اصول علم ربات – اسلاید دهم

Fundamentals of Robotics – Slide 10

Kinematics 4

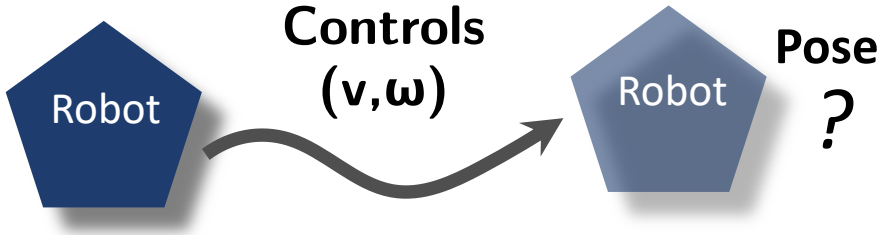
دکتر مهدی جوانمردی

زمستان ۱۴۰۰ – بهار ۱۴۰۱

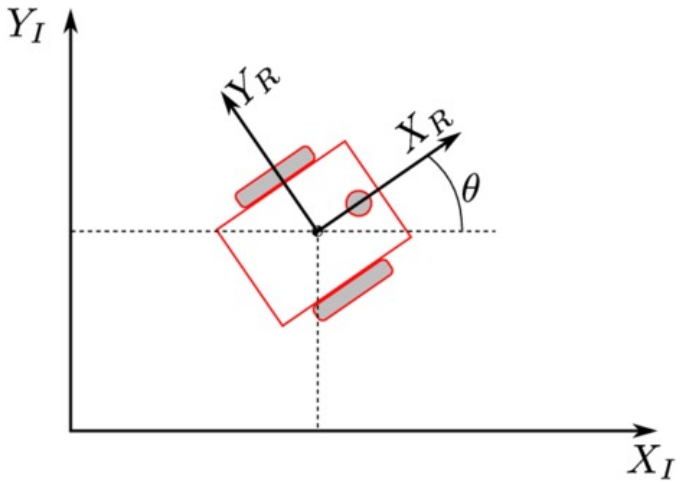
[slides adapted from Gianni Di Caro, @CMU with permission]

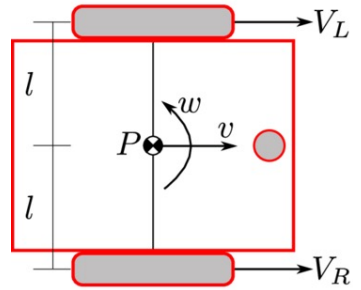
Pose prediction by integration of differential kinematic equations

Posture prediction: *Forward Kinematics*



For a two-wheeled differential robot:





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

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

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

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

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

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

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

Easy but useful cases: constant wheels' velocity

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

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

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

- 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

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

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

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

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

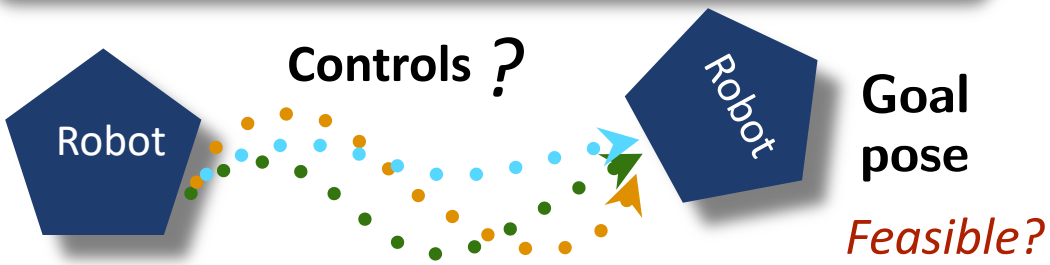
Robots moves long a circular trajectory of radius:

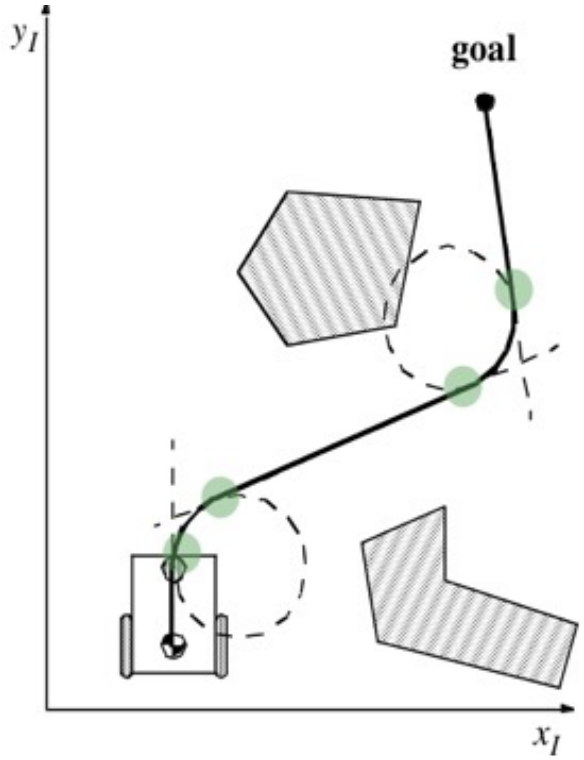
$$R = \ell\frac{v_R + v_L}{v_R - v_L}$$

What's this?

Inverse kinematics

Posture regulation: *Inverse Kinematics*





Path planning in continuous configuration space by defining velocity profiles over time

Simplified approach: decompose the trajectory in a discrete sequence of waypoints that can be joined by *primitive* motion controls, i.e., by:

- Moving in straight line
- Rotation in place or over segments of a circle

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

$v_L = v_R = v$

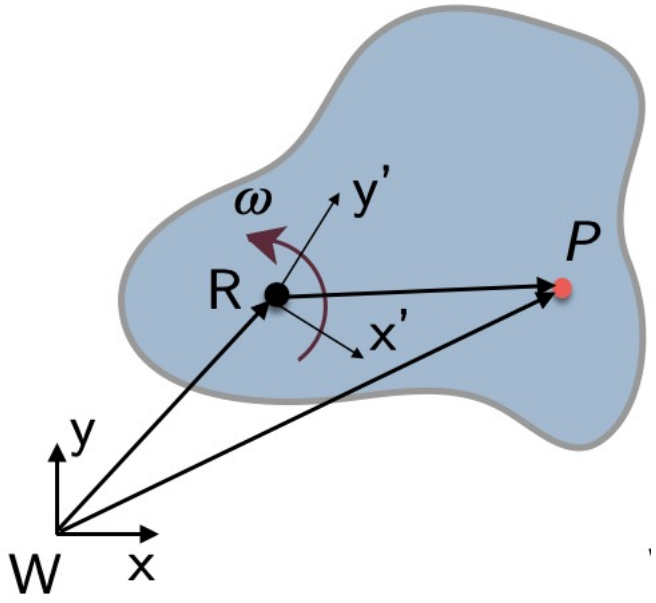
$x(t) = x_0 + vt\cos(\theta)$

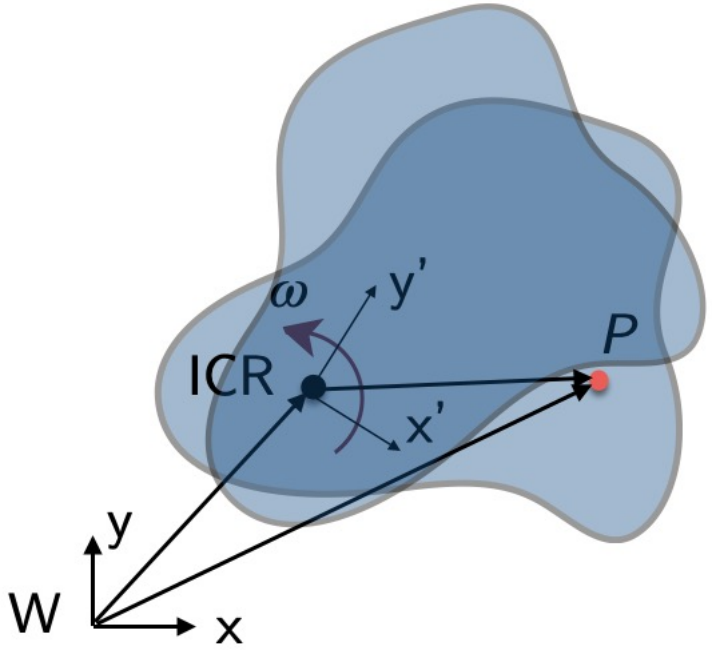
$y(t) = y_0 + vt\sin(\theta)$

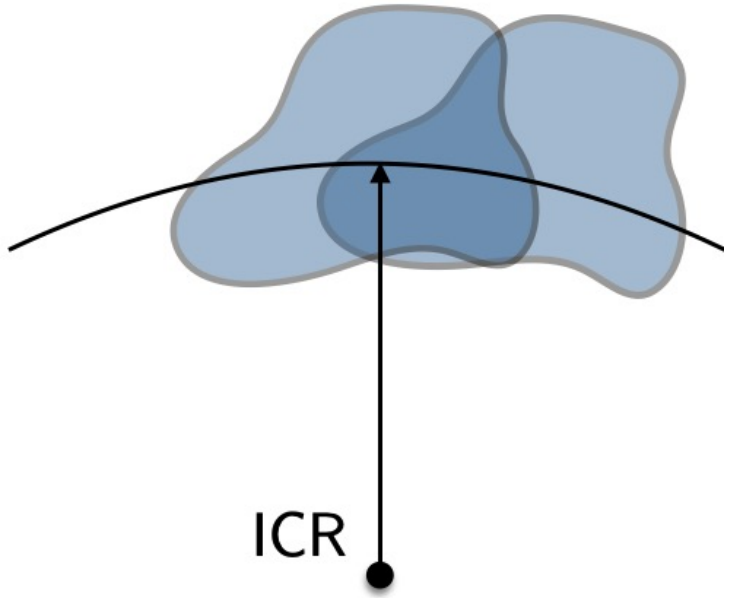
$\theta(t) = \theta_0$

How to move over segments of a circle?

Instantaneous center of rotation / curvature (ICR / ICC)







Pure rotation about the rotation center R at angular speed ω

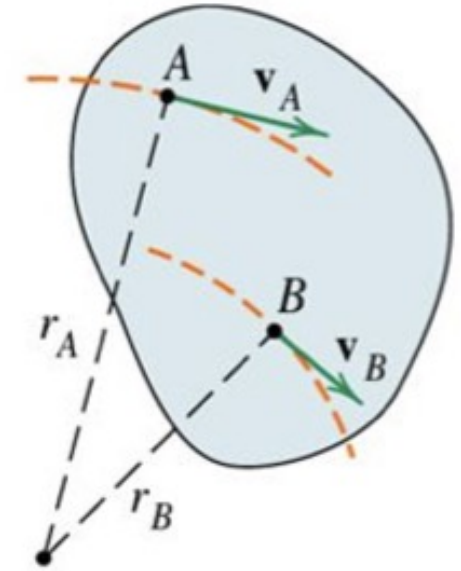
In W, R is the only point that doesn't move

(Instantaneous) center of rotation → ICR

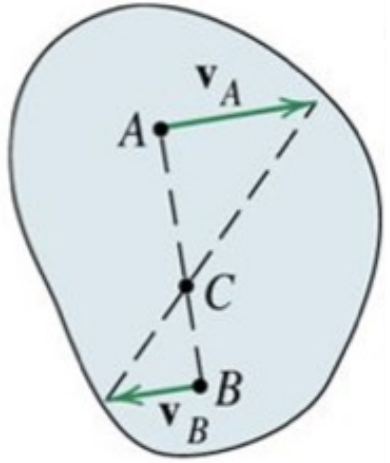
Rotation + Translation: it can be described as a **rotation only** about the ICR, which doesn't move

Instantaneous center of rotation / curvature (ICR / ICC)

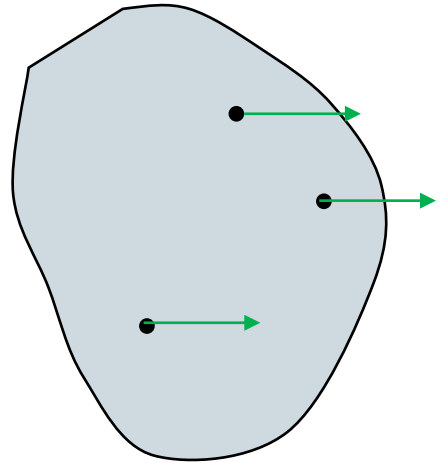
$$^{R}\boldsymbol{r}_{ICR} = \frac{1}{\omega^2}(\boldsymbol{\omega} \times {}^{W}\boldsymbol{v}_{R})$$



ICR



$ICR \equiv C$



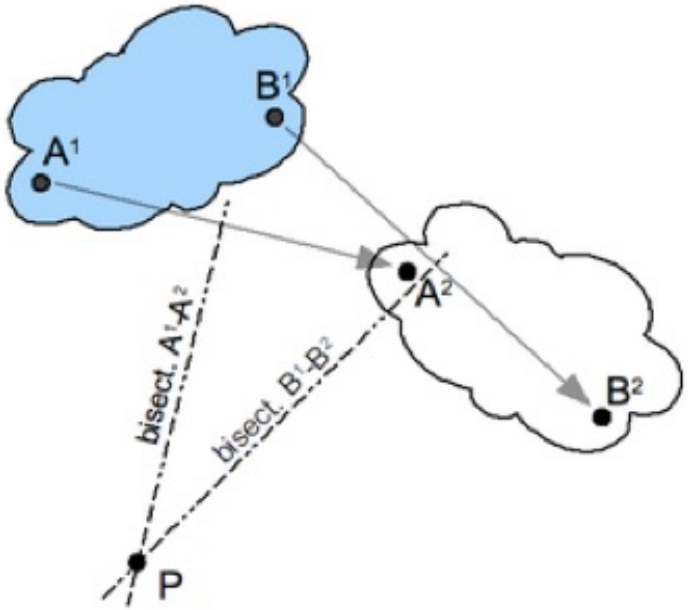
$ICR \to \infty$

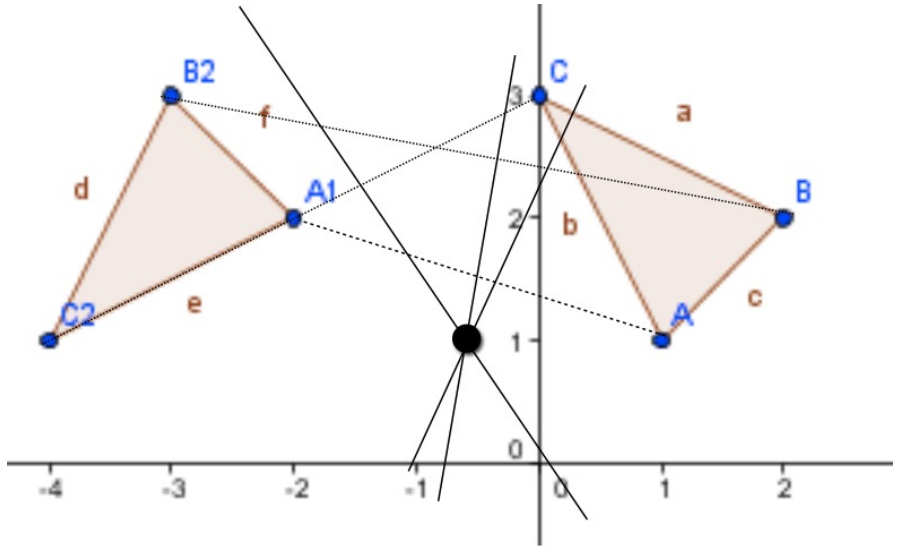
- Selected a point A in the body, the position vector $\overrightarrow{ICR} - \vec{A}$ is **perpendicular** to the velocity vector in A

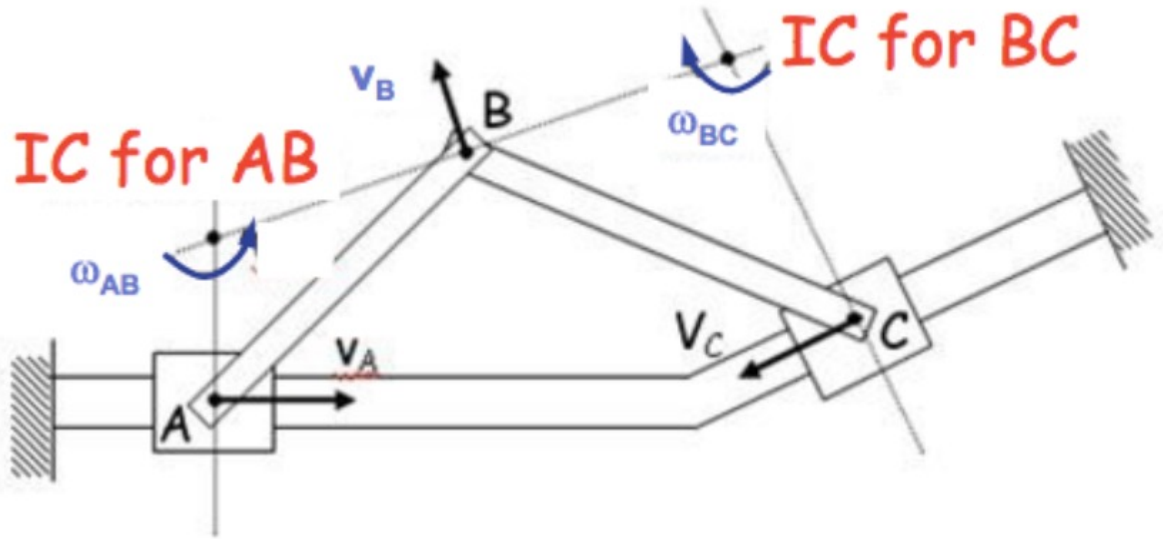
✓ If we know the velocity at two points of the body, A and B, then the location of ICR can be determined geometrically as the **intersection** of the lines which go through points A and B and are perpendicular to $\boldsymbol{v}_A$ and $\boldsymbol{v}_B$

○ When the angular velocity, $\boldsymbol{\omega}$, is very small, the center of rotation is very far away from the body. When it is zero (i.e., a pure translation), the center of rotation is at infinity.

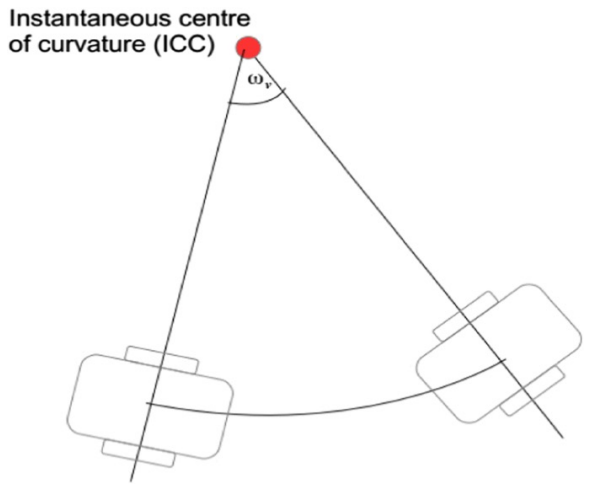
Geometric construction, multi-part bodies

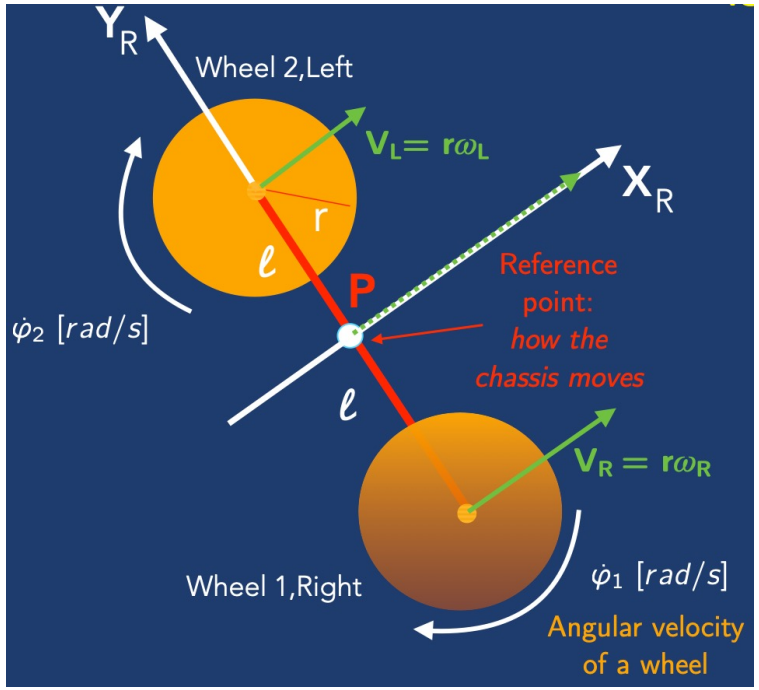


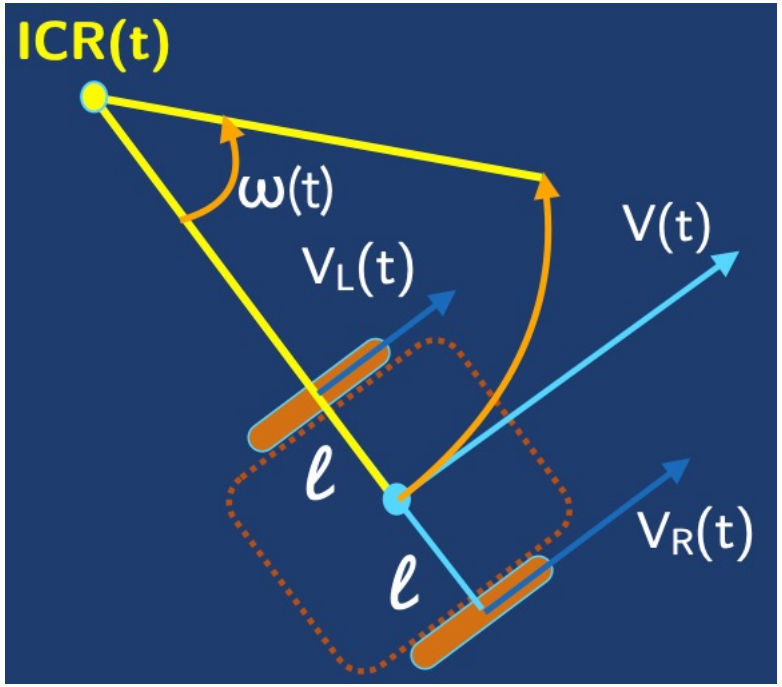




ICC for Mobile robots

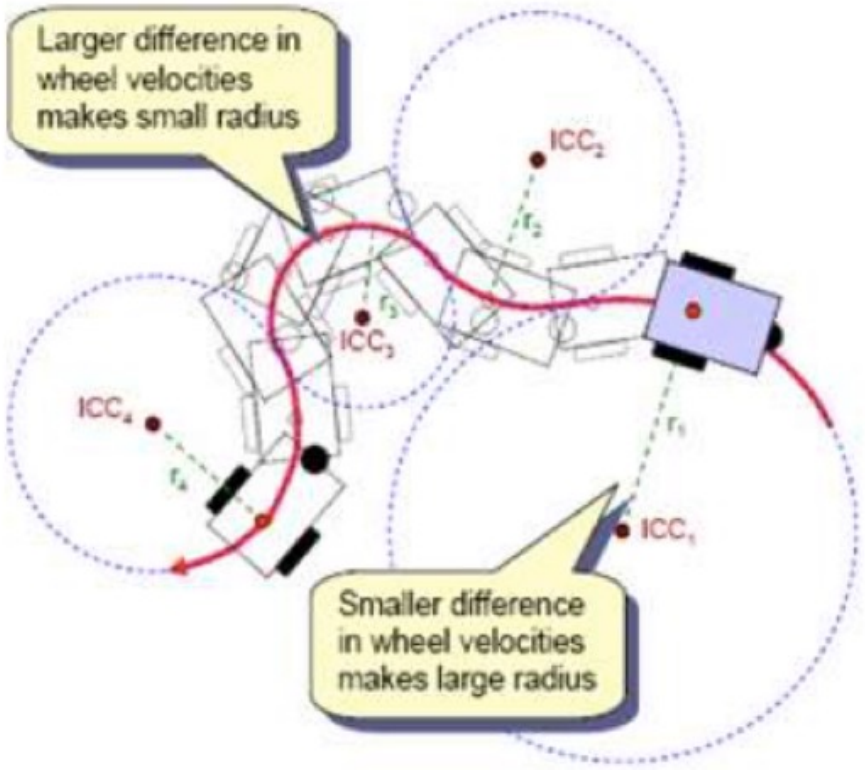






- No side-motion constraints: no motion along the line ⊥ to the plane of each wheel
- For each wheel, (**v** — **ICR**) vector overlaps with the no side-motion line
- **At any time t, ICR is the intersection of all zero motion lines from wheels**

ICC for Mobile robots

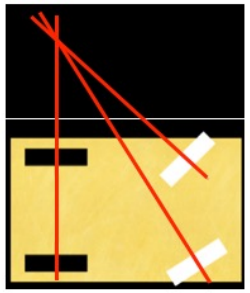


- At any time t, the robot reference point (between the wheels in the figure) moves along a circumference of radius R with center on the zero-motion line, the **center of the circle is the ICC**

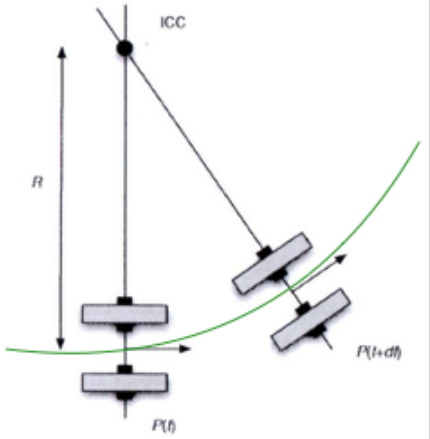
❖ The ICC changes over time as a function of the individual wheel velocities, and, in particular , of their relative difference

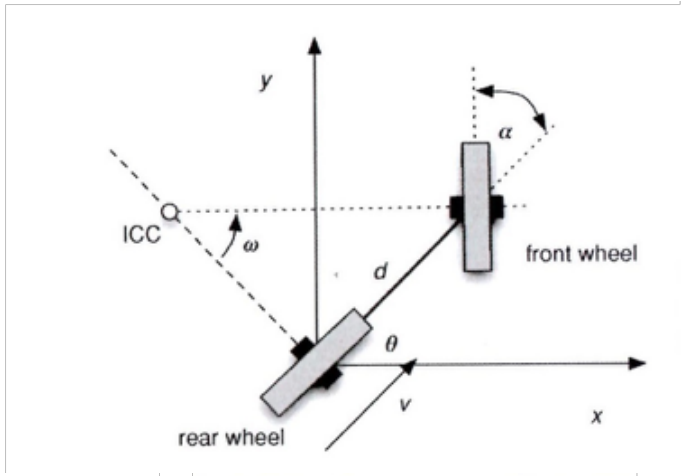
ICC for Mobile robots

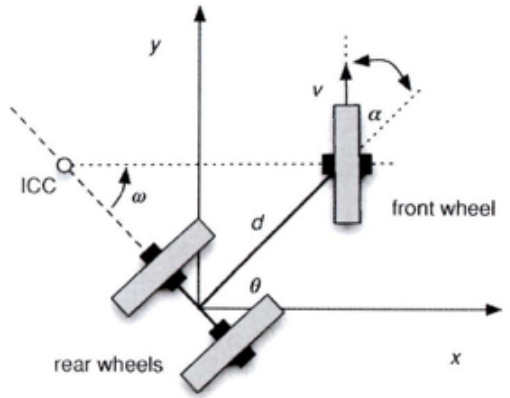
- **The ICR is the point around which each wheel makes a circular course, with a different radius, depending on wheel's position on the chassis**



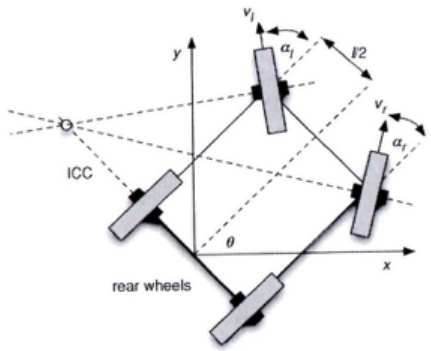
- ICR defines a zero motion line drawn through the horizontal axis perpendicular to the plane of each wheel on the chassis



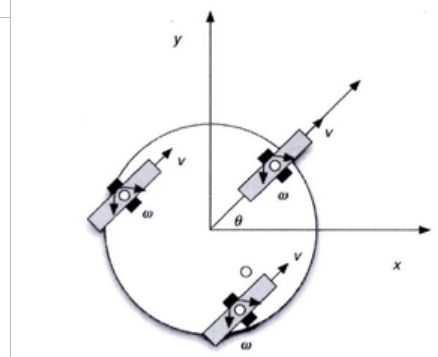




(a) Tricycle schematic



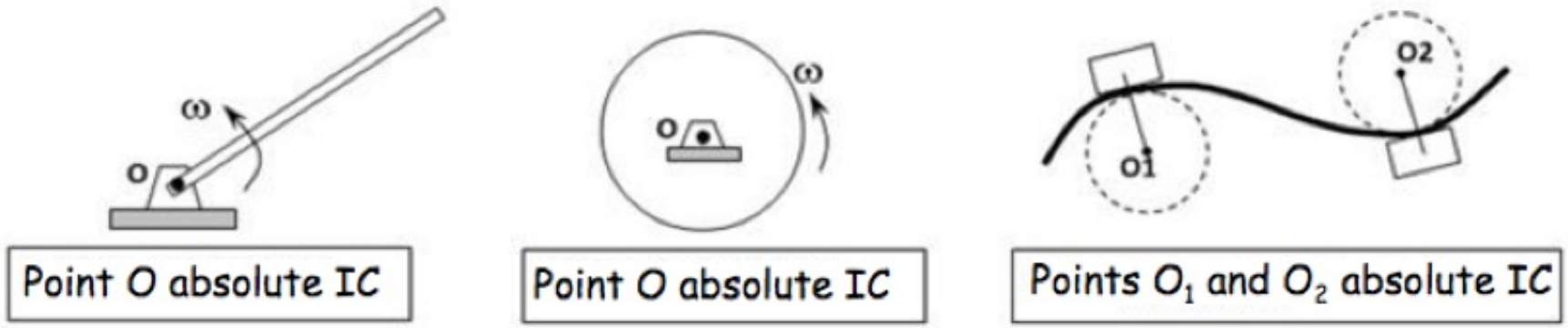
(a) Ackerman steering



(a) Synchronous drive

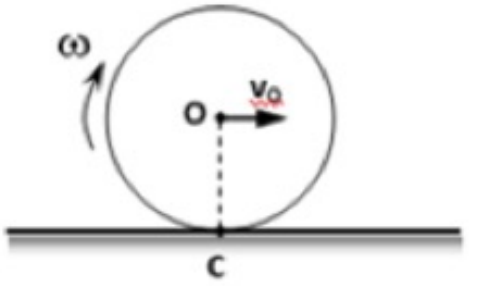
For a holonomic robot the ICC it's in the center of the robot

Other examples: Absolute ICC

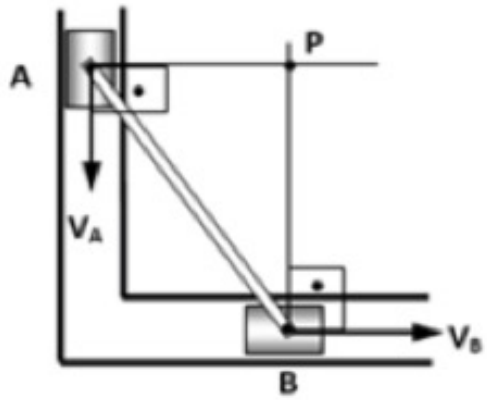


If the instantaneous center of curvature is fixed for a certain motion of the body, it can be referred to as absolute ICC

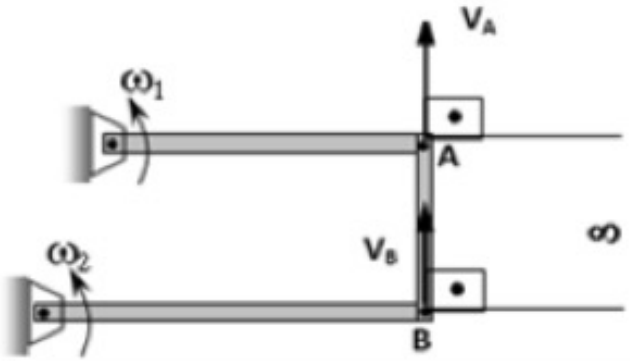
Other examples: Relative ICC



Point C relative IC



Point P relative IC

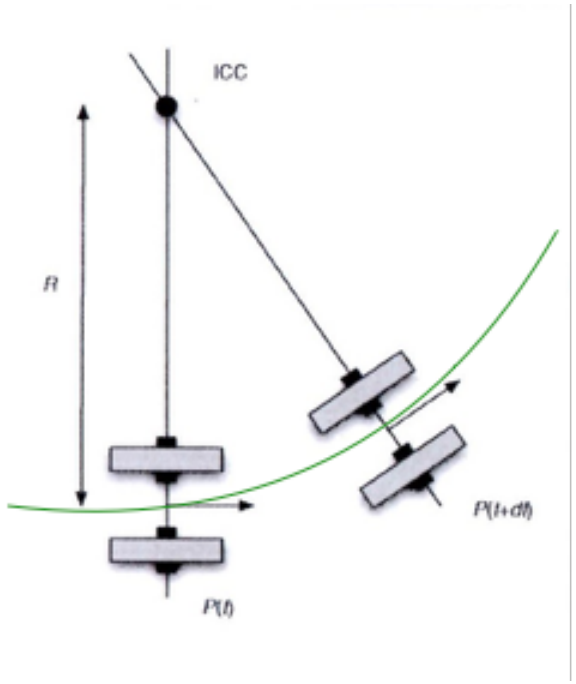


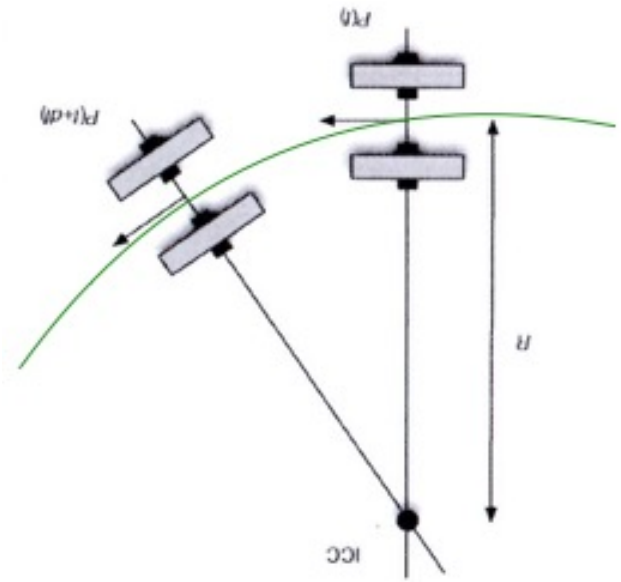
Relative IC in infinity

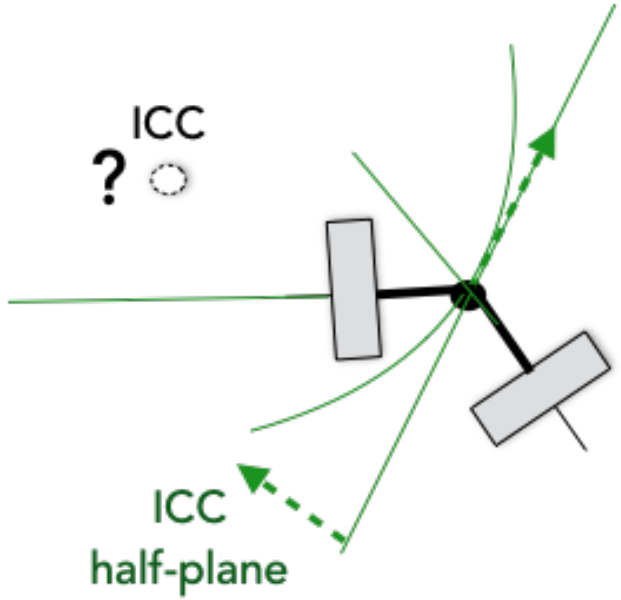
For the position shown, rod AB translates, $\omega_{AB}=0$

If the instantaneous center of curvature changes position for a certain motion of the body, it can be referred to as relative ICC

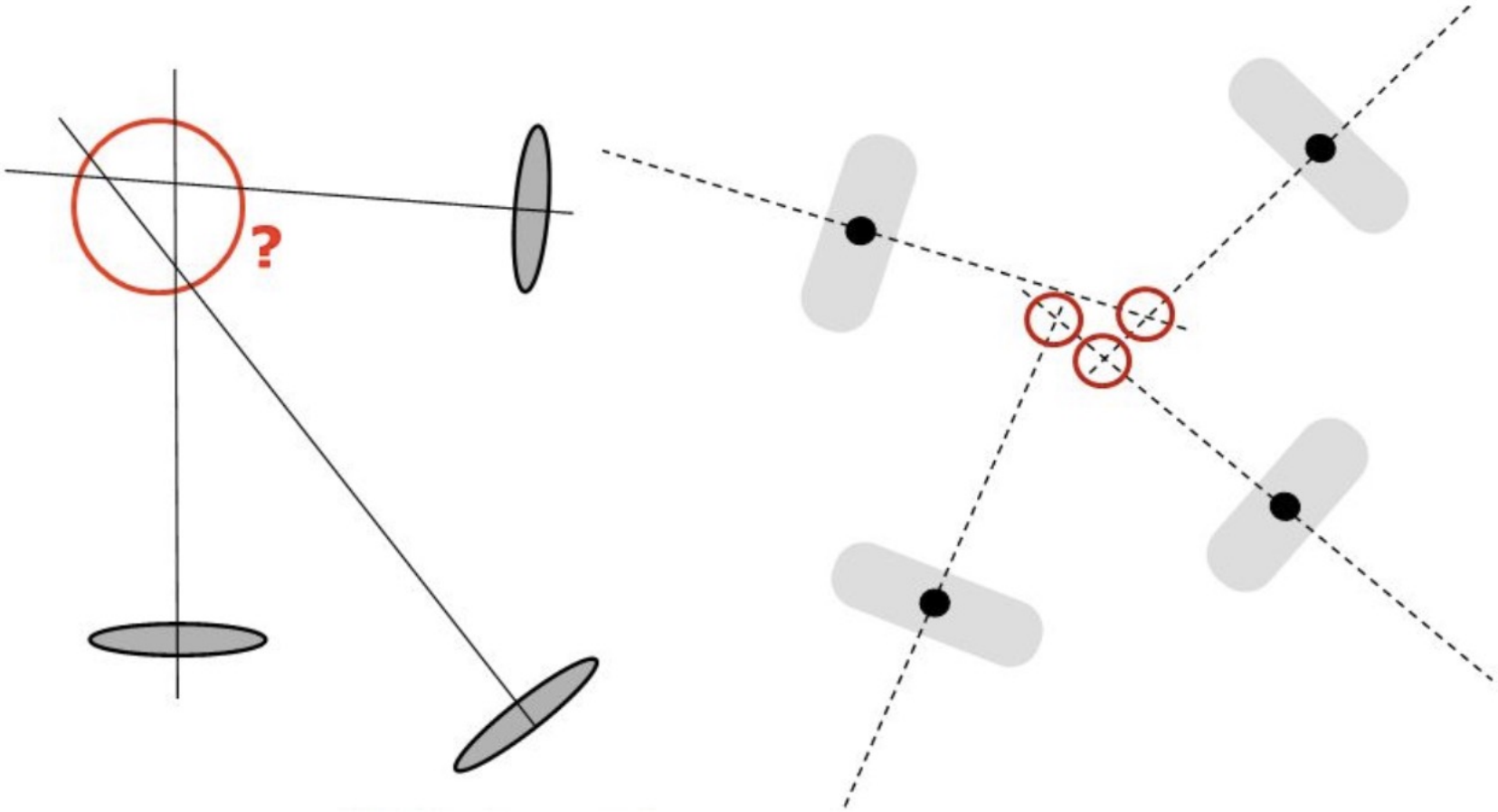
Other examples: Position of the ICC



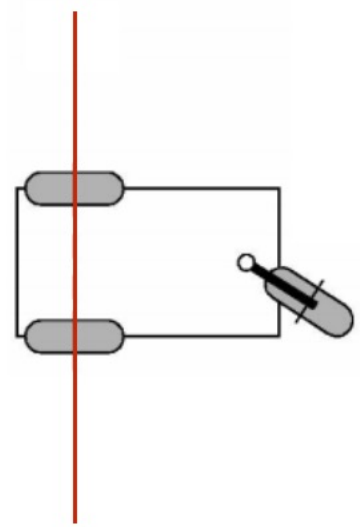




No ICC no motion without slippage

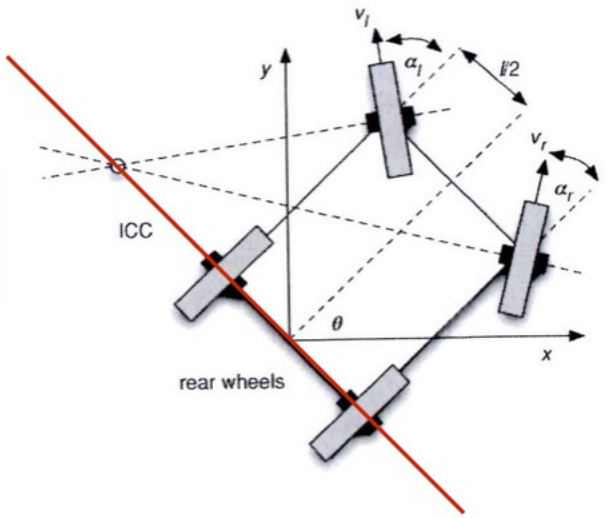


ICR and maneuverability

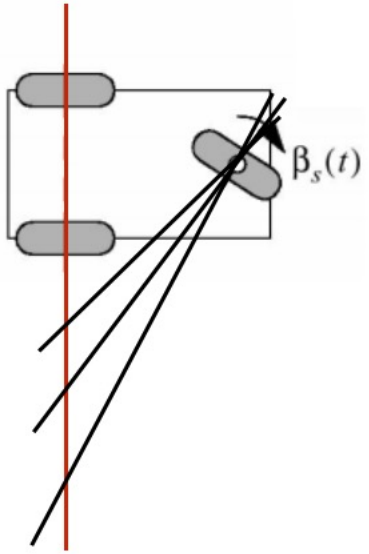


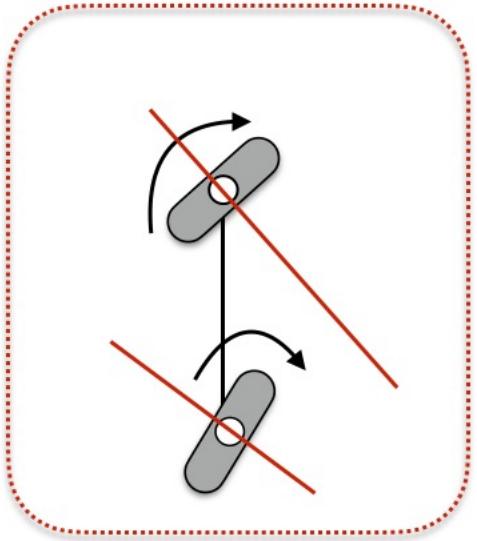
$\delta_M = (2+0) = 2$

$\delta_M = (1+1) = 2$



$\delta_M = (1+1) = 2$



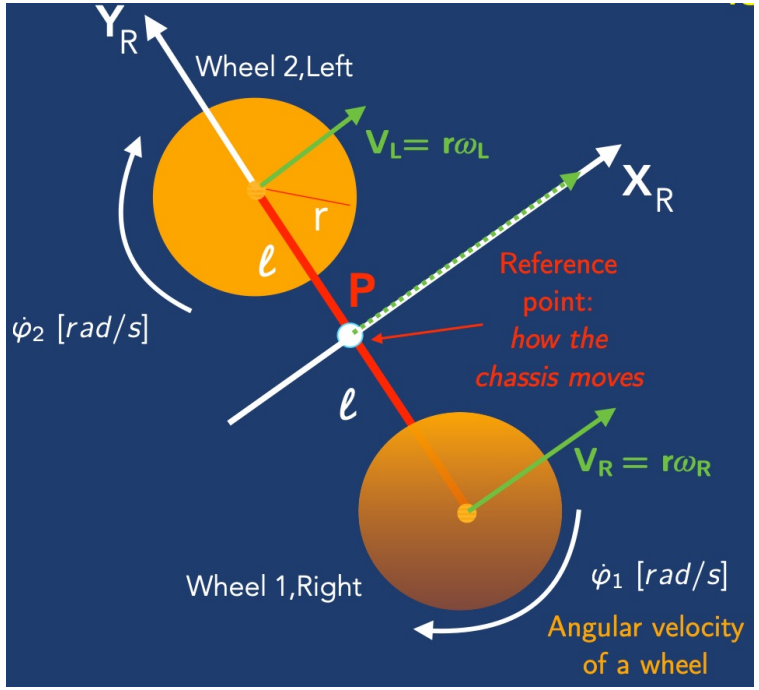


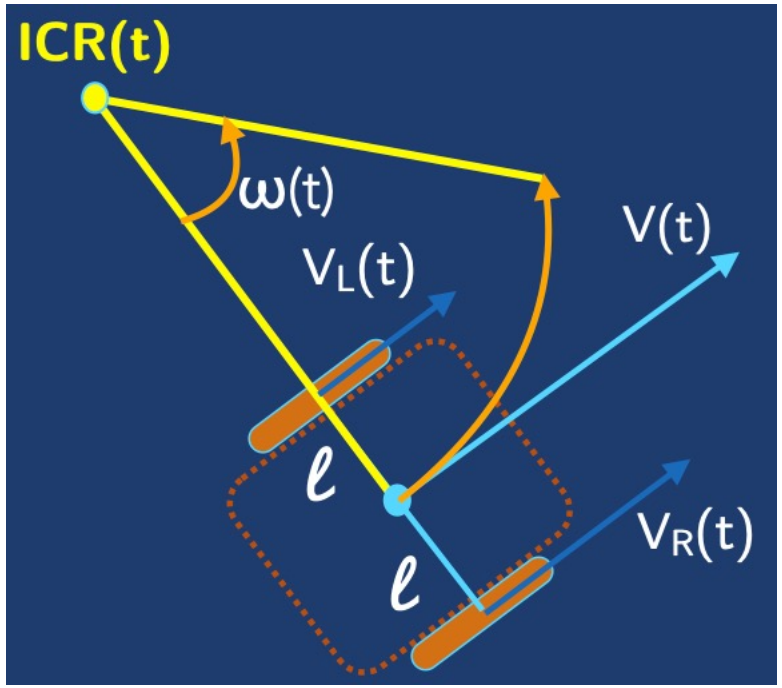
$\delta_M = (1+2) = 3$

- In the first three cases, the ICR cannot range anywhere on the plane, but it must lie on a predefined line with respect to the robot reference frame

- For any robot with $\delta_M = 2$, the ICR is always constrained on a line
- For any robot with $\delta_M = 3$, the ICR can be set to any point on the plane

ICR for Mobile robots





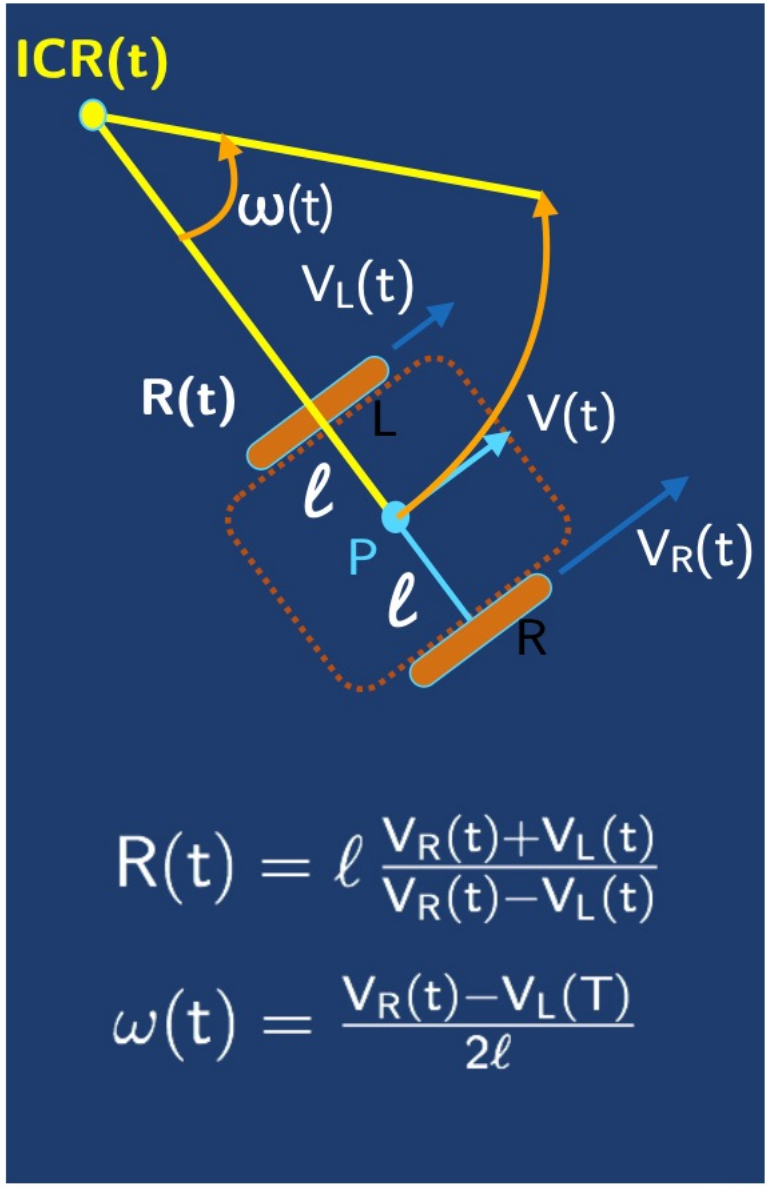
$$\omega(t)(R(t)+\ell) = V_R(t)$$
$$\omega(t)(R(t)-\ell) = V_L(t)$$

At any specific time instant *t:*

$$R(t) = \ell \frac{V_R(t)+V_L(t)}{V_R(t)-V_L(t)}$$

$$\omega(t) = \frac{V_R(t)-V_L(T)}{2\ell}$$

Special cases revisited



$$R(t) = \ell \frac{V_R(t)+V_L(t)}{V_R(t)-V_L(t)}$$

$$\omega(t) = \frac{V_R(t)-V_L(T)}{2\ell}$$

- $V_L = V_R$ ➔ $R = \infty$, and there is effectively no rotation, $\omega = 0$: Forward linear motion in a straight line
- $V_L = -V_R$ ➔ $R = 0$, meaning that it coincides with P, and $\omega = -V/\ell$: Rotation about the midpoint of the wheel axis (in place rotation)
- $V_L = 0$ ➔ $R = \ell$ (in the center of L), $\omega = V_R/2\ell$: Counterclockwise rotation about the left wheel
- $V_R = 0$ ➔ $R = -\ell$ (in the center of R), $\omega = -V_L/2\ell$: Clockwise rotation about the right wheel

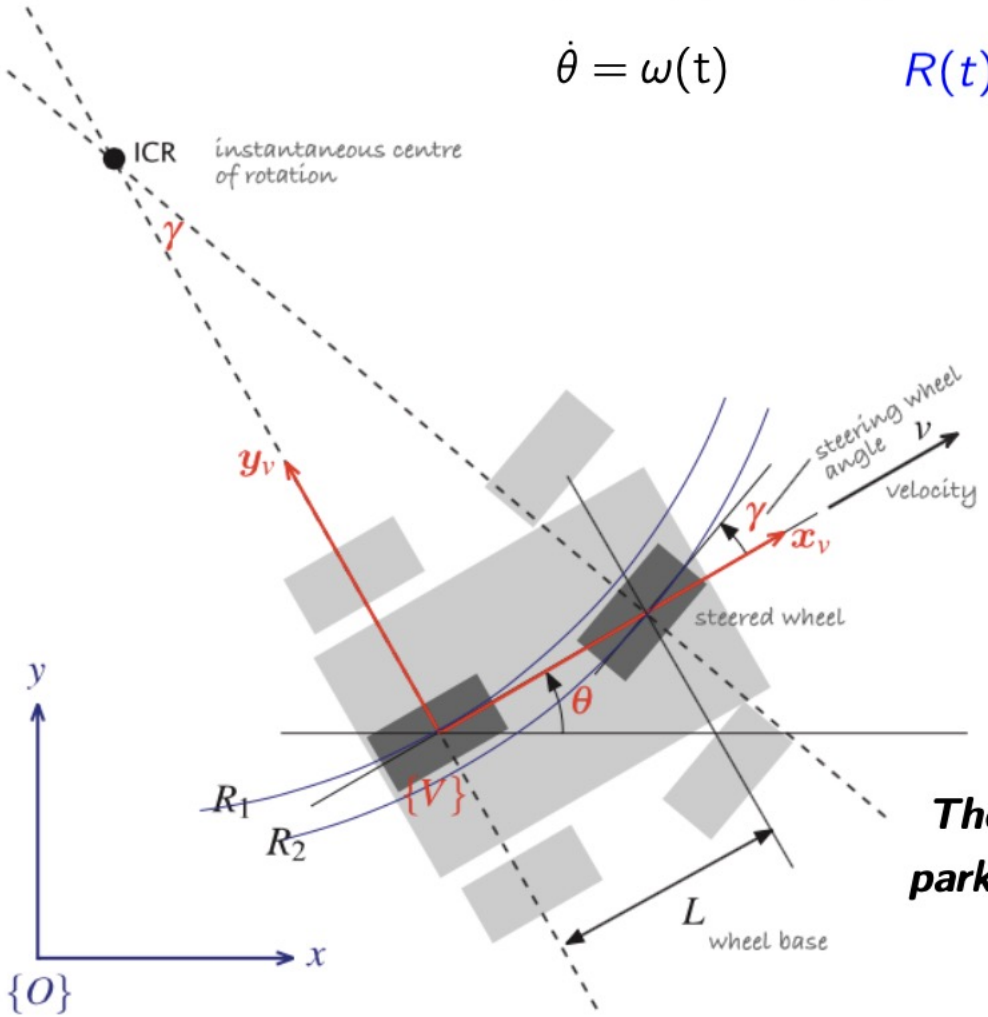
Kinematics equations for the bicycle model (of a car)

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

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

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

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



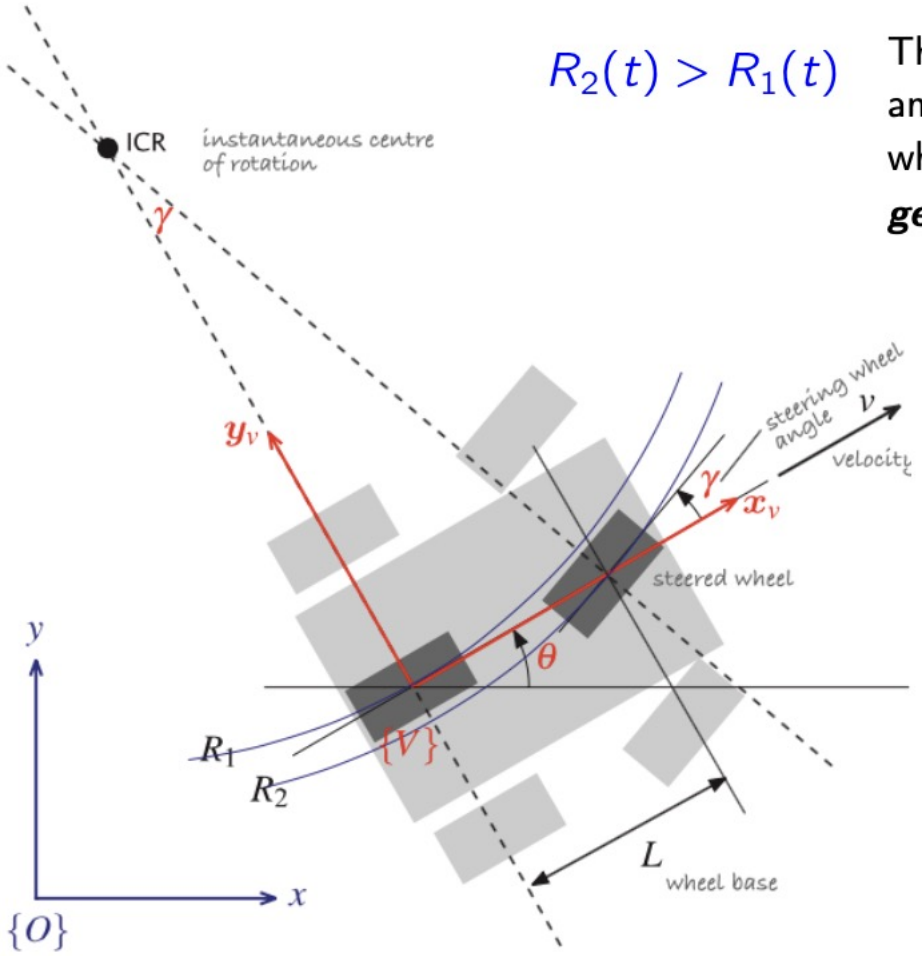
$$\dot{x} = v(t)\cos(\theta(t))$$

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

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

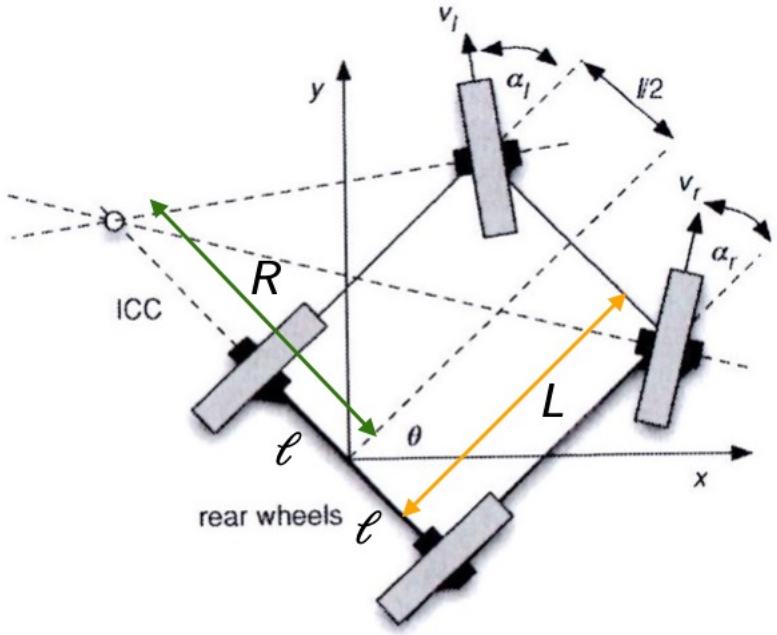
The max value of γ limits maneuverability: parking problem, complex inverse kinematics

Kinematics equations for the Ackermann steering model



$R_2(t) > R_1(t)$

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

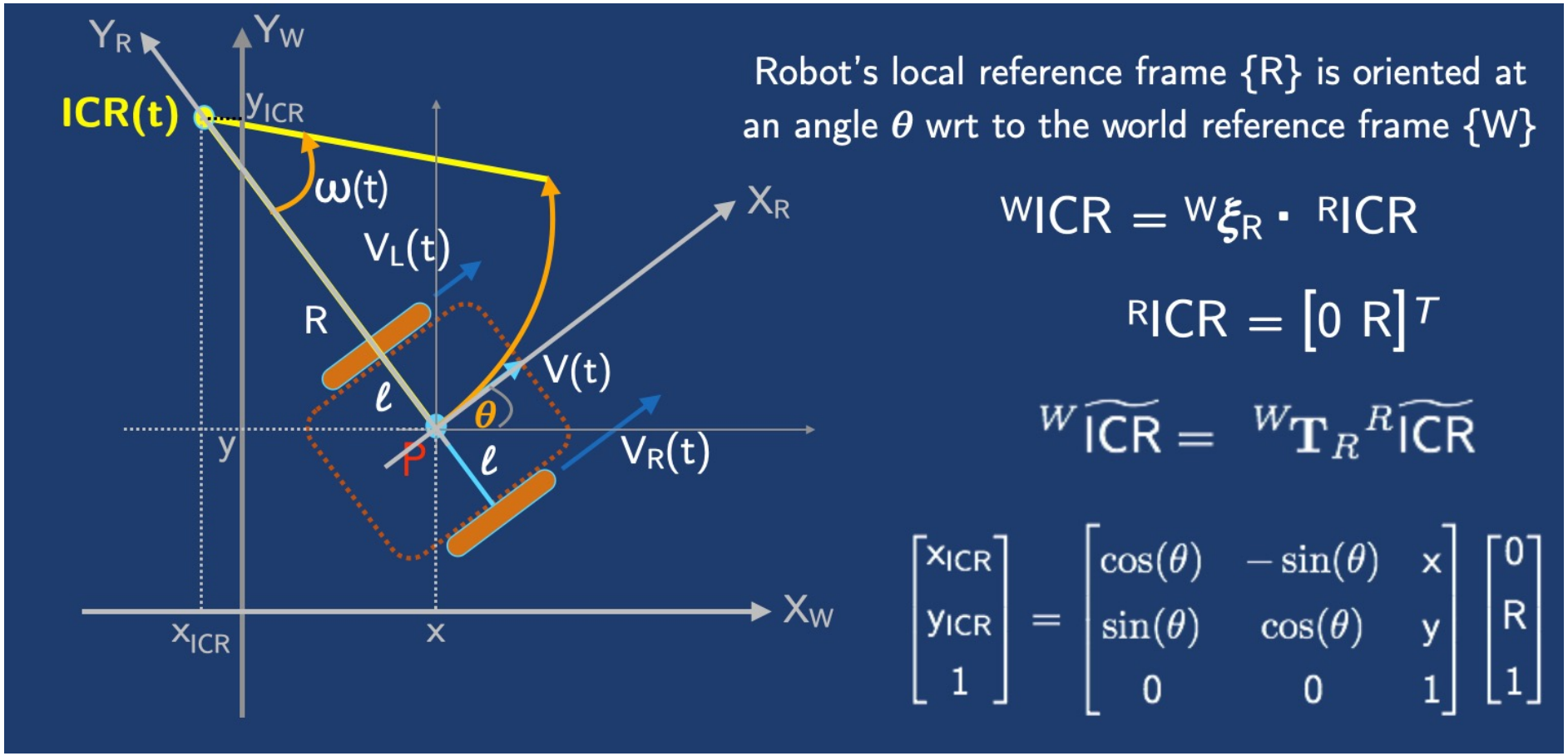


Once set the steering for the left wheel, the right wheel is constrained by rolling motion to steer a specific angle which is coherent with the vehicle's ICR

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

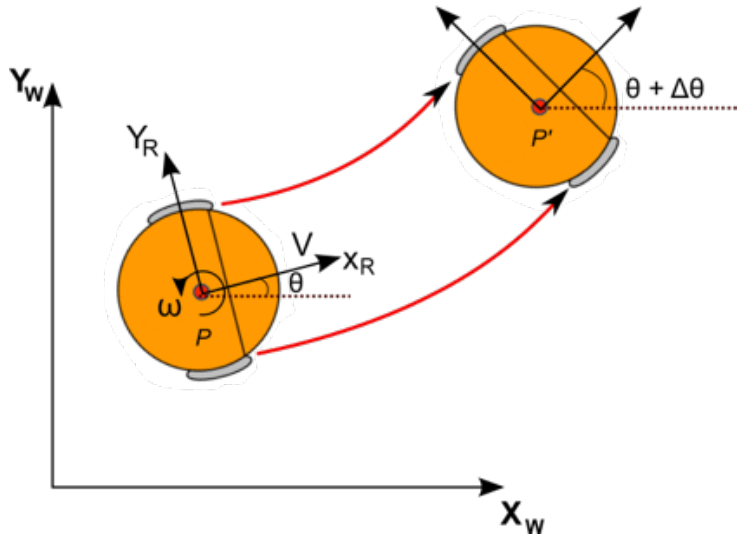
$$R(t) + \ell = \frac{L}{\tan(\alpha_r(t))}$$

Reference frames and position of the ICR



$$\begin{bmatrix} x_{ICR} \\ y_{ICR} \end{bmatrix} = \begin{bmatrix} x - R\sin(\theta) \\ y + R\cos(\theta) \end{bmatrix}$$

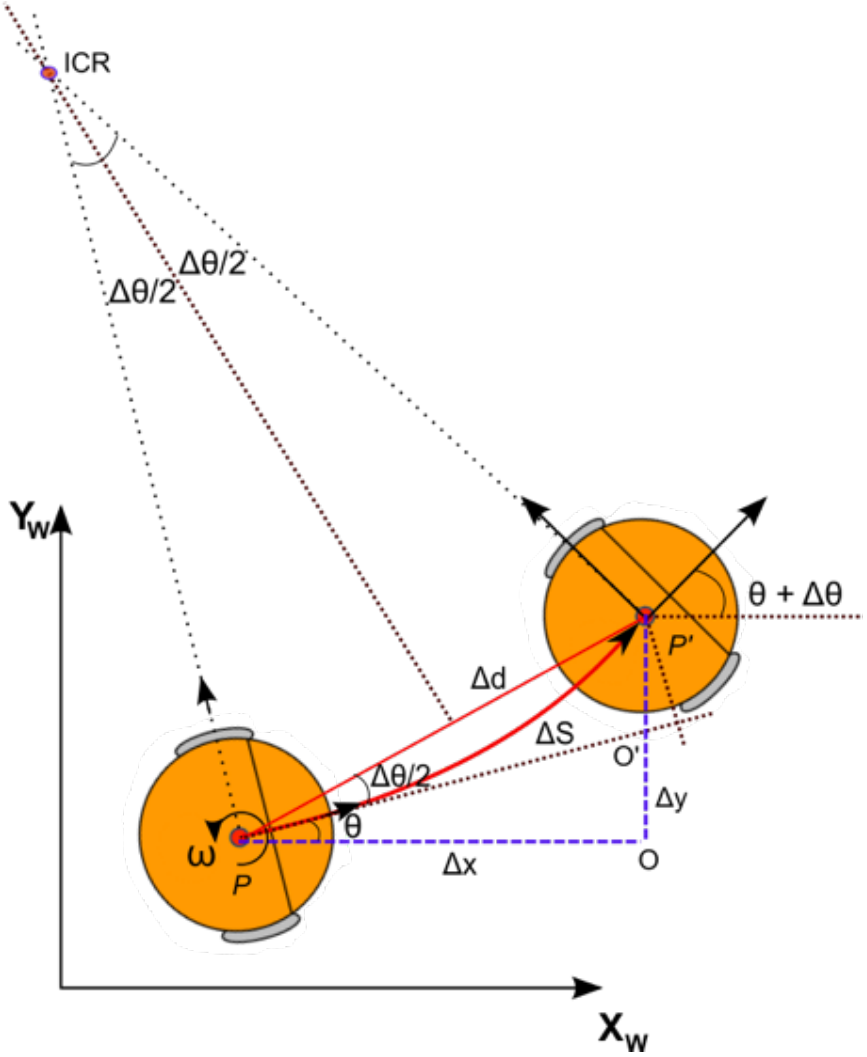
Robot pose evolution as a function of ICR



At a time t, an instantaneous motion of duration δt results in an infinitesimal change in orientation equal to $\Delta\theta$, and in an infinitesimal displacement ΔS:

what is the robot pose ${}^{W}\xi_{R}$ at time $(t + \delta t)$?

The ICR will not change, and the new pose is the result of a rotation $\Delta\theta = \omega\delta t$ of the robot about the ICR (ω is constant during the infinitesimal interval)



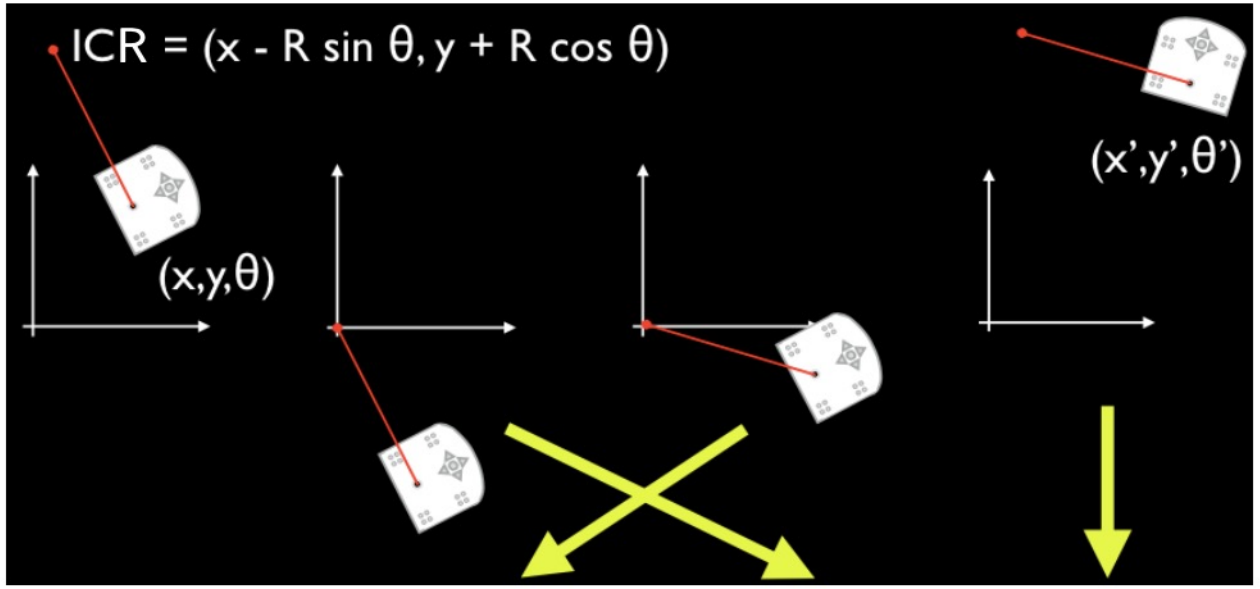
(1) translation of the ICR at {W} origin, (2) rotation of $\Delta\theta$, (3) translation back to the ICR

Robot pose evolution as a function of ICR

Motion of a robot rotating a distance R about its ICR with an angular velocity of ω

(1) translation of the robot, positioning the ICR at {W} origin
(2) rotation in place of $\Delta\theta = \omega\delta t$
(3) translation back of the ICR at its initial position

Equation valid for any mobile robot!



$$W\begin{bmatrix} x(t+\delta t) \\ y(t+\delta t) \\ \theta(t+\delta t) \end{bmatrix} = \begin{bmatrix} \cos(\omega\delta t) & -\sin(\omega\delta t) & 0 \\ \sin(\omega\delta t) & \cos(\omega\delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x(t) - x_{ICR}(t) \\ y(t) - y_{ICR}(t) \\ \theta(t) \end{bmatrix} + \begin{bmatrix} x_{ICR}(t) \\ y_{ICR}(t) \\ \omega\delta t \end{bmatrix}$$

✓ Based on the velocity inputs to the right and left wheels, robot's pose can be computed

Forward kinematic equations

$$W\begin{bmatrix} x(t+\delta t) \\ y(t+\delta t) \\ \theta(t+\delta t) \end{bmatrix} = \begin{bmatrix} \cos(\omega\delta t) & -\sin(\omega\delta t) & 0 \\ \sin(\omega\delta t) & \cos(\omega\delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x(t) - x_{ICR}(t) \\ y(t) - y_{ICR}(t) \\ \theta(t) \end{bmatrix} + \begin{bmatrix} x_{ICR}(t) \\ y_{ICR}(t) \\ \omega\delta t \end{bmatrix}$$

$$W\begin{bmatrix} x(t+\delta t) \\ y(t+\delta t) \\ \theta(t+\delta t) \end{bmatrix} = \begin{bmatrix} \big(x(t) - x_{ICR}(t)\big)\cos(\omega\delta t) - \big(y(t) - y_{ICR}(t)\big)\sin(\omega\delta t) + x_{ICR}(t) \\ \big(x(t) - x_{ICR}(t)\big)\sin(\omega\delta t) + \big(y(t) - y_{ICR}(t)\big)\cos(\omega\delta t) + y_{ICR}(t) \\ \theta(t) + \omega\delta t \end{bmatrix}$$

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

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

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

Forward kinematic equations

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

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

Function of the ICR

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)

Dead (Deduced) Reckoning

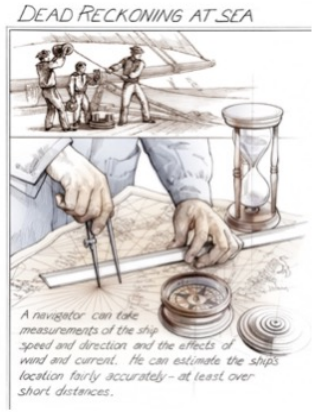
- In absence of an external infrastructure (e.g., GPS + Filters + Cameras) able to **track** the pose of the robot, **numerical integration** can be used, based on:
  - **Kinematic model** of the robot
  - Knowledge of the issued velocity commands, $v(t), \omega(t)$

→ **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

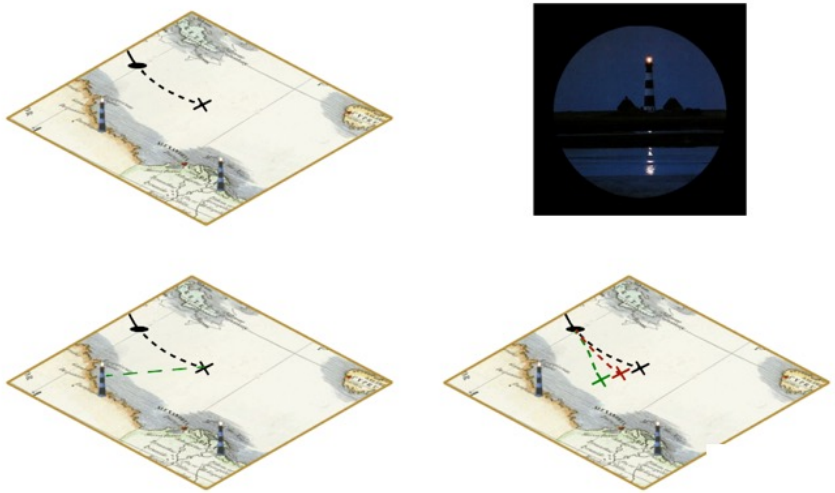


- ***Odometry*** is "basically" another way to say the same thing, that has different roots and etymology ...

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

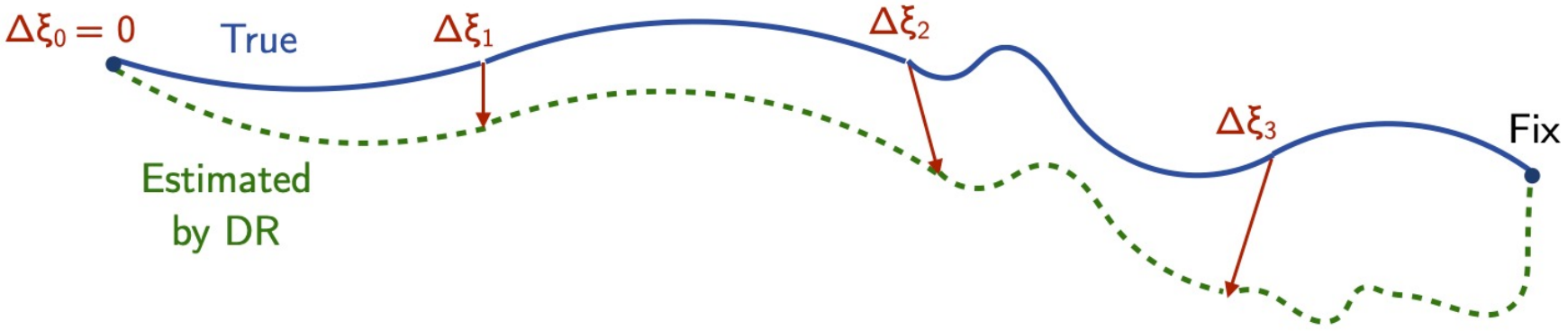
- The uncertainty of dead reckoning / odometry increases over time and maybe over distance ➔ A new fix is intermittently needed to determine a more reliable position from which a new dead reckoning process (i.e., integration) can be restarted

- 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!

Error propagation

- 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

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

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

Error propagation

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 $N \to \infty$

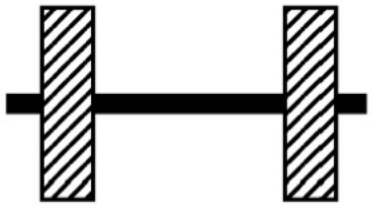
Dead Reckoning / Odometry: causes of error

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

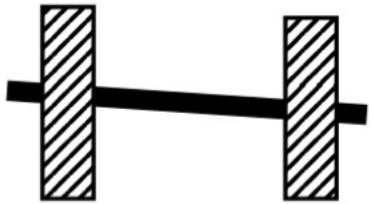
- *Numerical integration* errors (approximations, floating point rounding,....)
- Error readings from the *wheel encoders*
- Wheel *slippage* (friction issues, uneven ground, ...)
- Inaccurate measurement or calibration of *wheel and chassis parameters*, that are reflected in inaccuracies in the results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

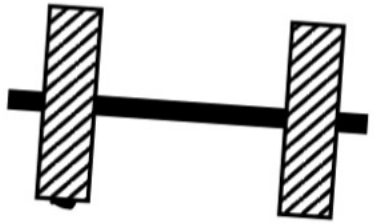
Dead Reckoning / Odometry: issues with wheels



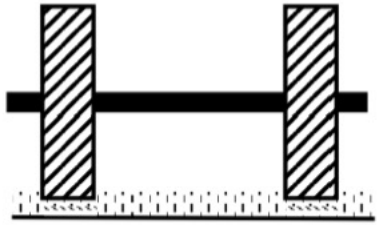
ideal case



different wheel diameters



bump

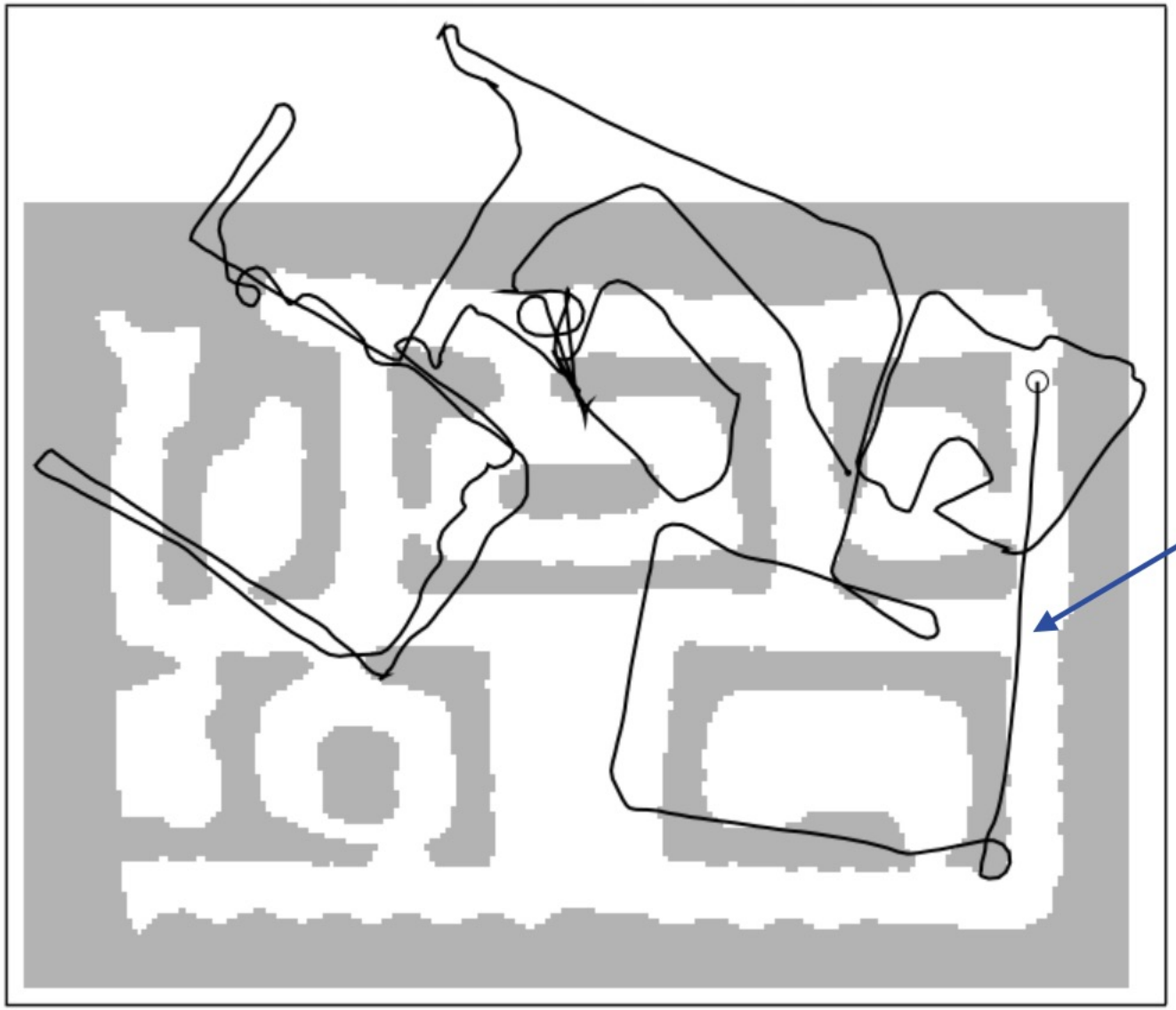


carpet

and many more ...

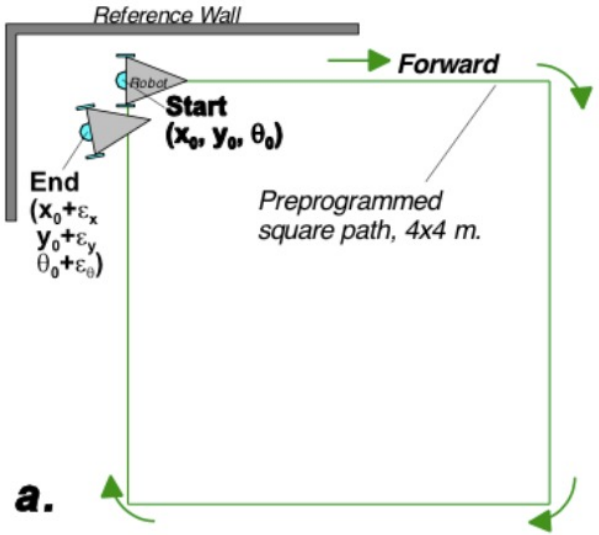
Using wheels' measures to compute odometry is not perfect at all, it comes with a number of *uncertainties*!

A typical odometry-estimated path

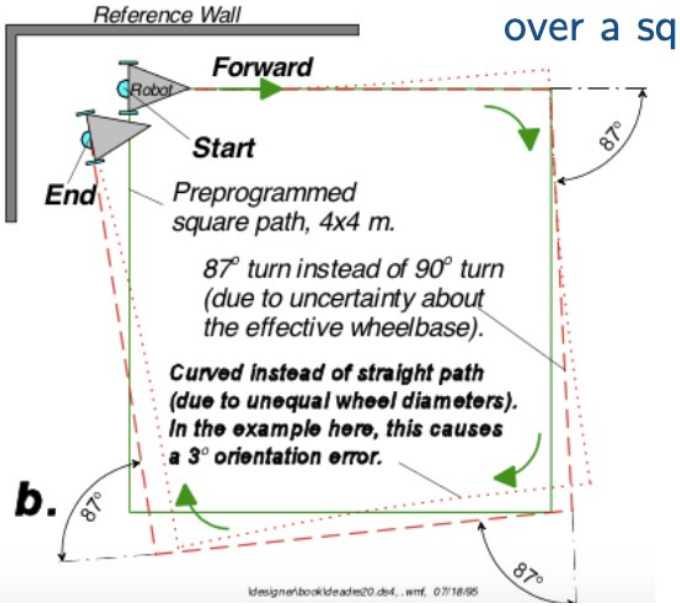


This the path the robot has estimated using odometry measures

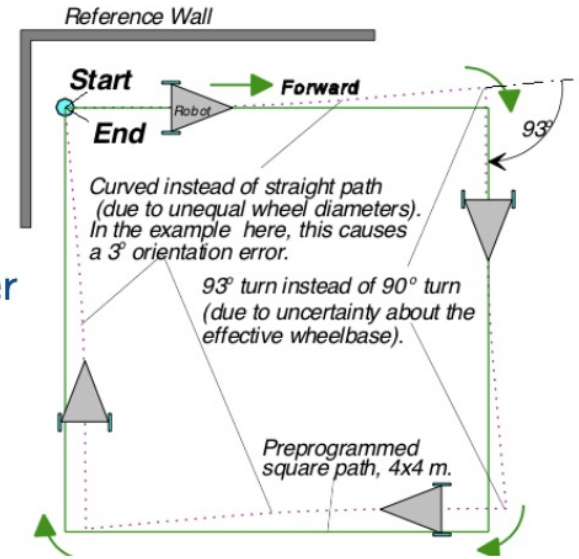
Interesting cases for odometry-based errors



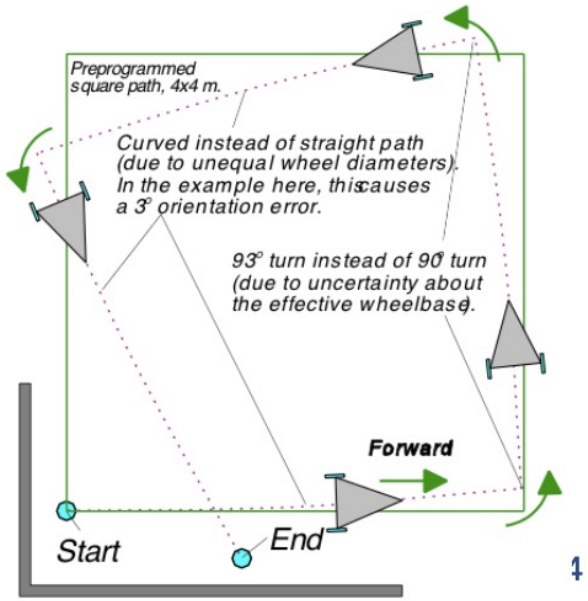
Cumulative error over a squared path



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

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

- 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 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

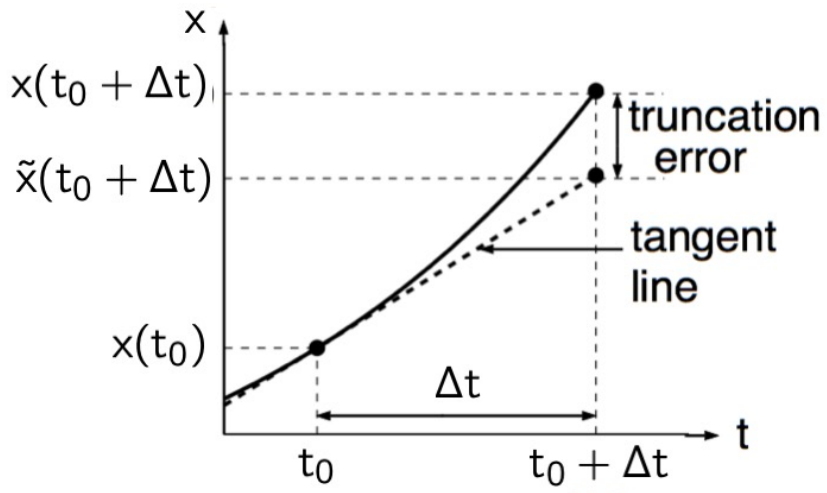
$$\dot{x} = v(t)\cos(\theta(t))$$
$$\dot{y} = v(t)\sin(\theta(t))$$
$$\dot{\theta} = \omega(t)$$

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

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

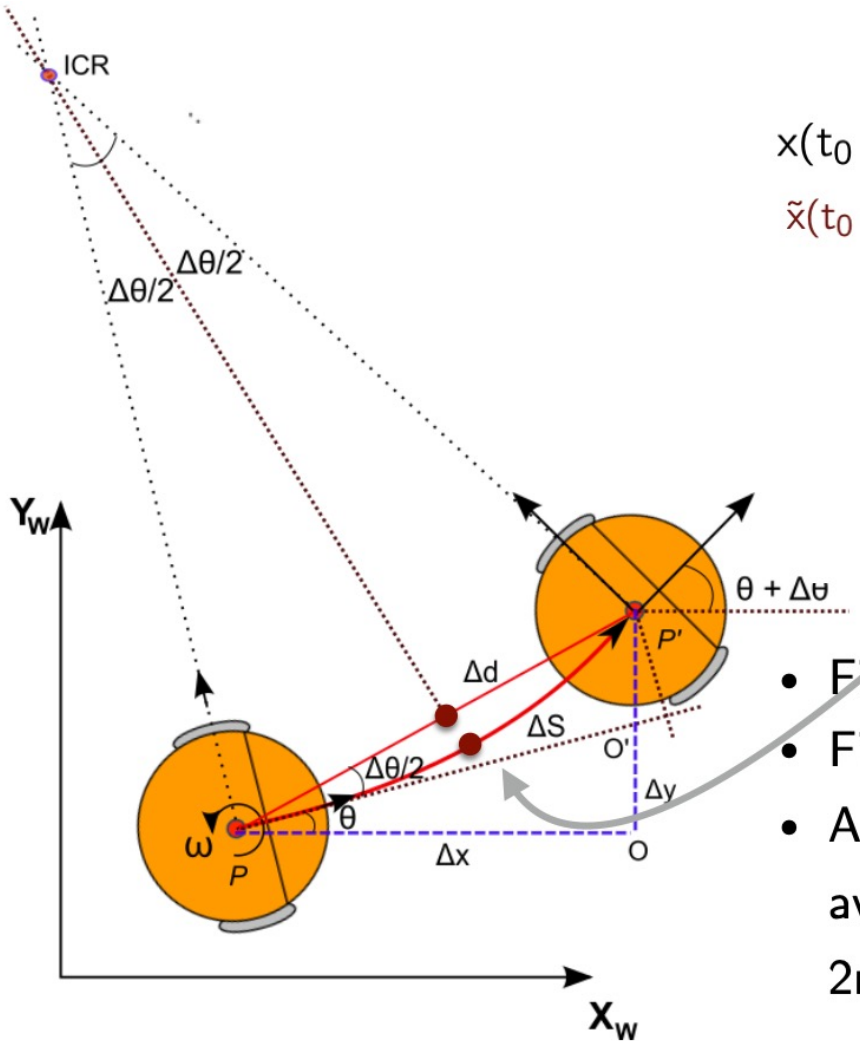
$$x(t_0+\Delta t) = x(t_0) + \Delta t\, \dot{x}(t_0) + \frac{(\Delta t)^2}{2!}\ddot{x}(t_0) + \frac{(\Delta t)^3}{3!}\dddot{x}(t_0) + \ldots$$

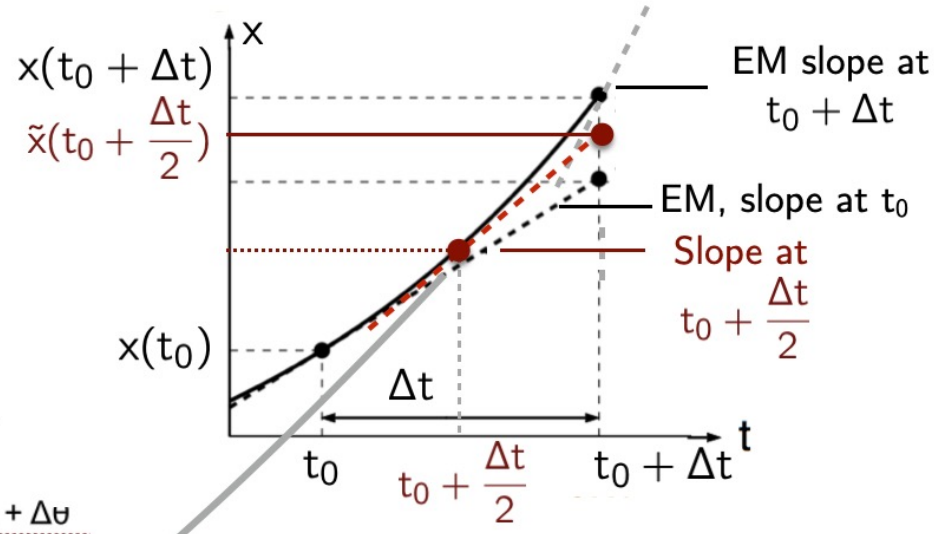
$$\tilde{x}(t_0+\Delta t) = x(t_0) + \Delta t\, \dot{x}(t_0)$$



- 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

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

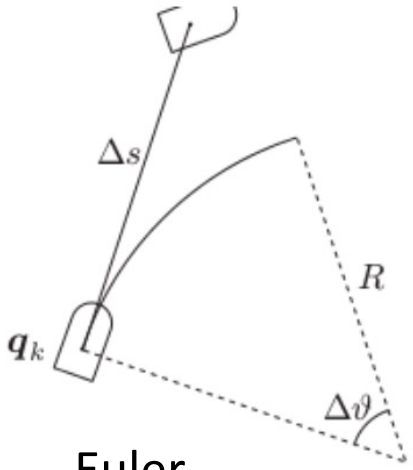
Exact numeric integration

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

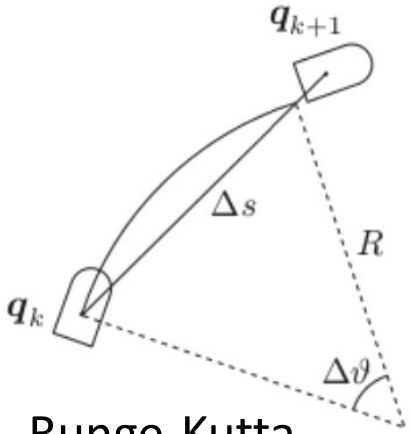
Euler numeric integration

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

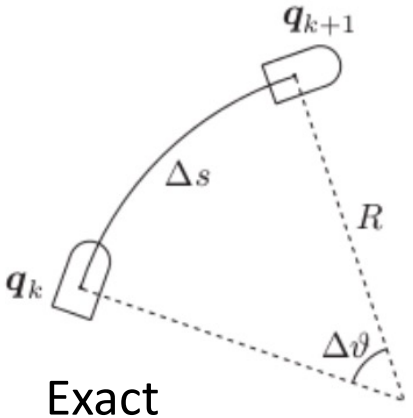
Runge-Kutta numeric integration



Euler



Runge-Kutta



Exact