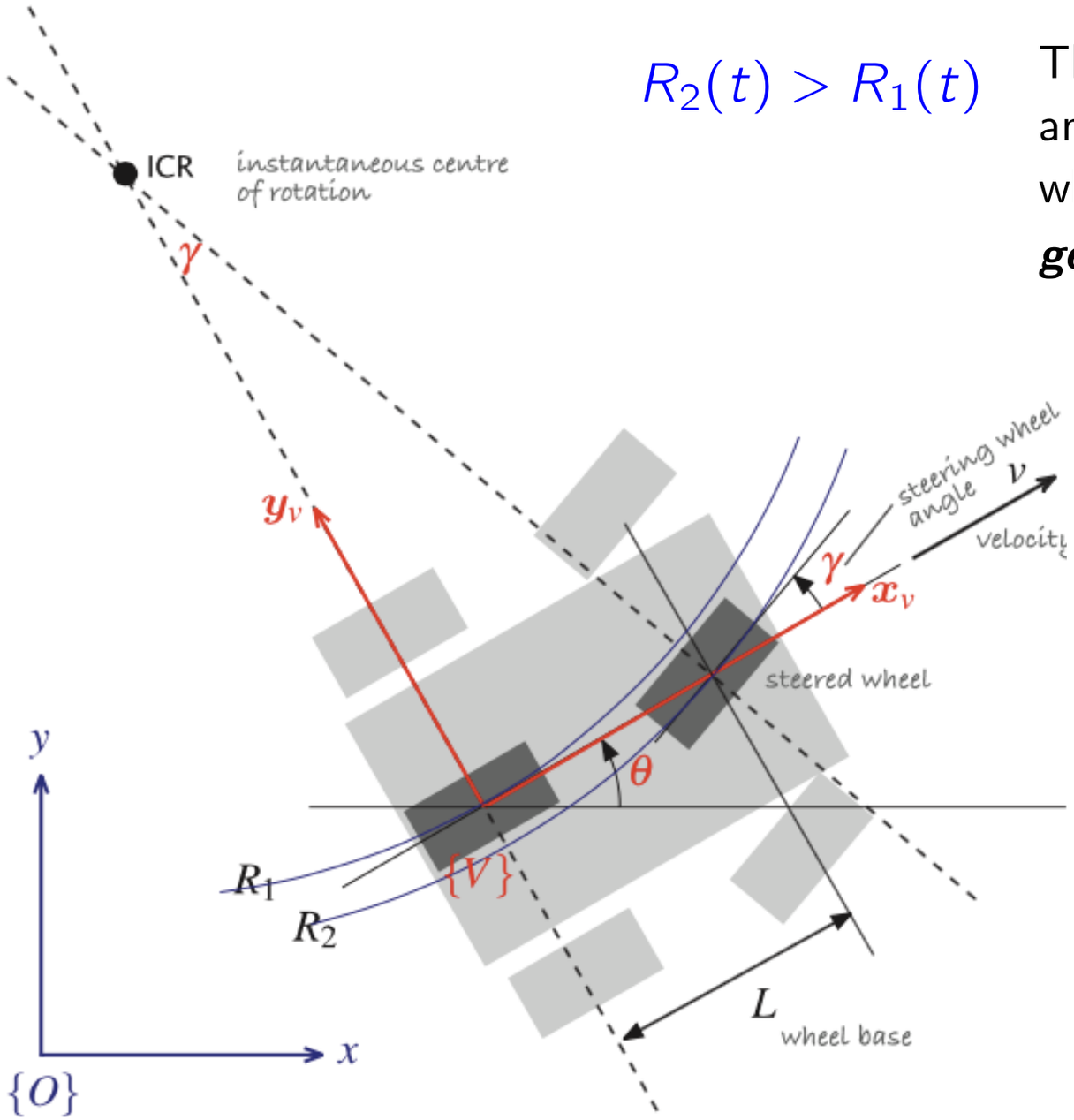
$$\dot{\theta} = \frac{v(t)}{L}\tan(\gamma(t))$$

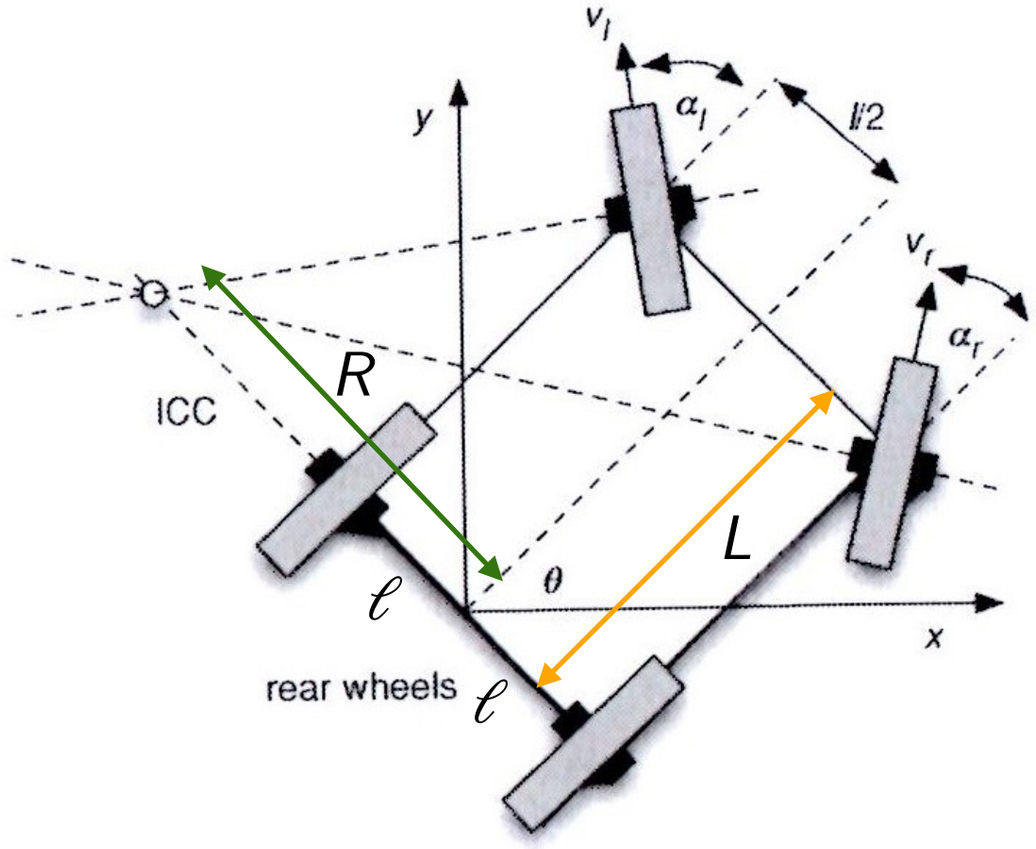
The max value of γ limits maneuverability: parking problem, complex inverse kinematics

ACKERMANN STEERING

$$R_2(t) > R_1(t)$$

The front wheel must follow a longer path, and therefore must rotate faster than the rear wheel. With two front wheels a ***differential gear*** is necessary to implement this difference





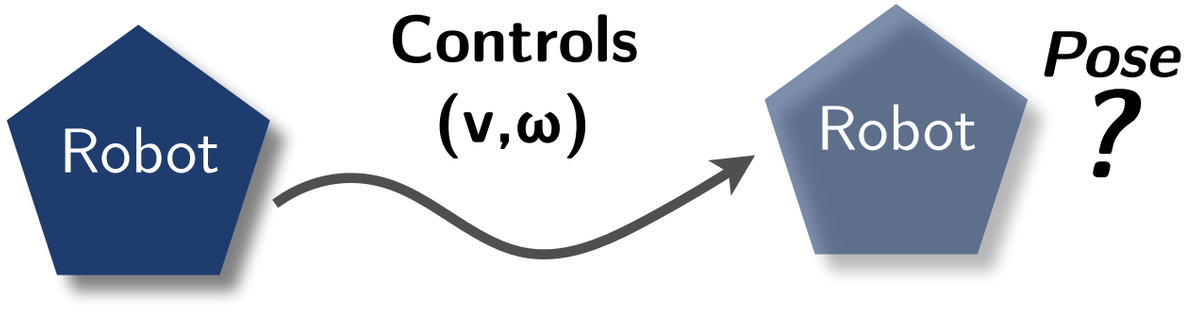
Once set the steering for the left wheel, the right wheel is constrained by rolling motion to steer a specific angle which is coherent with the vehicle's ICR

$$R(t) - \ell = \frac{L}{\tan(\alpha_l(t))}$$

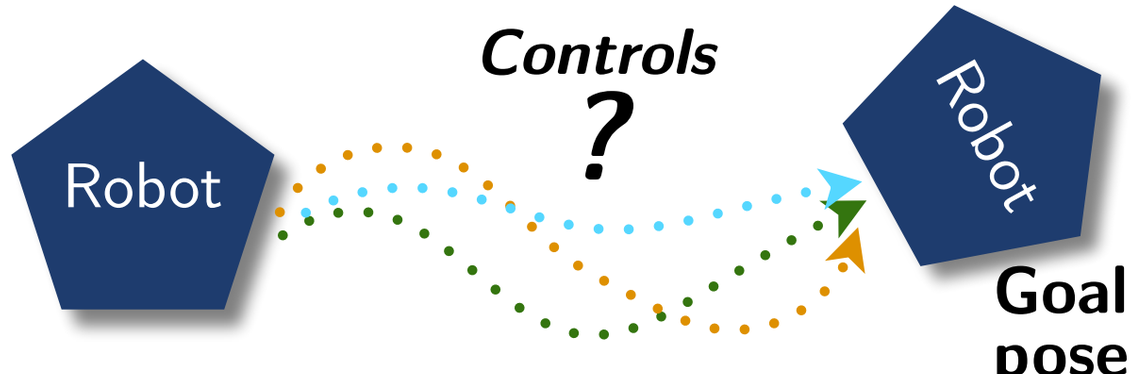
$$R(t) + \ell = \frac{L}{\tan(\alpha_r(t))}$$

FORWARD VS. INVERSE KINEMATICS

Posture prediction: *Forward Kinematics*



Posture regulation: *Inverse Kinematics*



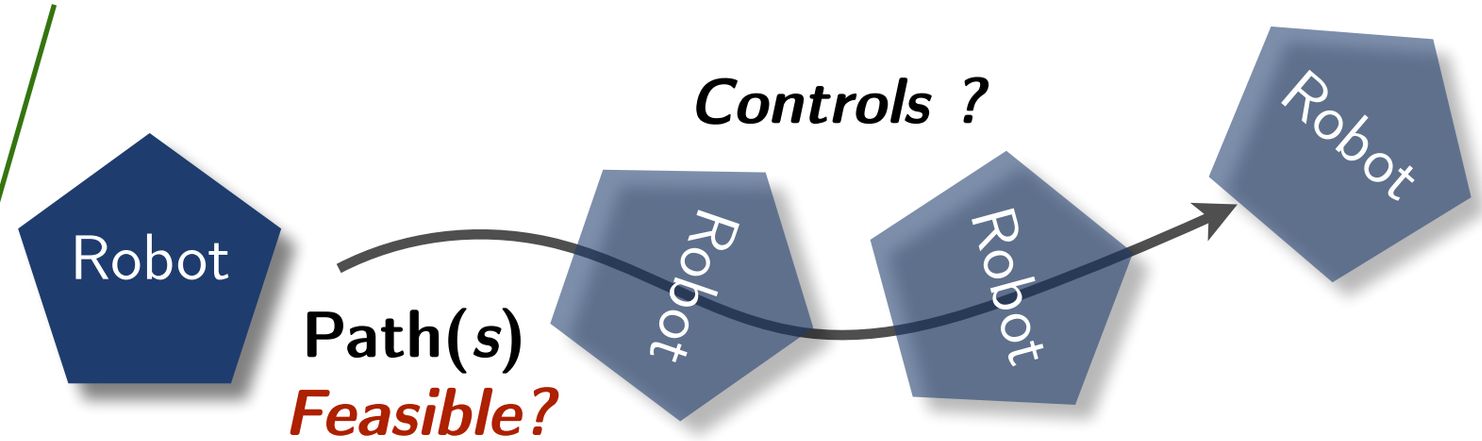
Path *Planning*

Path following (geometry)

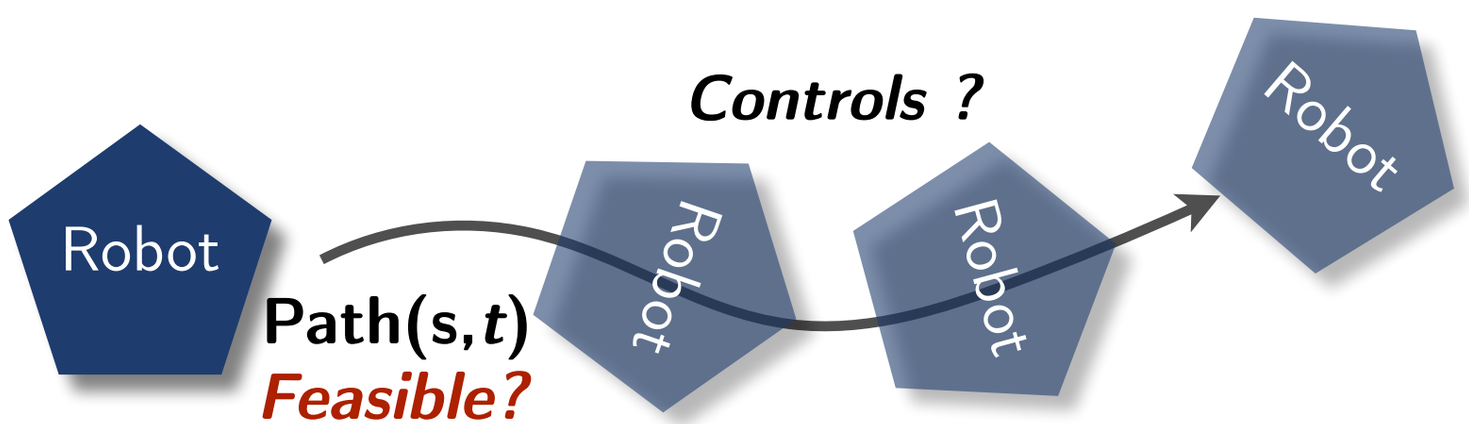
Trajectory *Planning*

Trajectory following (kinematics, time)

Control

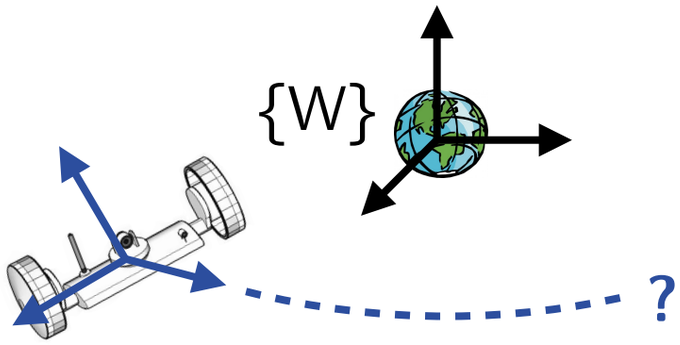


Feasible?



INVERSE KINEMATICS (FOR DIFFERENTIAL ROBOTS)

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

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

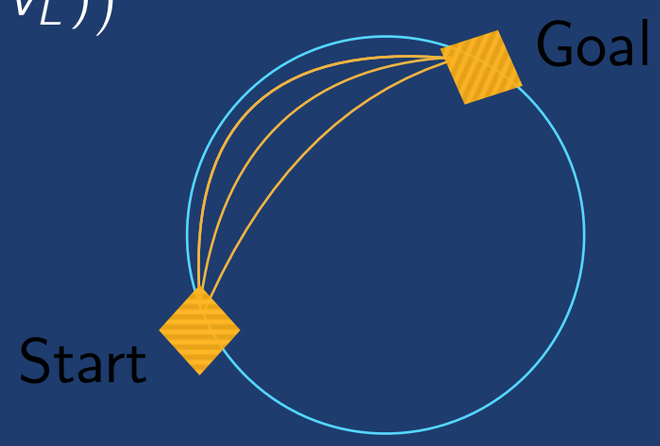
$$x(t) = x_0 + \frac{\ell}{2}\frac{v_R + v_L}{v_R - v_L}\sin\left(\tfrac{t}{\ell}(v_R - v_L)\right)$$

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

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

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 ($\delta_m = 2$)

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

SIMPLIFIED INVERSE KINEMATICS APPROACH

In general, a forward solution is already quite complex, a direct approach to inversion would be problematic for general relations between the two control velocities

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 ($v_L = v_R = v$)

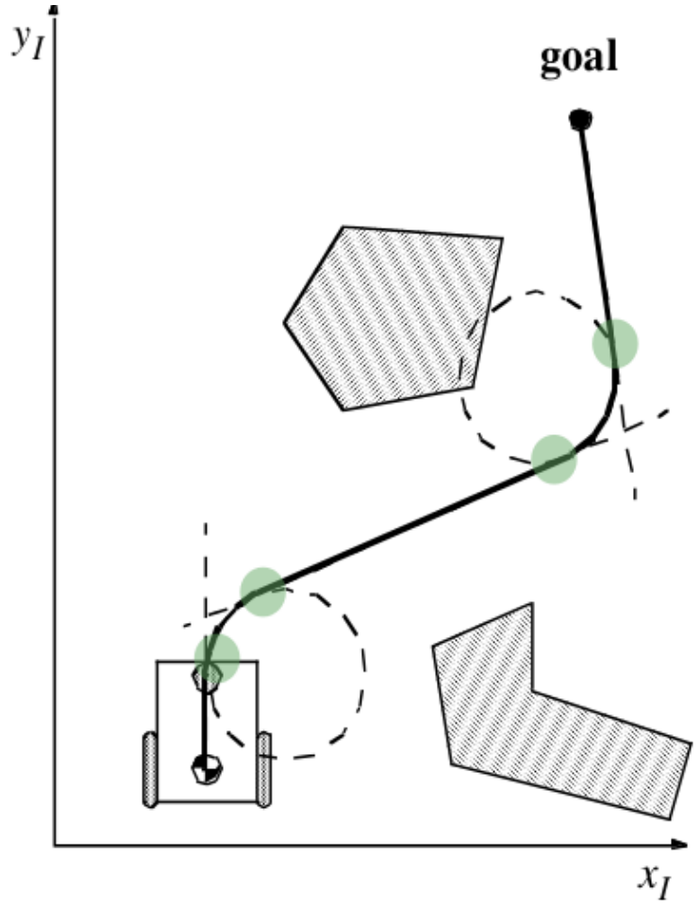
$$x(t) = x_0 + vt\cos(\theta)$$
$$y(t) = y_0 + vt\sin(\theta)$$
$$\theta(t) = \theta_0$$

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

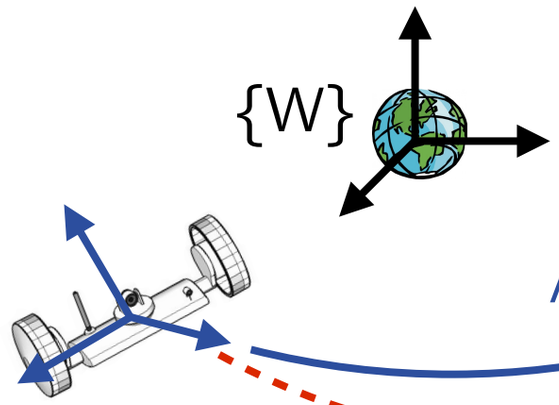
Other (better) ways to do it, later on ....



COMPUTING THE *STATE / POSE*

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

$\Delta\xi(t)$?

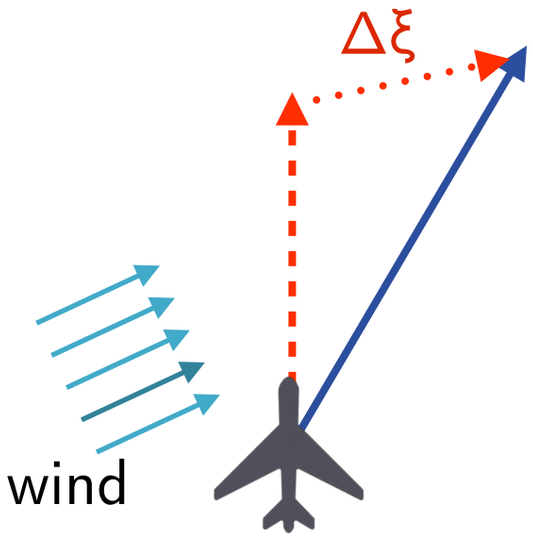


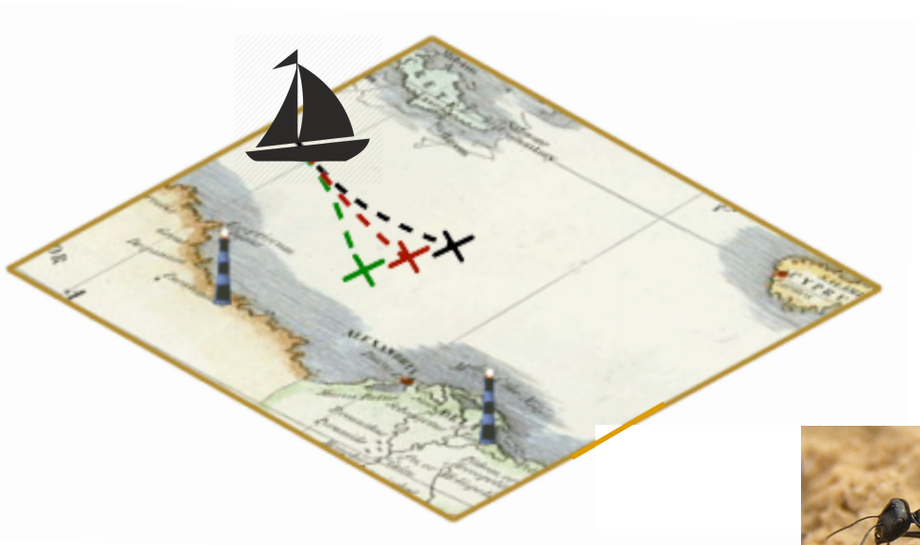
Actual trajectory $\xi(t) = [x(t)\ y(t)\ \theta(t)]$

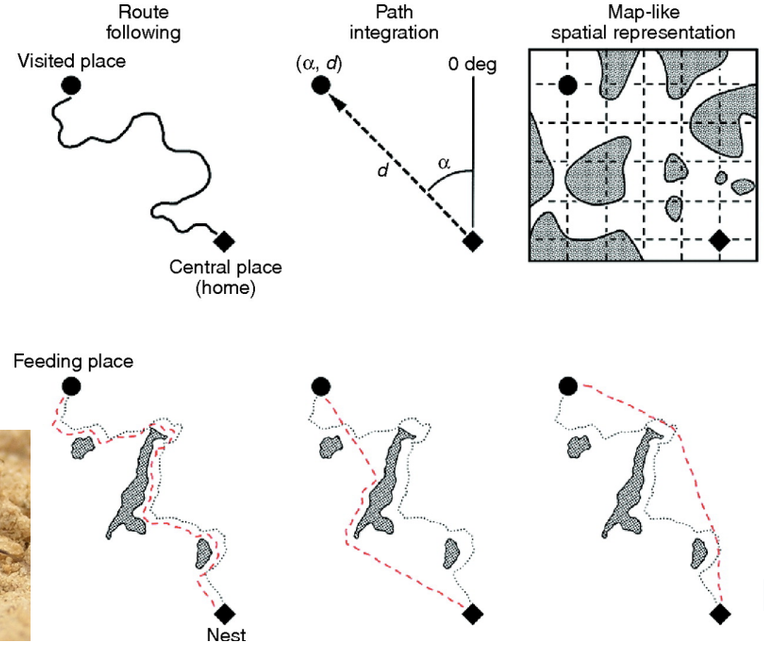
Target trajectory $\xi_d(t) = [x_d(t)\ y_d(t)\ \theta_d(t)]$

Where am I? / What is my pose?

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



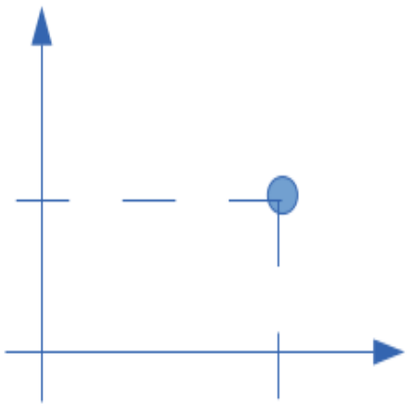




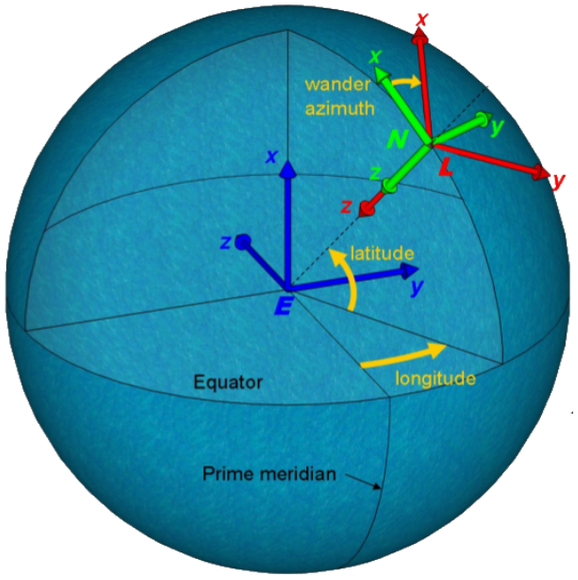
WHAT IS THE *TYPE* OF A POSE?

Pose / Position should be thought in a quite *general sense.*
The required form of a position depends on
the type of the **reference system that is of interest / available**

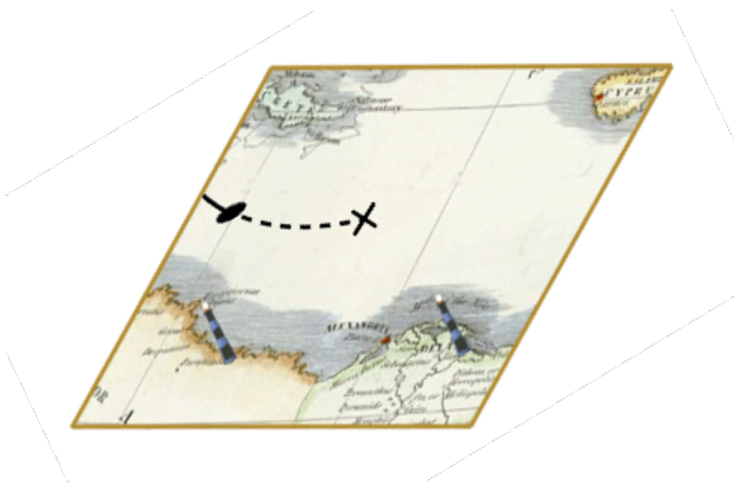
Cartesian, 2D

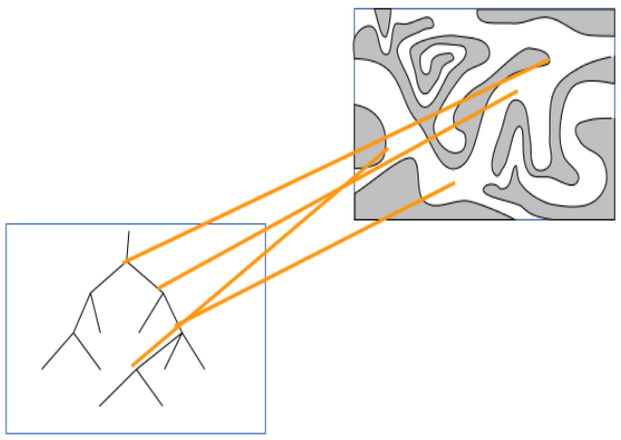


3D earth coordinates

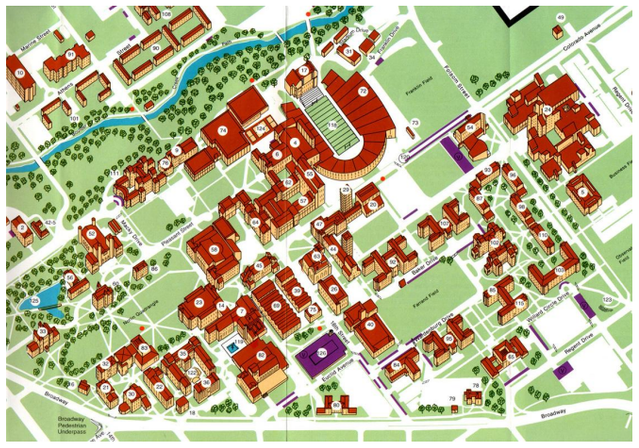


Geographical map, landmarks

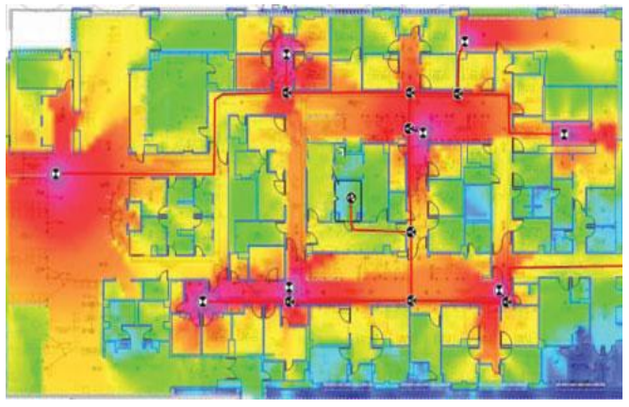




Toplogical map



Metric map



Sensor map

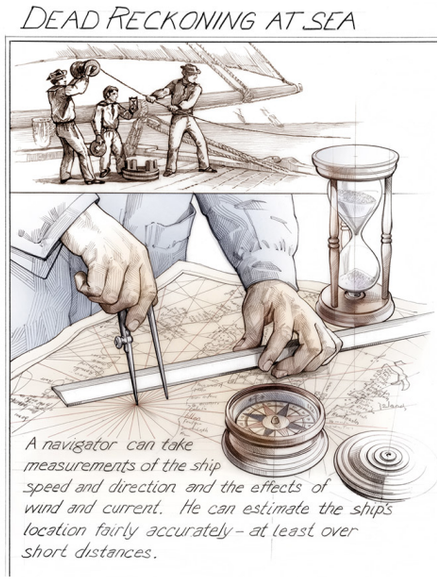
DEAD (DEDUCED) RECKONING

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 the kinematic model of the robot and on the knowledge of the issued velocity commands, $[v(t)\ \omega(t)]^T$

➔ Incrementally build the state using on-board information

Deduced reckoning: The process of (incrementally, at discrete time steps) determining its own position based on the knowledge of some reference point (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

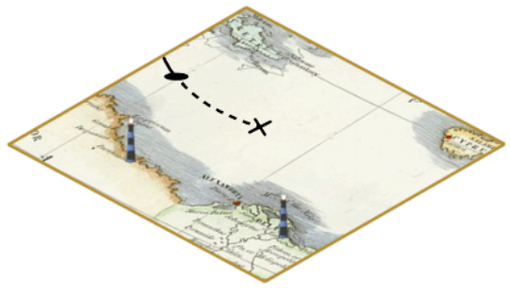


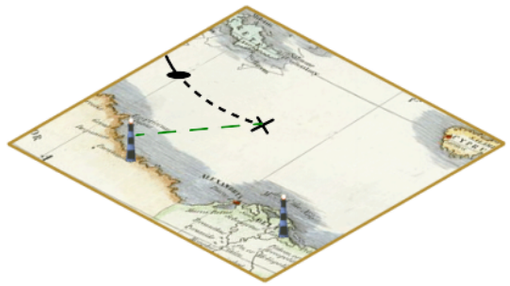
- ***Odometry*** is "basically" another way to say the same thing, that has different roots and etymology ...
- In the animal world, this is called ***path integration***

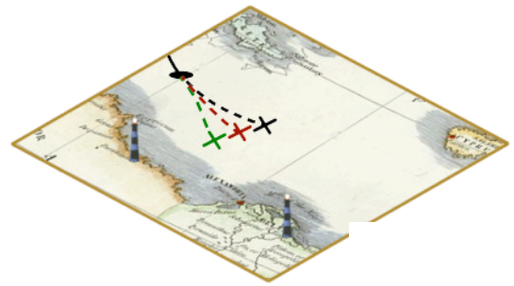
ERROR ACCUMULATION IN DEAD RECKONING

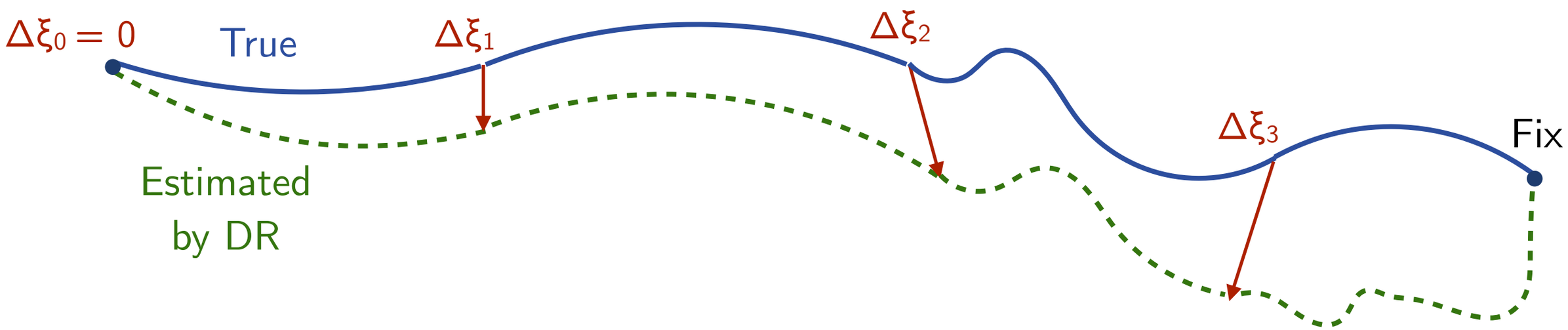
- The uncertainty of dead reckoning / odometry increases over time and maybe over distance ➔ A new fix is intermittently needed to determine a more reliable position from which a new dead reckoning process (i.e., integration) can be restarted

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



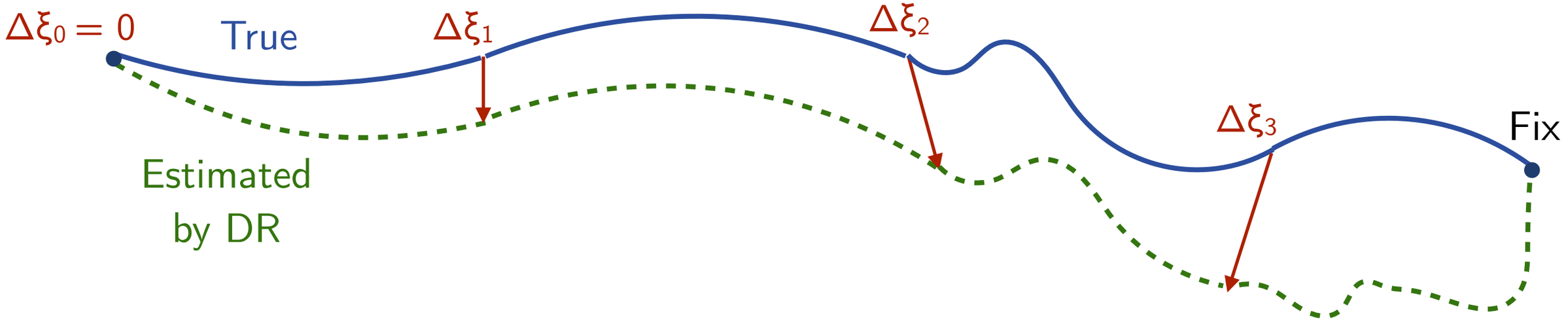






INTRINSIC ERRORS IN LOCALIZATION

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!

ERROR PROPAGATION EFFECTS

- At each time step i the controller tells the robot to move by some amount $(\Delta x_i, \Delta y_i)$, but the robot actually moves by $(\Delta x_i + \epsilon_i^x, \Delta y_i + \epsilon_i^y)$, where ϵ_i^x and ϵ_i^y are small random errors.

- Let's assume (as it is commonly done) that the errors are *independent, with zero mean, and stationary variance*:

$$\forall_i \mathsf{E}[\epsilon_i^x] = \mathsf{E}[\epsilon_i^y] = 0$$
$$\forall_i \mathsf{E}[(\epsilon_i^x - \mathsf{E}[\epsilon_i^x])(\epsilon_i^x - \mathsf{E}[\epsilon_i^x])] = \sigma^2$$
$$\forall_i \mathsf{E}[(\epsilon_i^y - \mathsf{E}[\epsilon_i^y])(\epsilon_i^y - \mathsf{E}[\epsilon_i^y])] = \sigma^2$$
$$\forall_{i \neq j} \mathsf{E}[(\epsilon_i^x - \mathsf{E}[\epsilon_i^x])(\epsilon_i^y - \mathsf{E}[\epsilon_i^y])] = 0$$

- After N steps, starting from (x_0, y_0), the robot is expected to be in

$$(x_N, y_N) = (x_0, y_0) + \sum (\Delta x_i, \Delta y_i)$$

ERROR PROPAGATION EFFECTS

$$
\begin{aligned}
\mathrm{E}[(x_N, y_N)] &= \mathrm{E}[(x_0, y_0) + \sum(\Delta x_i + \epsilon_i^x, \Delta y_i + \epsilon_i^y)] \\
&= (x_0, y_0) + (\mathrm{E}[\sum(\Delta x_i + \epsilon_i^x)], \mathrm{E}[\sum(\Delta y_i + \epsilon_i^y)]) \\
&= (x_0, y_0) + (\sum \Delta x_i + \sum \mathrm{E}[\epsilon_i^x], \sum \Delta y_i + \sum \mathrm{E}[\epsilon_i^y]) \\
&= (x_0, y_0) + \sum(\Delta x_i, \Delta y_i).
\end{aligned}
$$

The expected position is at the commanded location
(no error bias)

ERROR PROPAGATION EFFECTS

What is the uncertainty on the final position?

$$\Sigma = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{yx} & \sigma_{yy} \end{bmatrix}$$

$$\sigma_{xx} = \mathrm{E}[(x - \mathrm{E}[x])^2] \ldots = N\sigma^2$$
$$\sigma_{yy} = \mathrm{E}[(y - \mathrm{E}[y])^2] \ldots = N\sigma^2$$
$$\sigma_{xy} = \sigma_{yx} = 0$$

$$\Sigma = N \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix}$$

Errors in x and y grow unbounded for $N \to \infty$

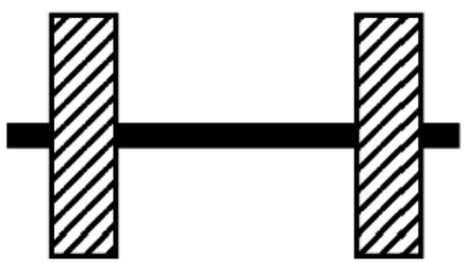
ERROR SOURCES IN ODOMETRY / DEAD RECKONING

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

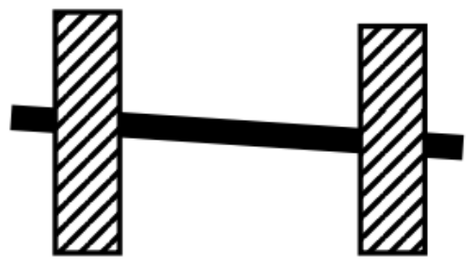
- *Numerical integration* errors (approximations, floating point rounding,....)
- Error readings from the *wheel encoders*
- Wheel *slippage* (friction issues, uneven ground, ...)
- Inaccurate measurement or calibration of *wheel and chassis parameters*, that are reflected in inaccuracies in the results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

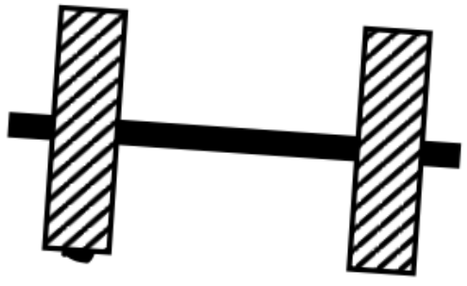
TYPICAL ISSUES RELATED TO WHEELS



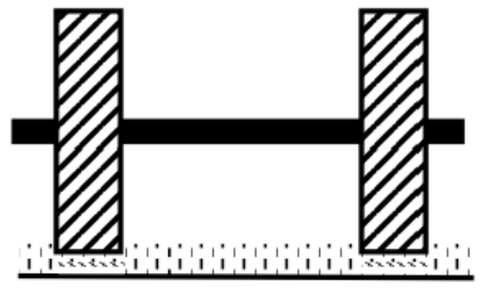
ideal case



different wheel diameters



bump

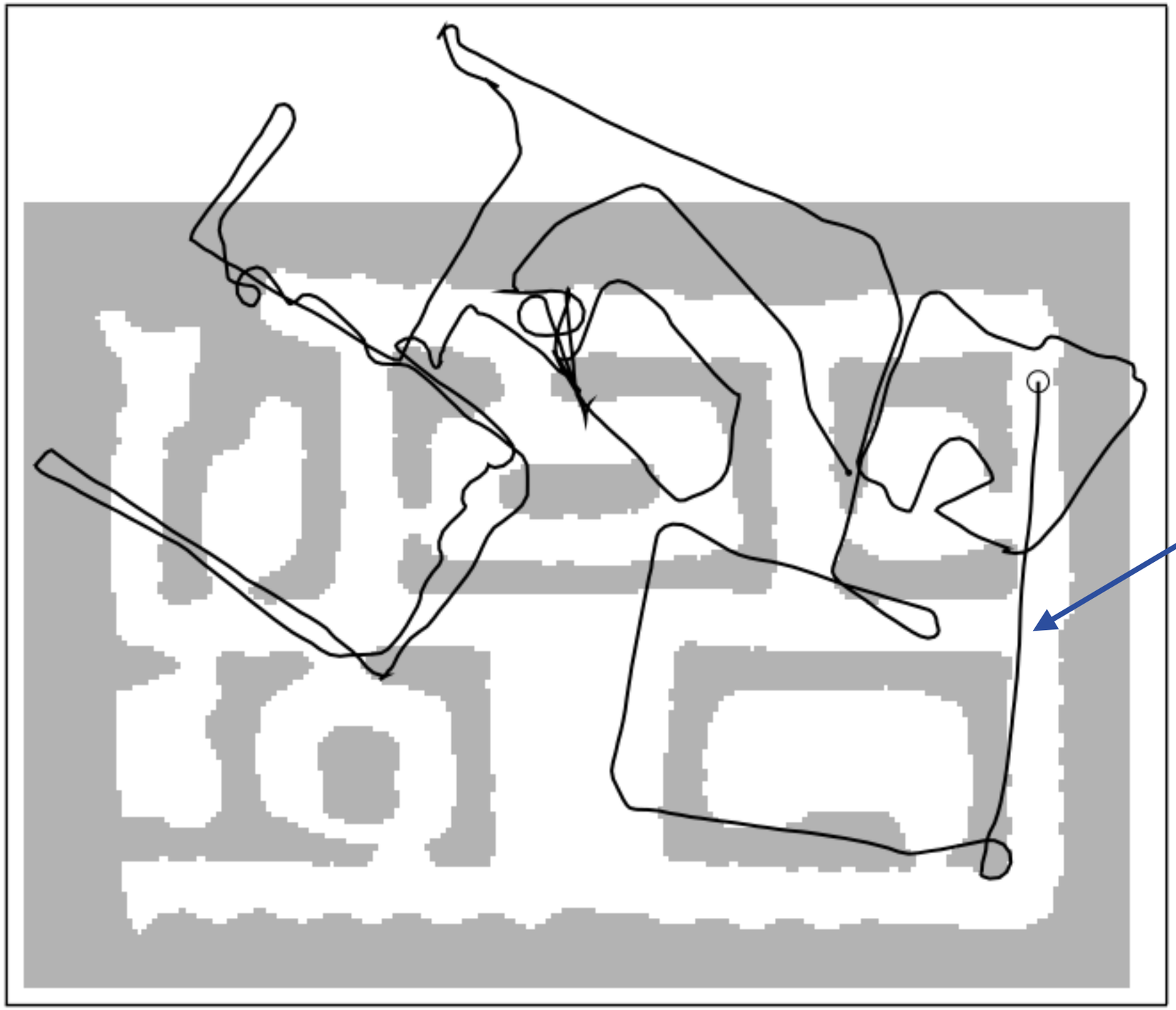


carpet

and many more ...

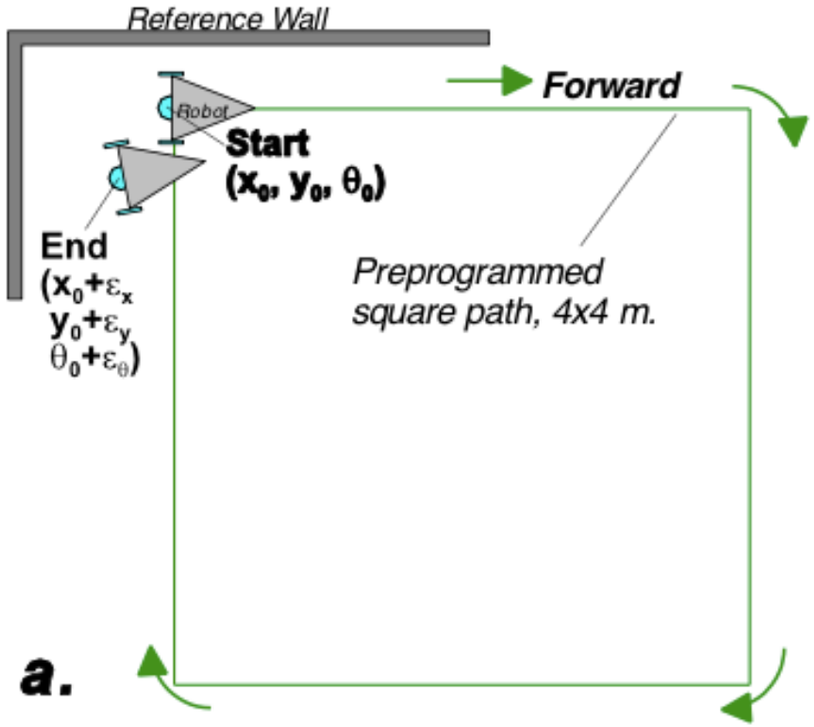
Using wheels' measures to compute odometry is not perfect at all, it comes with a number of *uncertainties*!

A TYPICAL RESULT FROM ODOMETRY

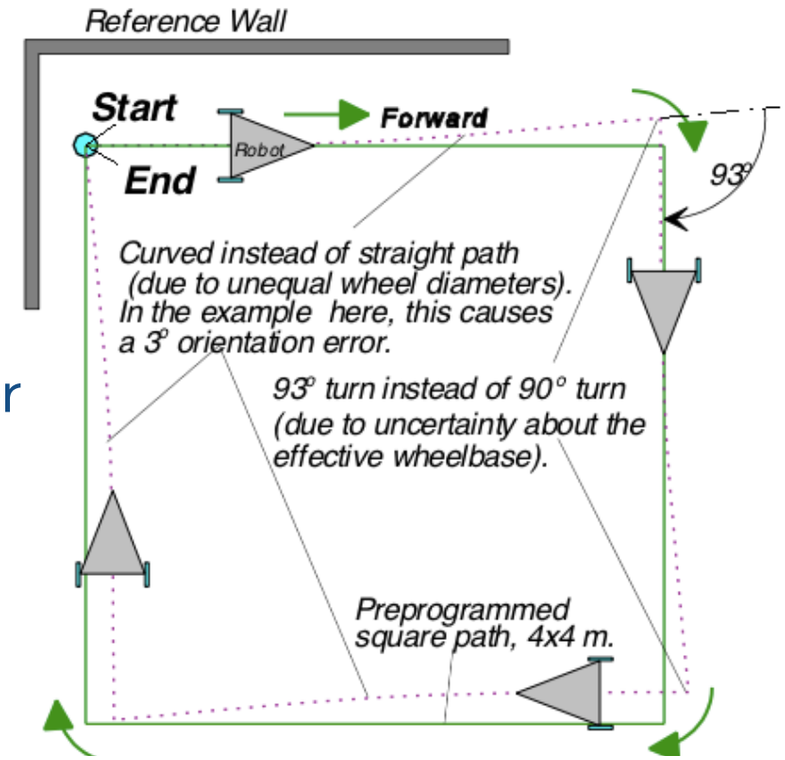


This the path the robot has estimated using odometry measures

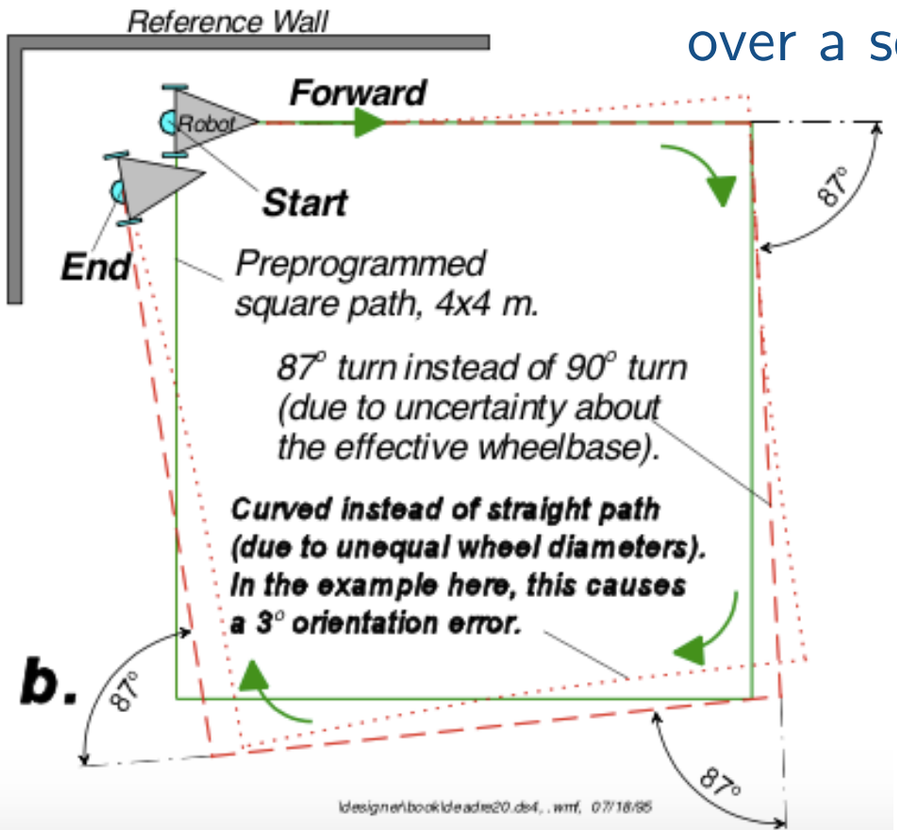
INTERESTING CASES FOR ODOMETRY ERRORS



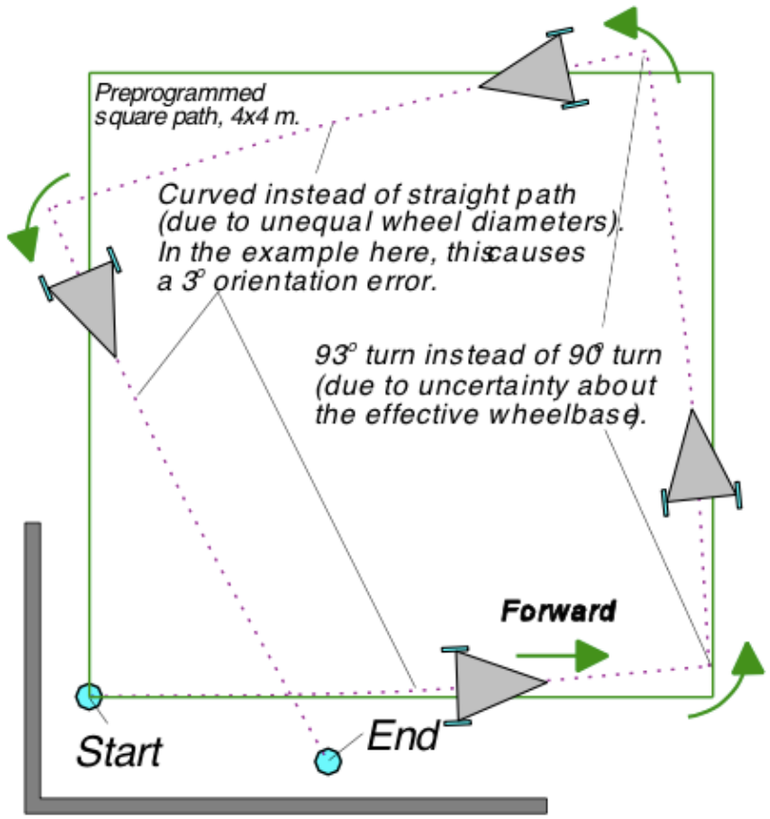
Different errors
canceling with each other



Cumulative error
over a squared path



Different errors
summing up



FROM CONTINUOUS TIME TO DISCRETE TIME

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

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

- 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 THE 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: 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

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

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

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

Problem: During the interval, the orientation changes because of the issued velocity commands, but this is not taken into account in the calculation of sin and cos, since θ_k is kept as at the beginning of the interval

EULER APPROACH: INITIAL ORIENTATION

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

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

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

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

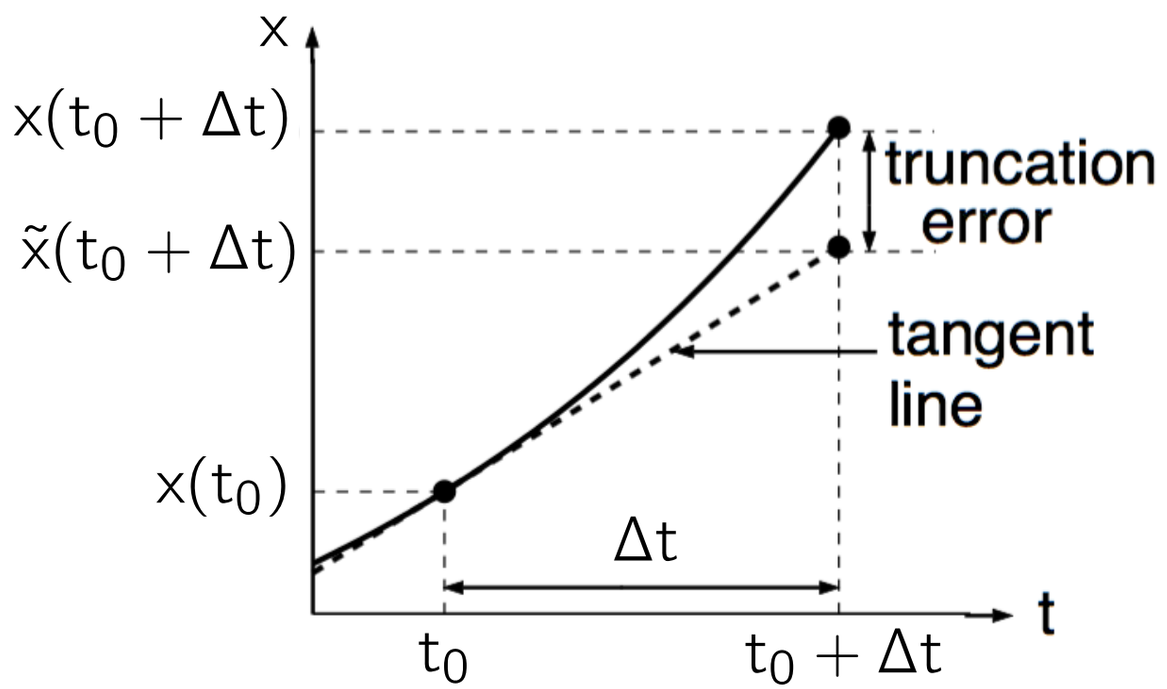
$$\dot{x} \approx \frac{x(t+\Delta t) - x(t)}{\Delta t}$$

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

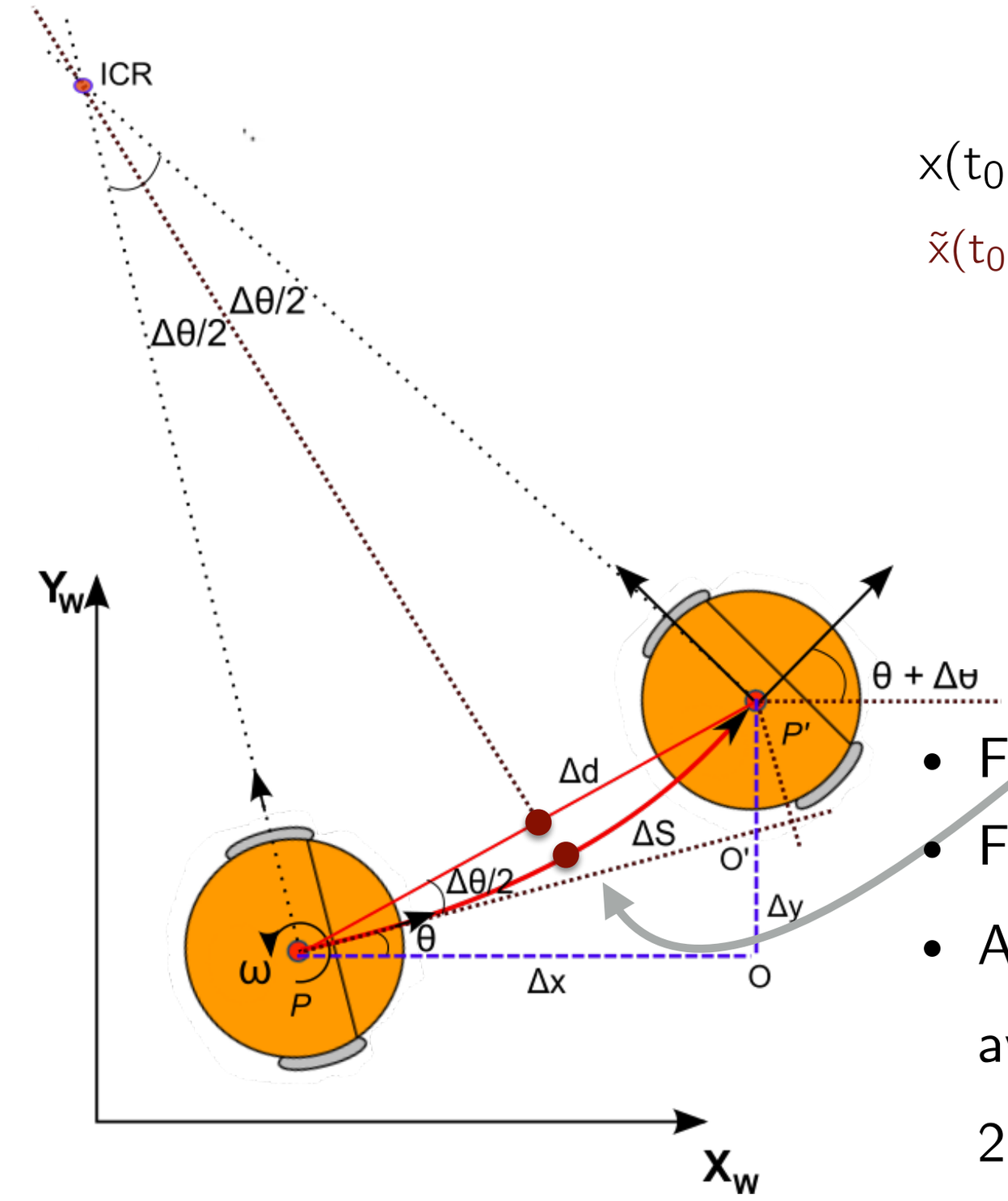
Euler method: first term of Taylor series, 1st order approximation

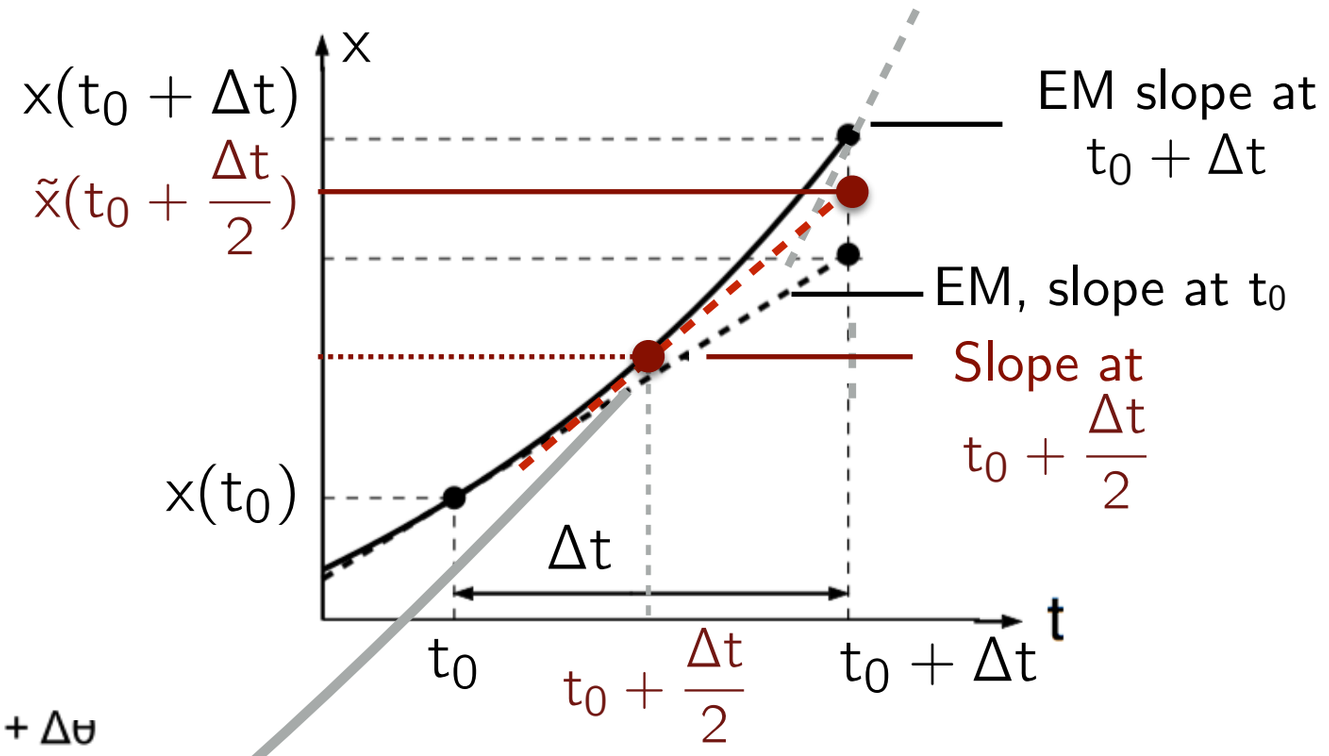
$$x(t_0+\Delta t) = x(t_0) + \Delta t\ \dot{x}(t_0) + \frac{(\Delta t)^2}{2!}\ \ddot{x}(t_0) + \frac{(\Delta t)^3}{3!}\ \dddot{x}(t_0) + \ldots$$

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



RUNGE-KUTTA APPROACH: 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 $t_0 + \Delta t \rightarrow$ 2nd order approx, Taylor in $t_0 + \Delta t/2$

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

RUNGE-KUTTA APPROACH: AVERAGE ORIENTATION

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

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

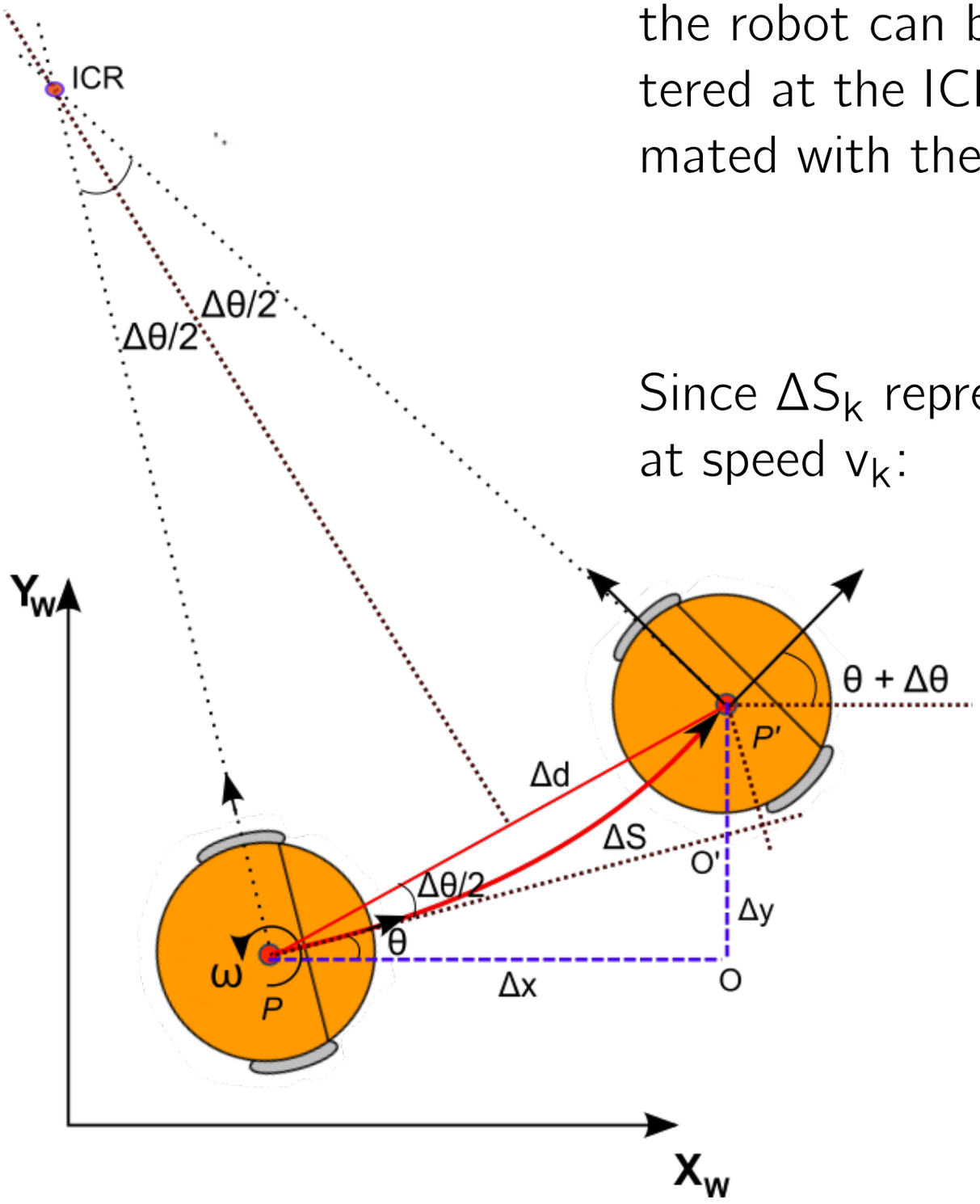
- The <u>average orientation</u> over the integration interval is considered, therefore the x_{k+1} and y_{k+1} values are better than in the Euler case, when only the initial orientation is taken, but still they are **approximations**, while θ_{k+1} is exact

- 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)

- In the right set of equations, ΔS_k is used instead of $v_k \Delta t_k$, which are equivalent, but ΔS_k can be more direct to compute with wheels

Geometric derivation of the Runge-Kutta equations → ...

RUNGE-KUTTA: GEOMETRIC DERIVATION



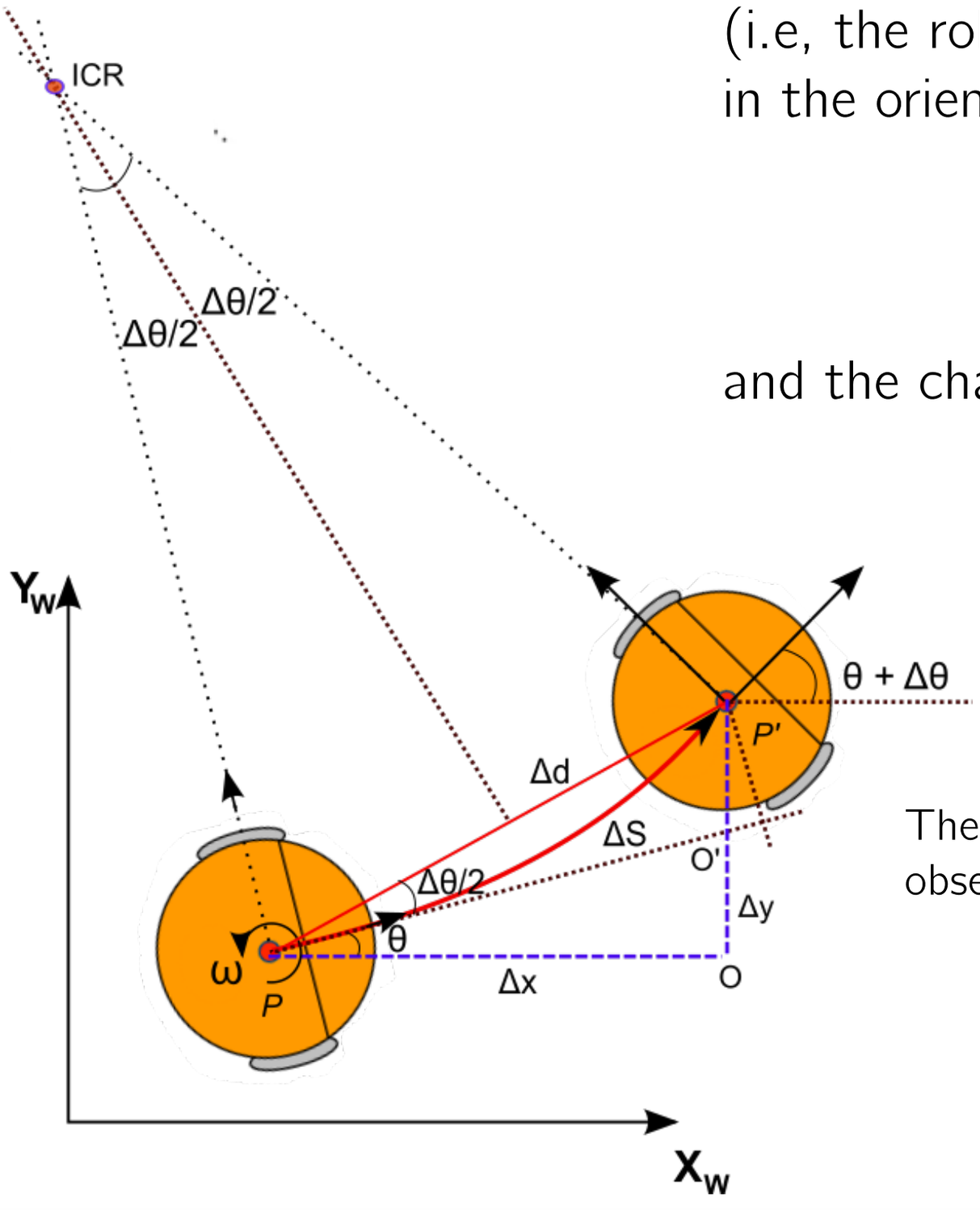
- **Travel length:** At time-step k, if Δt_k sufficiently small, the robot can be seen as traveling on a *circular arc*, centered at the ICR, for a distance ΔS_k that can be approximated with the length of the corresponding cord:

$$\Delta S_k \approx \Delta d_k.$$

Since ΔS_k represents the distance traveled in Δt_k moving at speed v_k:

$$v_k \Delta t_k = \Delta S_k \approx \Delta d_k$$

RUNGE-KUTTA: GEOMETRIC DERIVATION



- **Orientation change:** With the cord approximation (i.e, the robot is moving along the cord), the change in the orientation angle becomes equal to

$$\frac{\Delta\theta_k}{2},$$

and the change in x and y:

$$\Delta x_k = \Delta d_k \cos\left(\theta_k + \frac{\Delta\theta_k}{2}\right)$$

$$\Delta y_k = \Delta d_k \sin\left(\theta_k + \frac{\Delta\theta_k}{2}\right)$$

The relations for Δx_k and Δy_k result from trigonometry, observing that:

$$\mathrm{P} - \widehat{\mathrm{ICR}} - \mathrm{P}' = \Delta\theta$$

$$\Rightarrow \mathrm{ICR}\widehat{-\mathrm{P}-\mathrm{P}'} = 90 - \Delta\theta/2$$

$$\Rightarrow \mathrm{P}'\widehat{-\mathrm{P}-\mathrm{O}'} = 90 - (90 - \Delta\theta/2) = \Delta\theta/2$$

EXACT APPROACH: VARIATION OF THE ORIENTATION

Discretization of the kinematic equations previously found using the ICR:

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

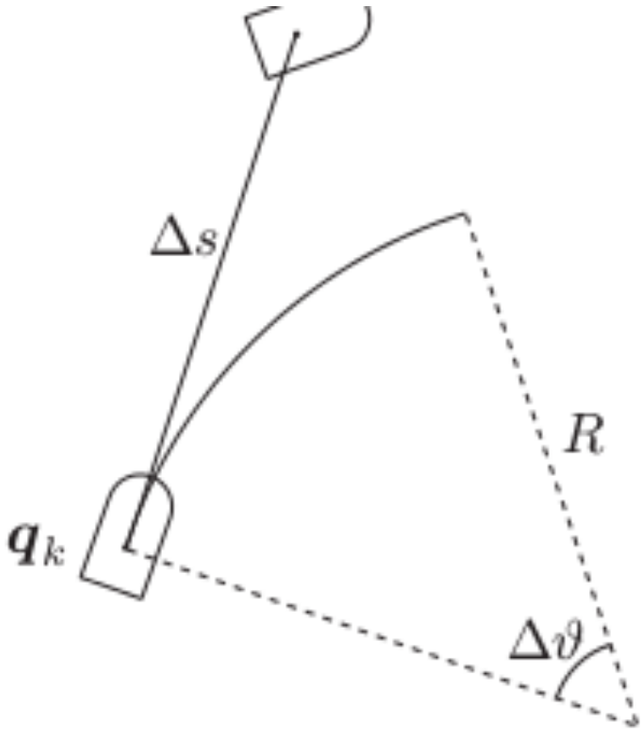
$$x_{k+1} = x_k + \frac{v_k}{\omega_k}(\sin\theta_{k+1} - \sin\theta_k)$$

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

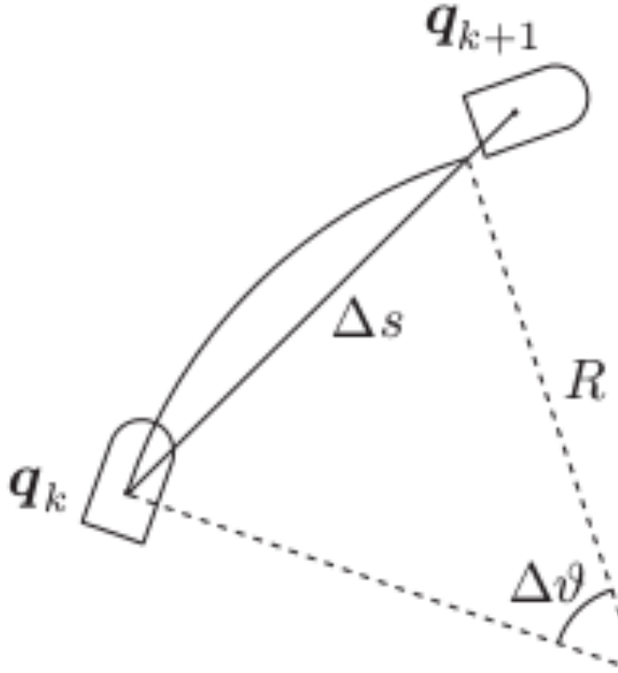
$$\theta_{k+1} = \theta_k + \omega_k \Delta t_k$$

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

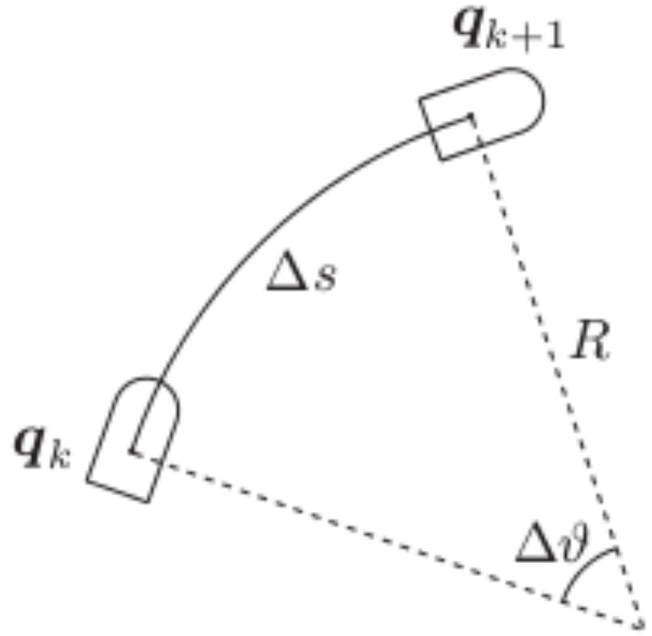
COMPARISON AMONG THE THREE METHODS



Euler



Runge-Kutta



exact