اصول علم ربات – اسلاید هشتم

Fundamentals of Robotics – Slide 08

Kinematics 2

دکتر مهدی جوانمردی

زمستان ۱۴۰۰ – بهار ۱۴۰۱

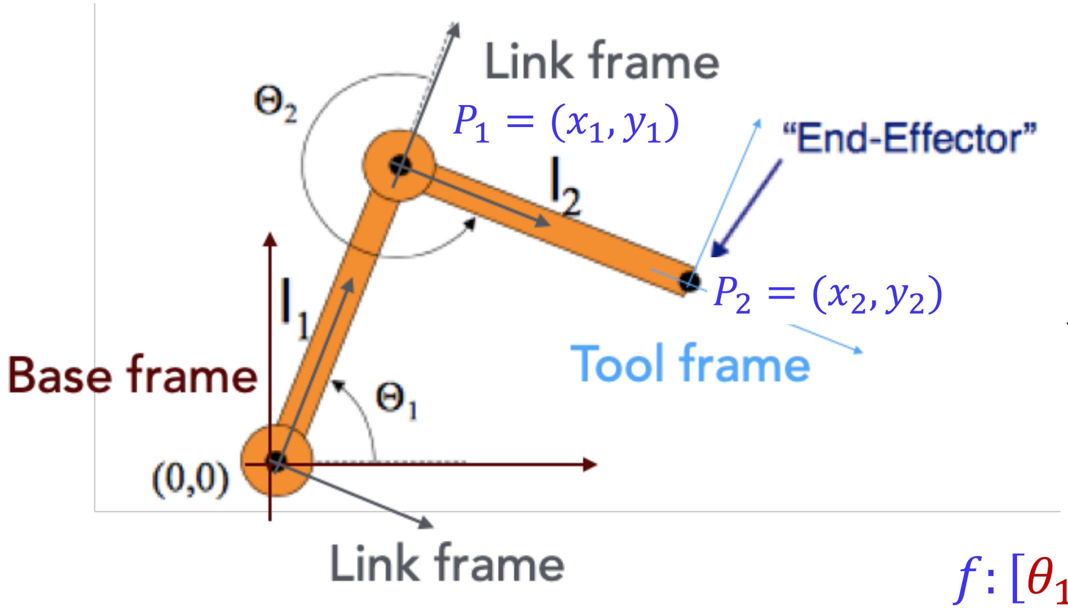
[slides adapted from Gianni Di Caro, @CMU with permission]

Forward kinematics: relationship f between controls, q, and pose

Forward kinematics equations:

$$\boldsymbol{r} = f(q)$$

$$^{I}\boldsymbol{\xi}_R \sim \boldsymbol{r} = \begin{bmatrix} x \\ y \\ z \\ \varphi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} f_x(q) \\ f_y(q) \\ f_z(q) \\ f_\varphi(q) \\ f_\theta(q) \\ f_\psi(q) \end{bmatrix}$$



$$\boldsymbol{r} = [x\ y\ \theta]^T$$

$$q = [\theta_1\ \theta_2]^T$$

$$f : [\theta_1\ \theta_2] \rightarrow [x\ y\ \theta]$$

$$f \equiv \begin{cases} f_x(\theta_1, \theta_2) = l_1 \cos\theta_1 + \ l_2 \cos(\theta_1 + \theta_2) \\ f_y(\theta_1, \theta_2) = l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2) \\ f_\theta(\theta_1, \theta_2) = \theta_1 + \theta_2 \end{cases}$$

Forward differential kinematics: changes in controls, $\dot{q}$, vs. velocity

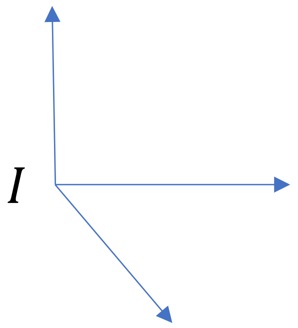
$$\underbrace{\nu = \begin{bmatrix} v \\ \omega \end{bmatrix}}_{\text{Twist}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} \dot{f}_x(q(t)) \\ \dot{f}_y(q(t)) \\ \dot{f}_z(q(t)) \\ \dot{f}_\varphi(q(t)) \\ \dot{f}_\theta(q(t)) \\ \dot{f}_\psi(q(t)) \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} & \cdots & \frac{\partial x}{\partial q_n} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} & \cdots & \frac{\partial y}{\partial q_n} \\ \frac{\partial z}{\partial q_1} & \frac{\partial z}{\partial q_2} & \cdots & \frac{\partial z}{\partial q_n} \\ \frac{\partial \omega_x}{\partial q_1} & \frac{\partial \omega_x}{\partial q_2} & \cdots & \frac{\partial \omega_x}{\partial q_n} \\ \frac{\partial \omega_y}{\partial q_1} & \frac{\partial \omega_y}{\partial q_2} & \cdots & \frac{\partial \omega_y}{\partial q_n} \\ \frac{\partial \omega_z}{\partial q_1} & \frac{\partial \omega_z}{\partial q_2} & \cdots & \frac{\partial \omega_z}{\partial q_n} \end{bmatrix}}_{\textbf{Jacobian}\text{ matrix, } J(q)} \underbrace{\begin{bmatrix} \frac{dq_1}{dt} \\ \frac{dq_2}{dt} \\ \vdots \\ \vdots \\ \frac{dq_n}{dt} \end{bmatrix}}_{\dot{q}} = J(q)\dot{q}$$

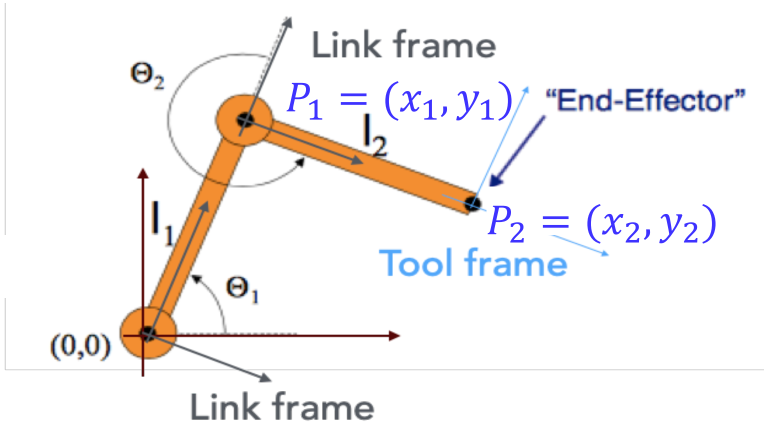
$${}^{I}\dot{\xi}_R \sim \upsilon = [v \quad \omega]^T = J(q) \cdot [\dot{q}_1 \quad \dot{q}_2 \quad \cdots \quad \dot{q}_n]^T = J(q) \cdot \dot{q}$$

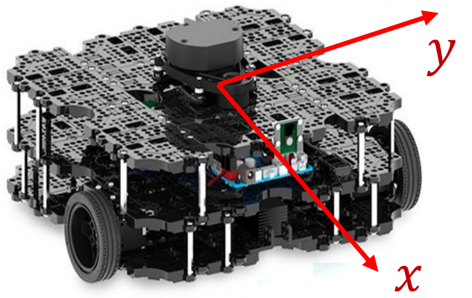
Computing Pose: Robotic manipulators vs. Mobile robots

- **Question**: do we need the *forward differential kinematics* model to know the pose ${}^{I}\xi_R$ of our robot in I?

  - Pose of end-effector for a robotic manipulator
  - Pose of a robot reference frame for a mobile (single body) robot







E.g., from $\dot{x} = F(q(t))$

$$x(t) = x(0) + \int_0^t F(q(t))dt$$

...the same for the other components of the pose

F defined by J

Computing Pose: Robotic manipulators vs. Mobile robots

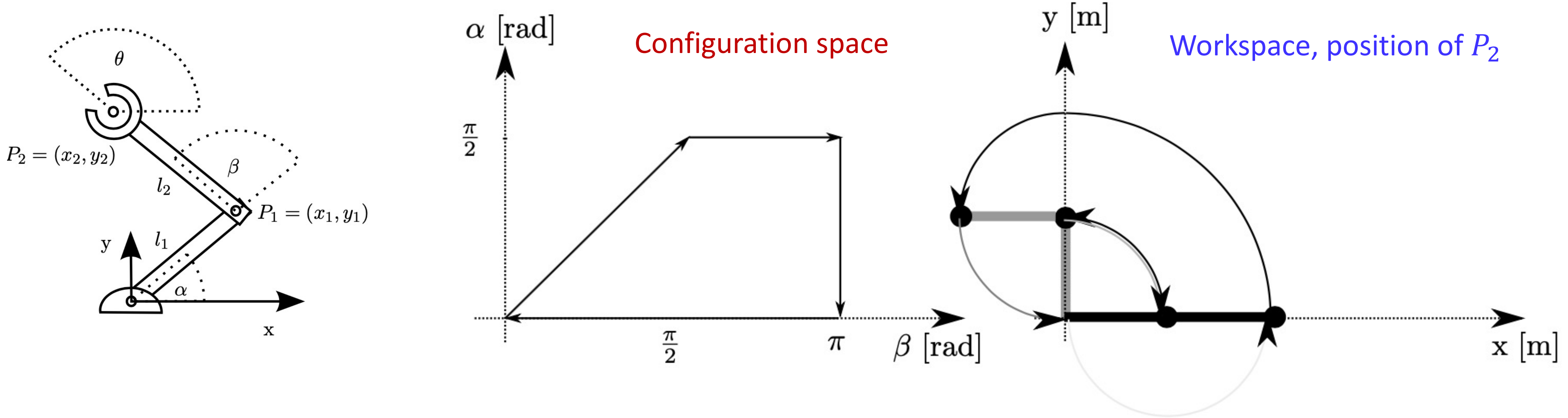
- ✓ **Manipulator**: configuration q in the joint space + forward kinematic equations, $f(q) \rightarrow$ Pose
  - ✓ We don't strictly need to write a forward differential model and integrate it to get the pose
  - It's sufficient to read joint status from motor encoders and plug the values in $f(q)$

- ○ **Mobile robot**: configuration q in the joint (wheels) space + forward kinematic equations, $f(q) \rightarrow$?
  - ➢ Usually (depending on wheels type and arrangement), we need to write a forward differential model and integrate it to get the pose

  \+ we need to deal with *uncertainties* in measurements

Computing Pose: Robotic manipulators

- ✓ **Manipulator**: configuration q in the joint space + forward kinematic equations, $f(q) \rightarrow$ Pose
  - ✓ We don't strictly need to write a forward differential model and integrate it to get the pose
  - It's sufficient to read joint status from motor encoders and plug the values in $f(q)$



- A **closed trajectory** in the joint space will position the robot's end-effector at the exact same initial position in operational space, robot's end-effector will make a closed trajectory also in the workspace
- Each pair of joint positions $[\alpha \ \beta]$ correspond to a **unique point** in both configuration space and workspace

✓ No need to integrate velocities to know position of end-effector!

(Computing Pose) Mobile robots

- **Mobile robot**: configuration q in the joint (wheels) space + forward kinematic equations, $f(q) \rightarrow$?
  - Usually, we need to write a forward differential model and integrate it to get the pose

  \+ we need to deal with *uncertainties* in measurements

- Mobile robots → **Ground** robots

Wheeled, Tracked

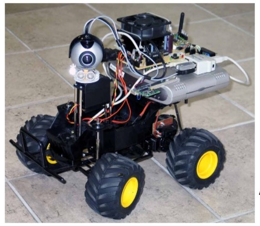
Robot's **maneuverability** depends on:

- **Geometry**: Relative placement of wheels/tracks on robot's chassis
- **Type**: degrees of freedom and constraints of each wheel/track

2- and 4-wheel
Differential driving

Tricycle

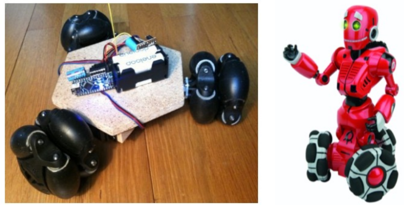
Murata Boy and Girl
Bicycle and Unicycle



Car-like
Ackermann

6-wheel
space rovers

4-wheel
steering



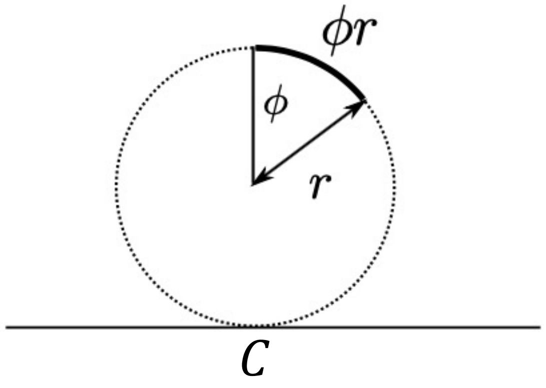
Tri-bots
Omniwheels

Tracked robots

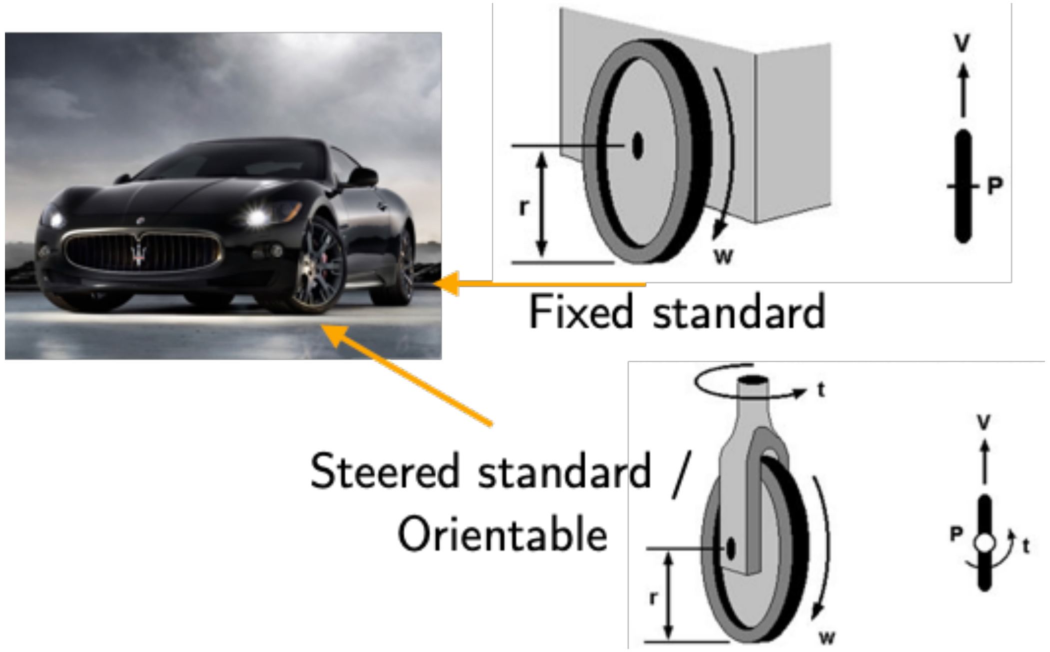
(Computing Pose) Mobile robots: wheels

✓ We focus on **wheeled** robots!

- **Configuration space of a wheeled robot**: configuration q of **all the wheels**, defined by:
  - angle of rotation (cumulative, if it's possible to measure it / integrate it)
  - angle of steering (if available)
  - …

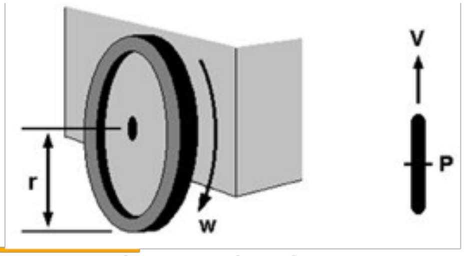


Fixed Standard wheel,
only rolling, no steering



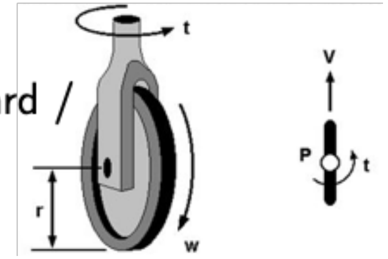
- q = angle of rotation ϕ
  - ➢ Configuration ϕ corresponds to a *motion* (for the robot's chassis in I) of length ϕr
  - ➢ r is wheel's radius

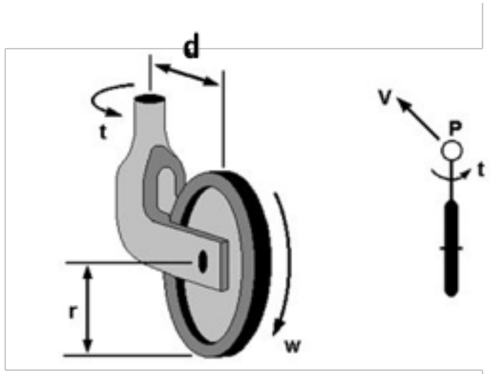
(Computing Pose) Mobile robots: wheel types



Fixed standard

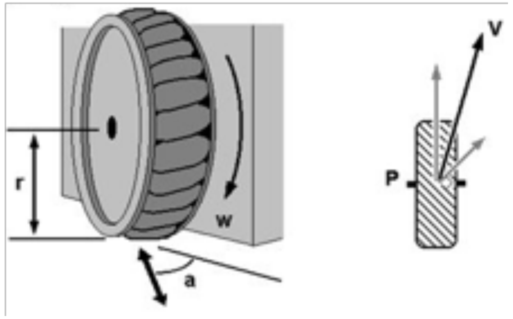
Steered standard / Orientable

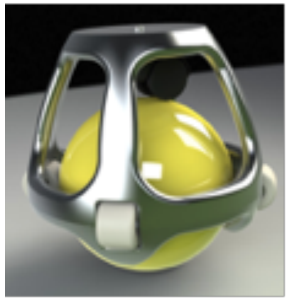


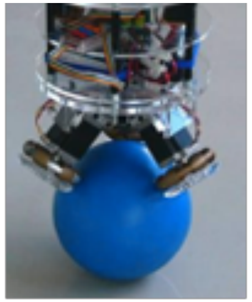


Castor / Off-centered orientable

Mecanum/Swedish

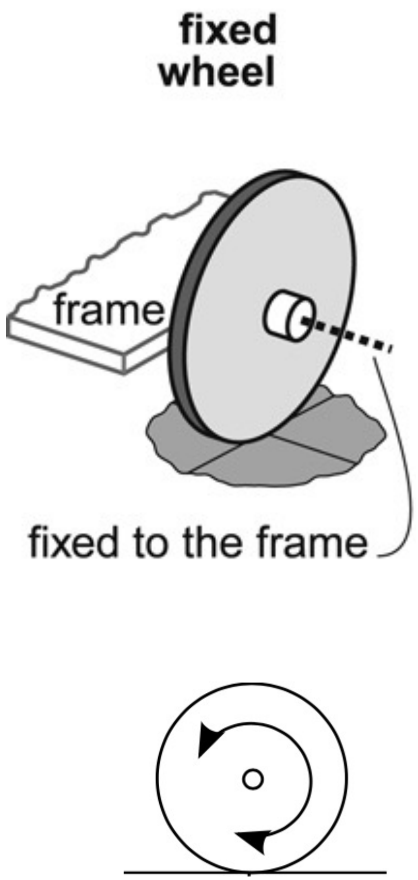






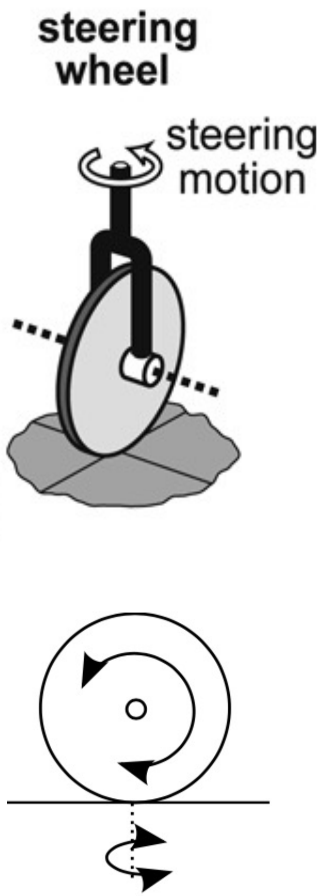
Spherical

(Computing Pose) Mobile robots: wheel types and DoF



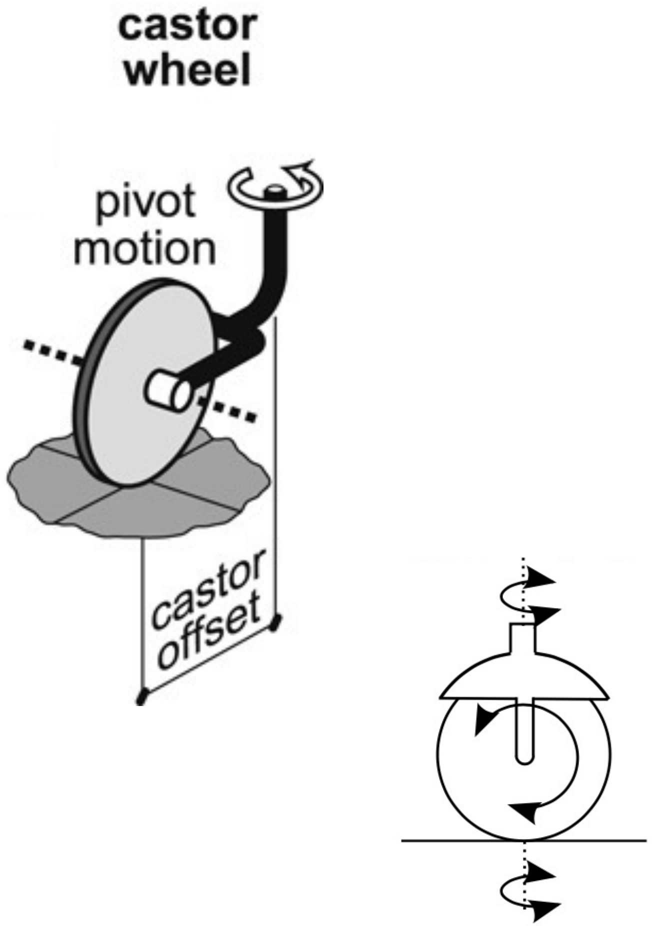
1 DoF:

- Rotation around the wheel axle



2 DoF:

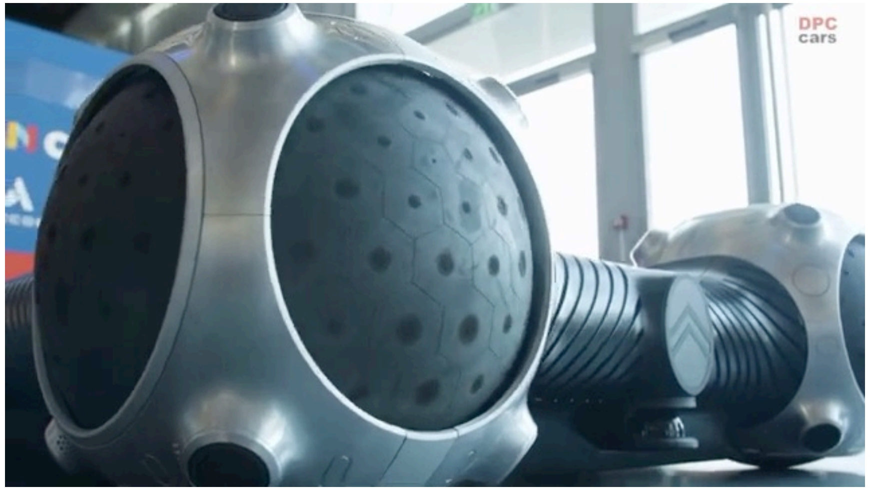
- Rotation around the wheel axle
- Rotation around its contact point with the ground

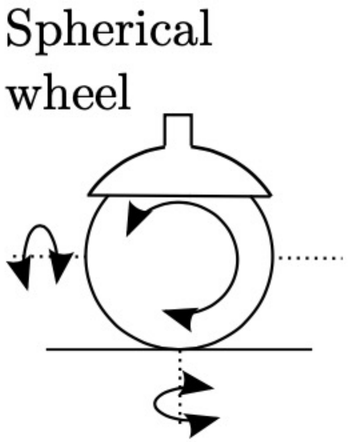


3 DoF:

- Rotation around the wheel axle
- Rotation around its contact point with the ground
- Rotation around the caster axis

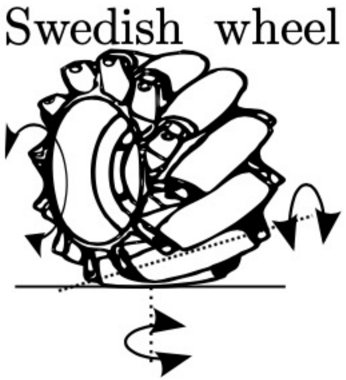
(Computing Pose) Mobile robots: wheel types and DoF





3 DoF:

- Rotation in any direction
- Rotation around its contact point



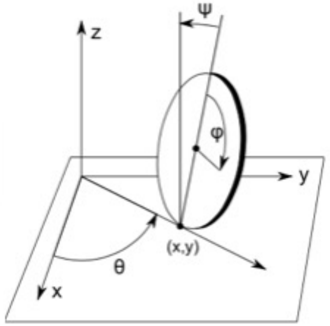
3 DoF:

- Rotation around the wheel axle
- Rotation around its contact point with the ground
- Rotation around the roller axles

(Computing pose) Mobile robots: Geometry, arrangement of wheels

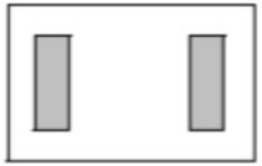
One wheel

Unicycle



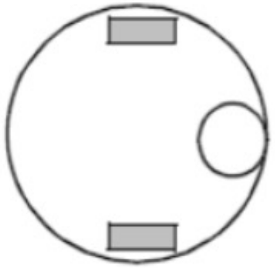
Two wheels

Bicycle

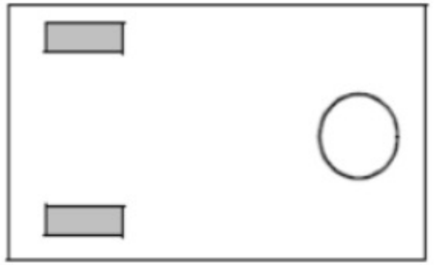


Differential drive

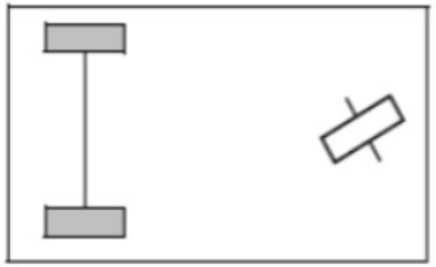
Three wheels



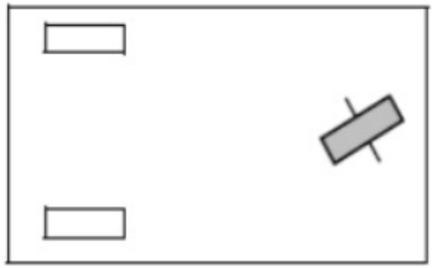
Differential with castor



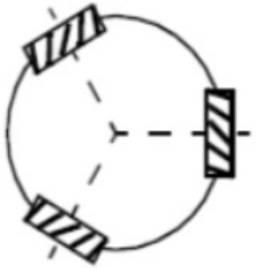
Differential cart with castor



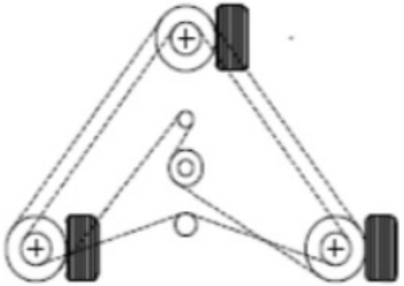
Tricycle



Tricycle - Horse buggy



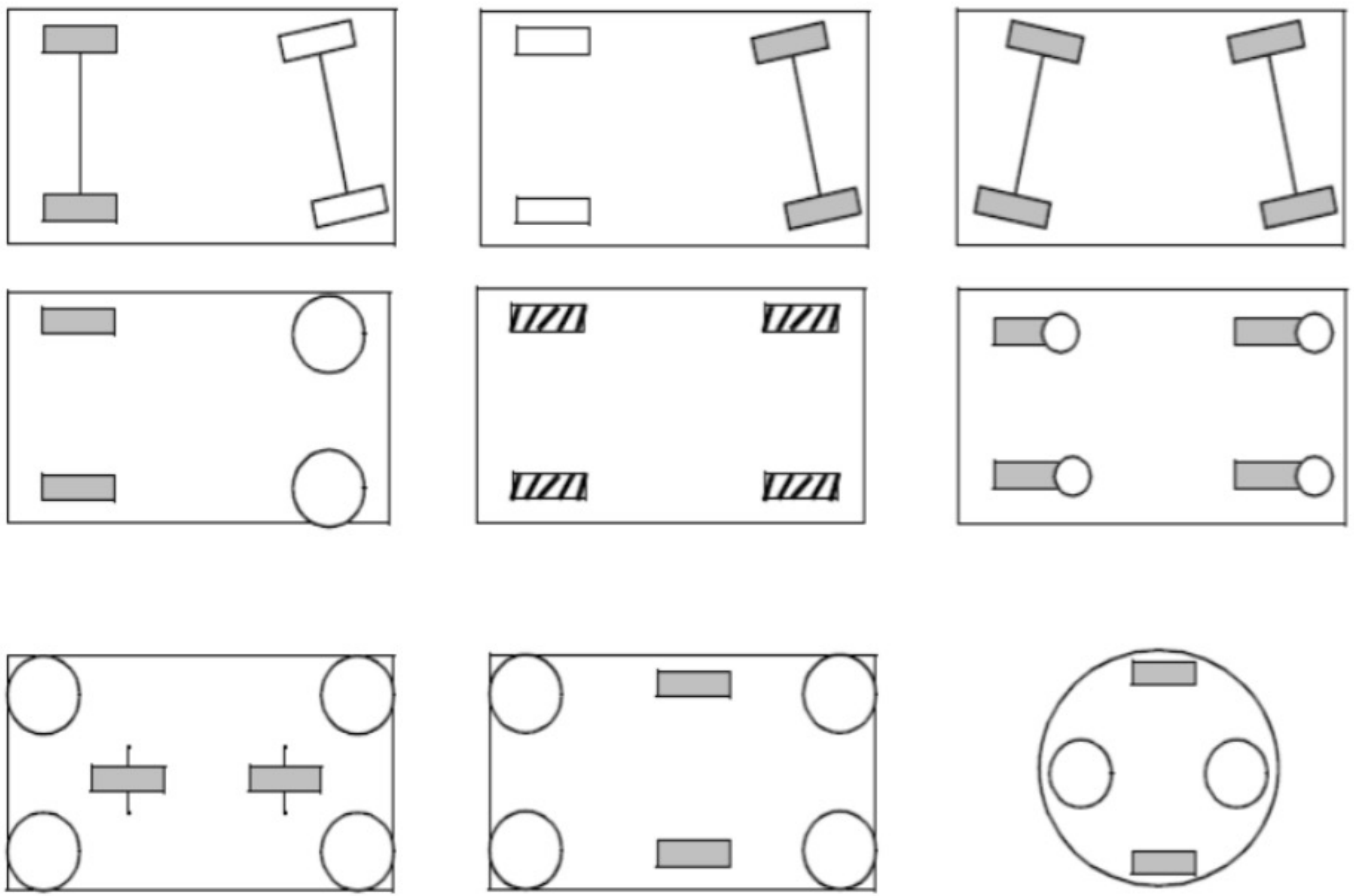
Omni drive



Synchro drive

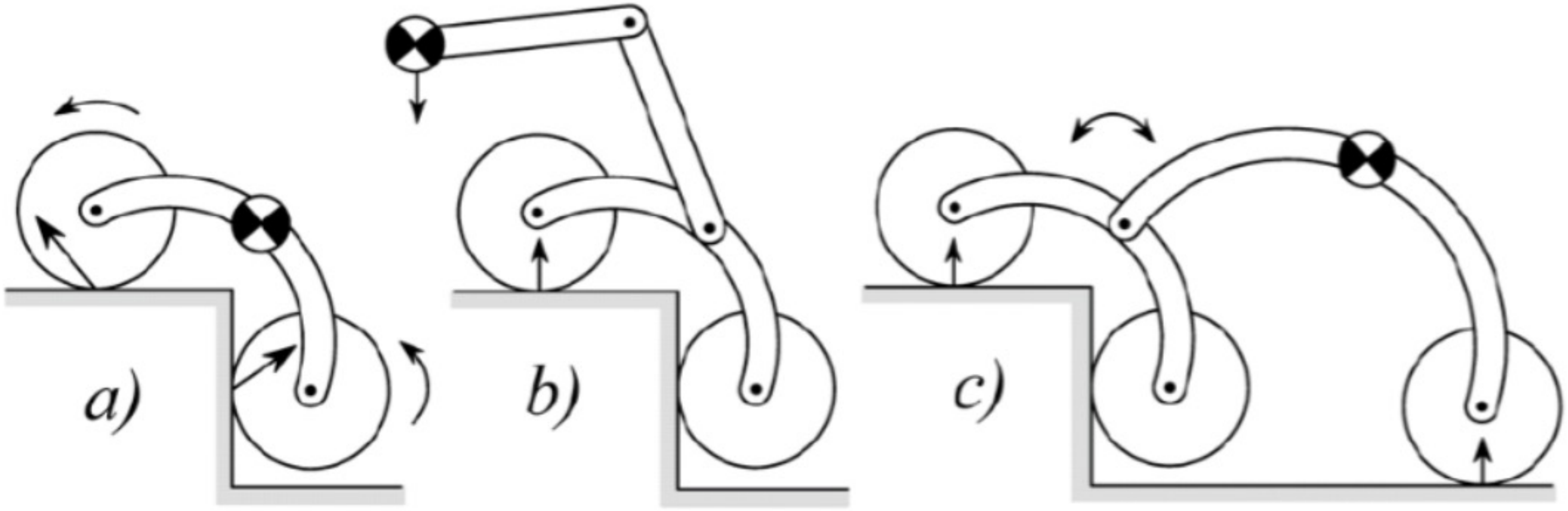
(Computing pose) Mobile robots: Geometry, arrangement of wheels

Four wheels



(Computing pose) Mobile robots: Geometry, arrangement of wheels

Rovers for climbing

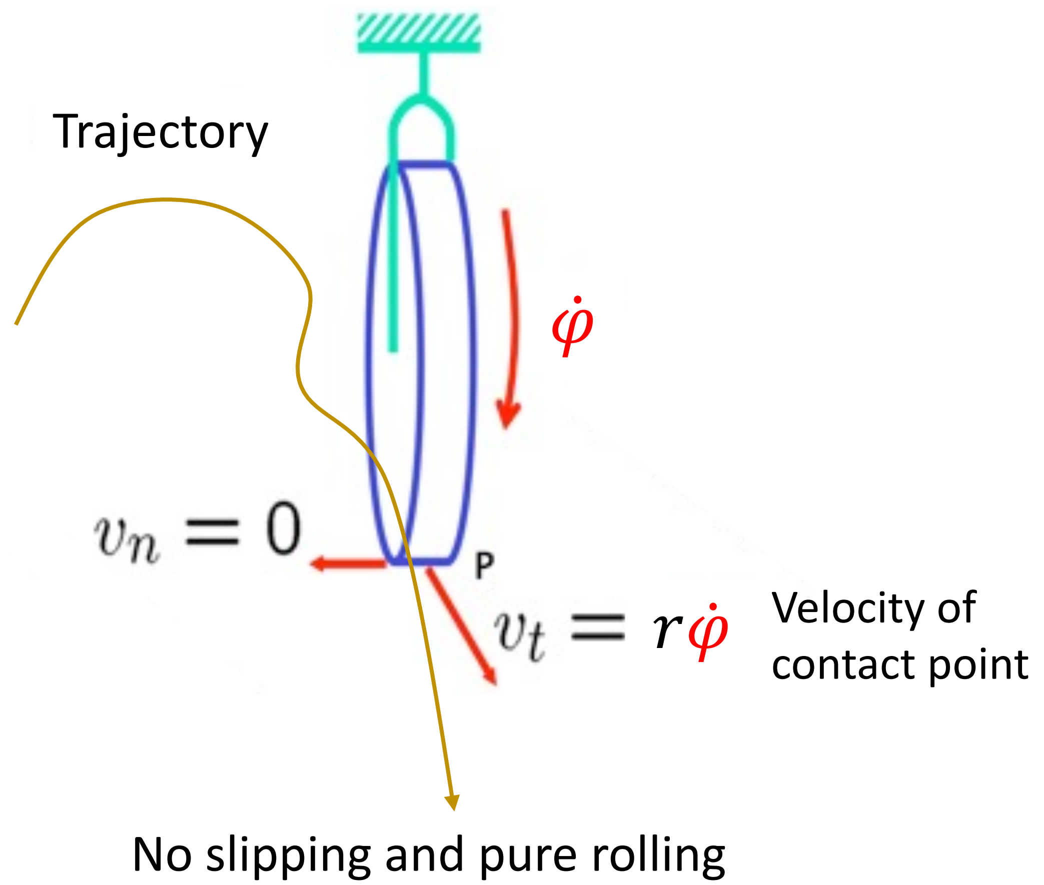


Purely friction based

Change of center of gravity (CoG)

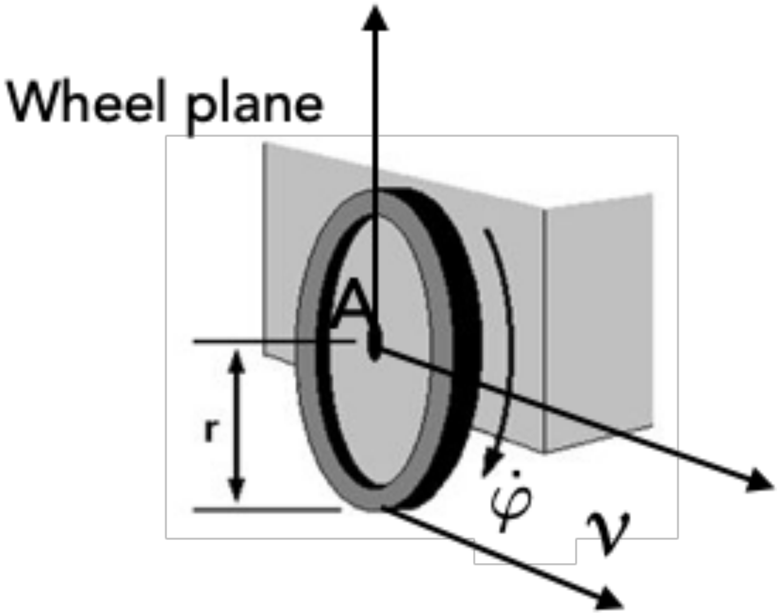
Adapted suspension mechanism with passive or active joints

(Computing pose) Mobile robots: Assumption on wheels (for kinematics)



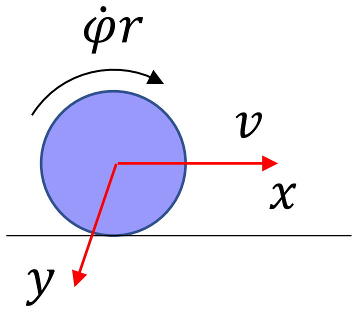
- The wheel plane must remain vertical at all times
- There's one single point of instantaneous contact between wheel and ground
- There's no sliding at the single point of contact
- Pure rolling: $v = 0$ at contact point
- No slipping, skidding, sliding
- No friction for rotation around contact points
- Steering axes are orthogonal to the surface
- Wheels are connected by a rigid frame (chassis)
- Movement is on a horizontal plane
- Wheels not deformable

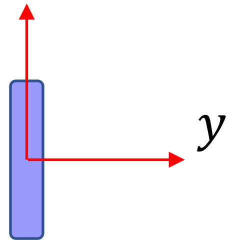
(Computing pose) Mobile robots: Assumption on wheels (for kinematics)



- **Rolling constraint** (pure rolling at the contact point): All motion along the direction of the wheel plane is determined by wheel spin, $\dot{\varphi}r$ (all motion is transferred)

- **Sliding constraint**: The component of the wheel's motion orthogonal to the wheel plane must be zero

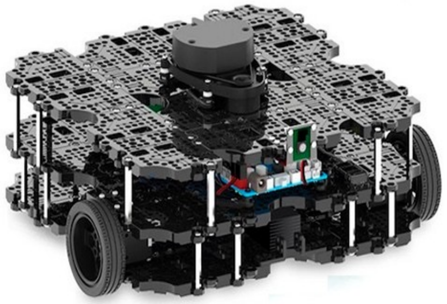


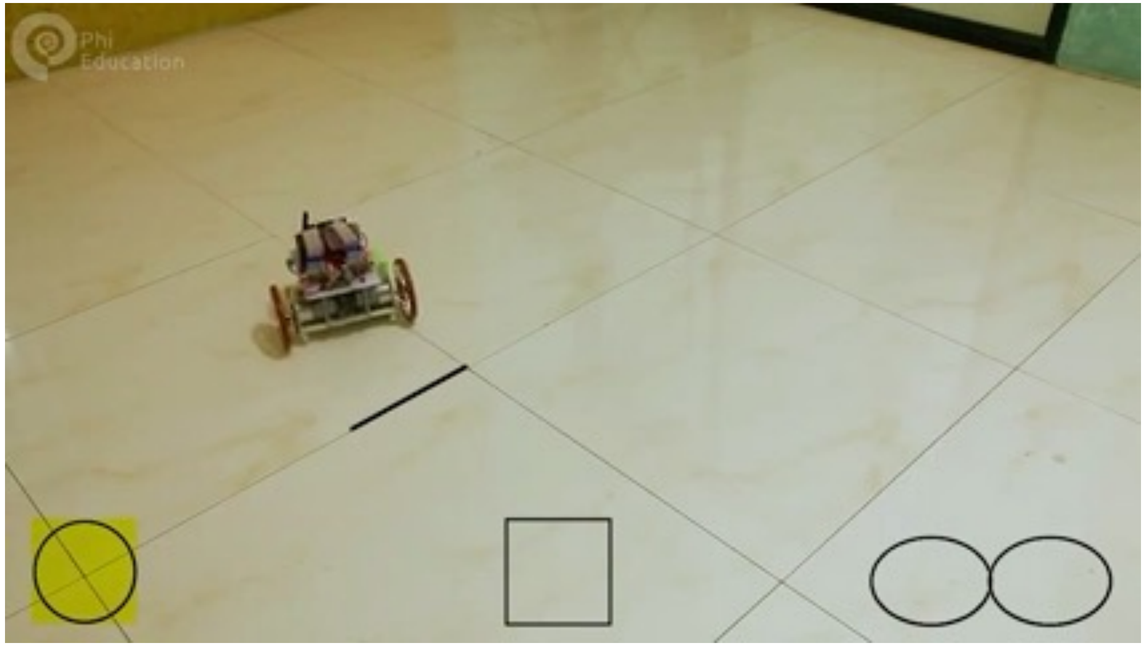


(Computing pose) Mobile robots: Differential (steering) robot

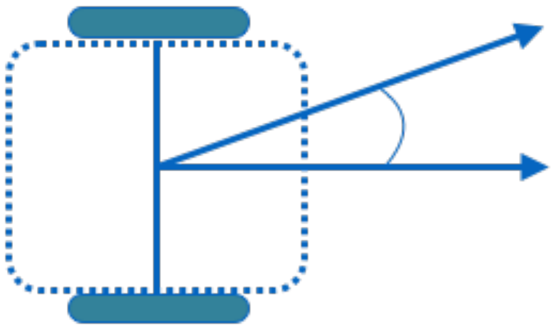
Differential steering (vehicle, robot):

One or more pairs of **standard wheels** mounted on a single axis, **independently** powered and controlled, providing both drive and steering functions through the motion difference between the wheels.

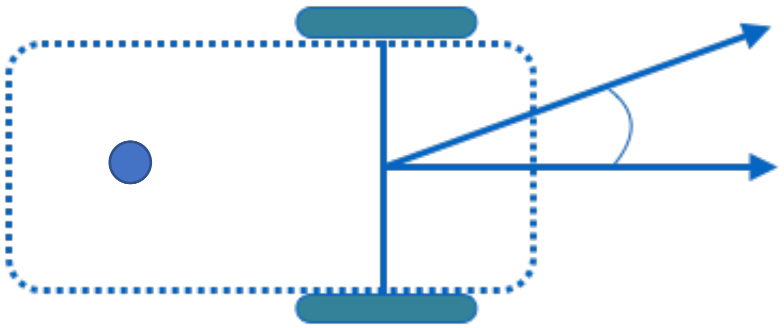




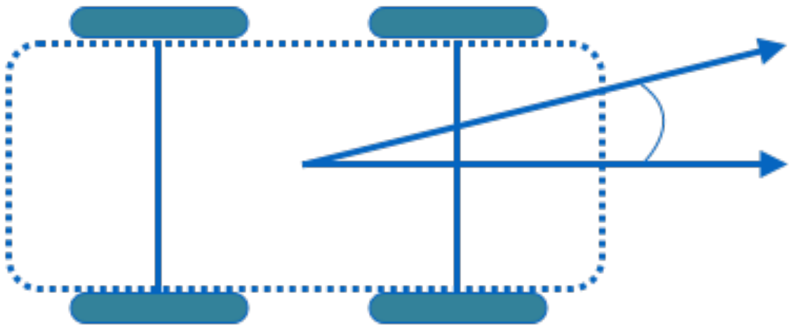
(Computing pose) Mobile robots: Differential (steering) robot



Any type of chassis



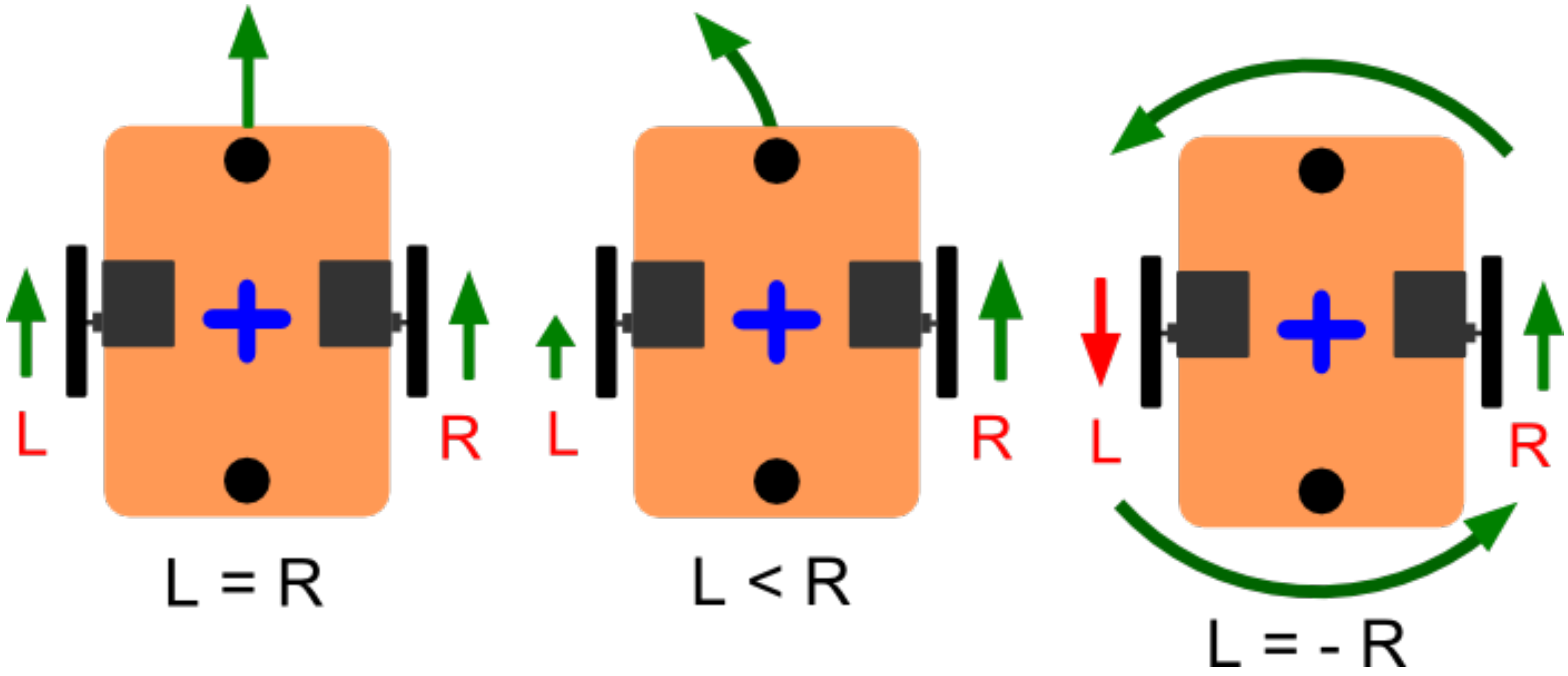
Additional passive wheels for stability can be added (e.g., caster wheels)



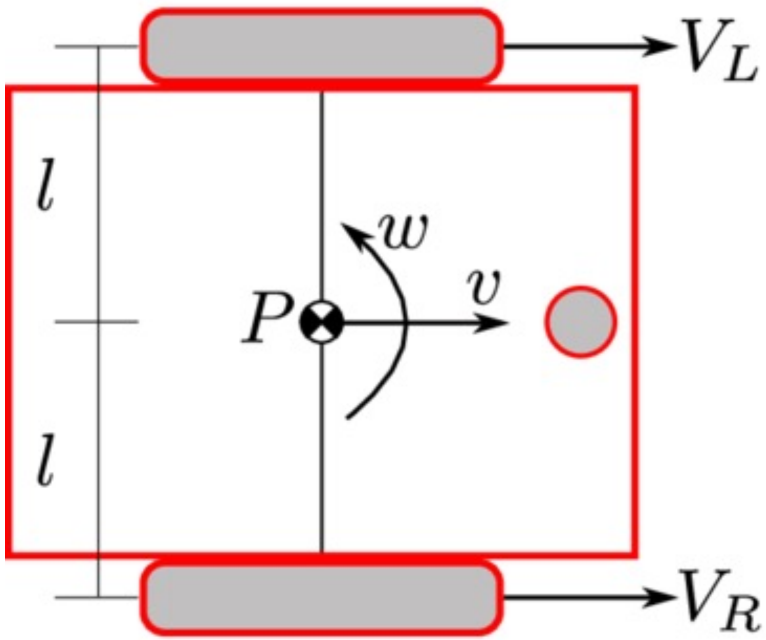
Total wheel pairs can be more than one, making control more complex

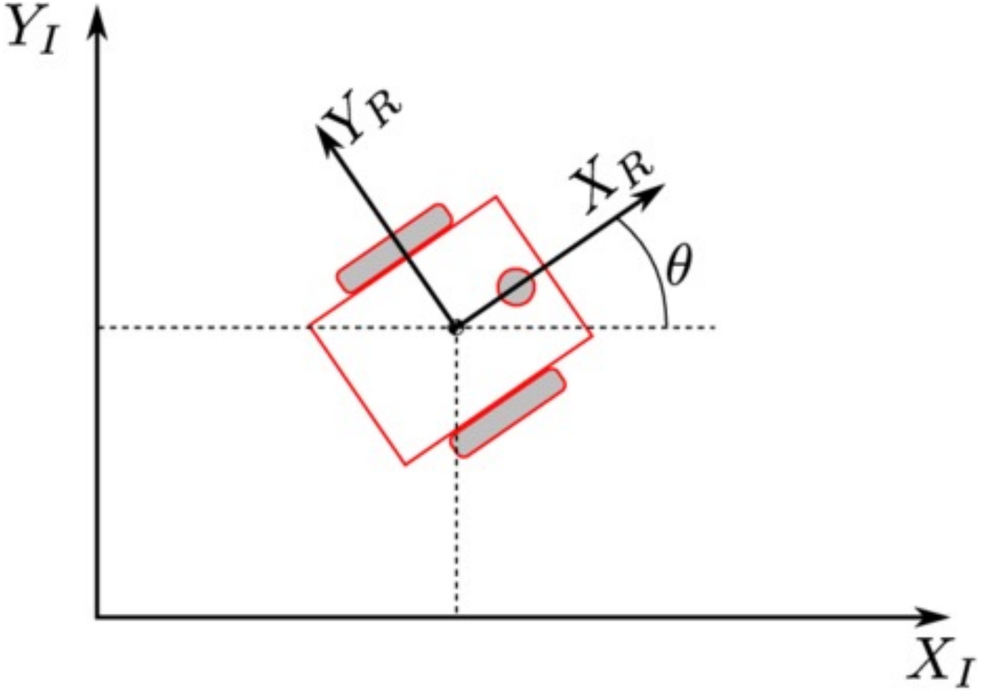
Differential drive: in automotive engineering, refers to the presence of a *differential gear* or related device to transfer different motion to the steering wheels on a same axis (e.g., the frontal wheels of a normal car)

(Computing pose) Mobile robots: Differential (steering) robot

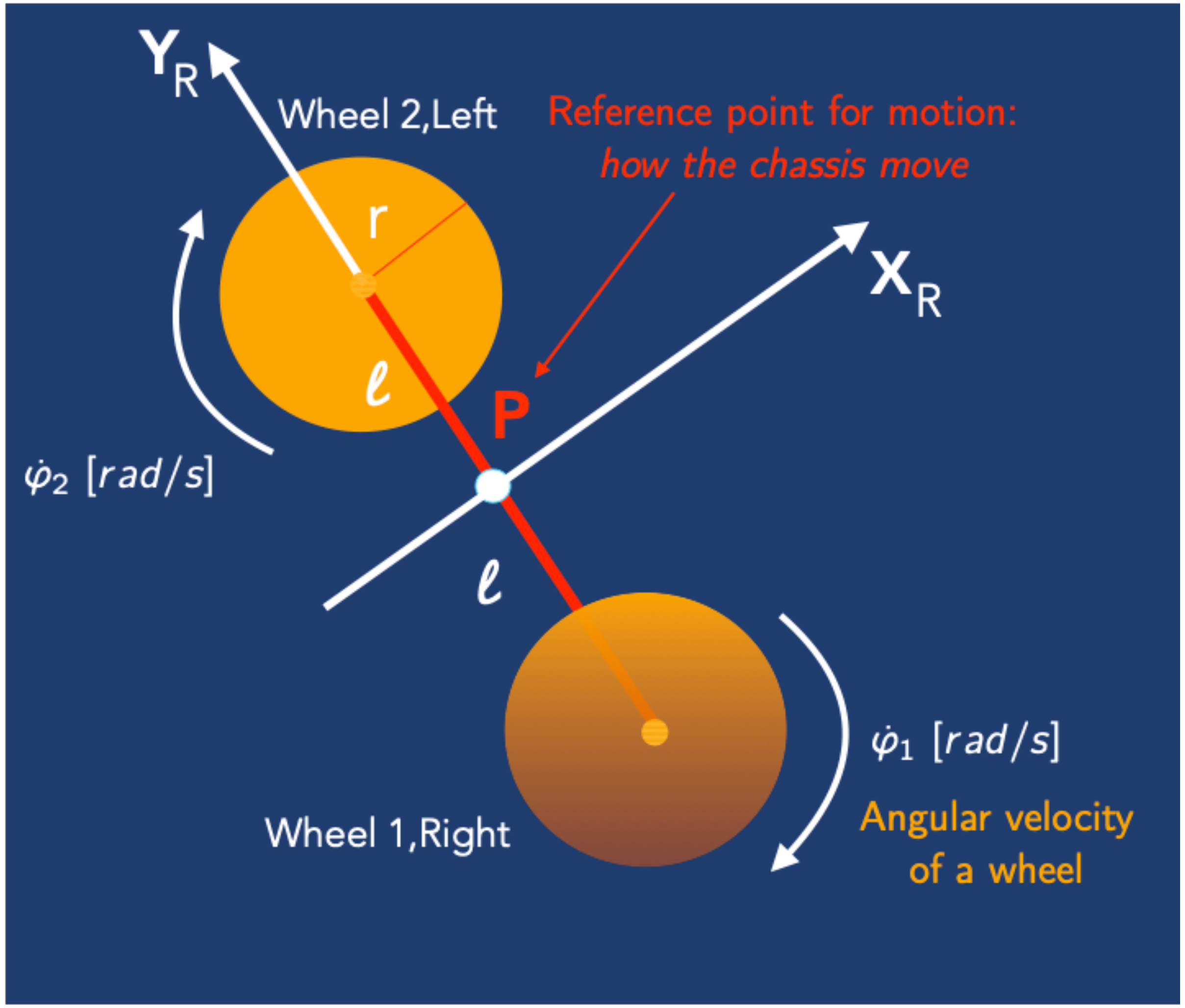


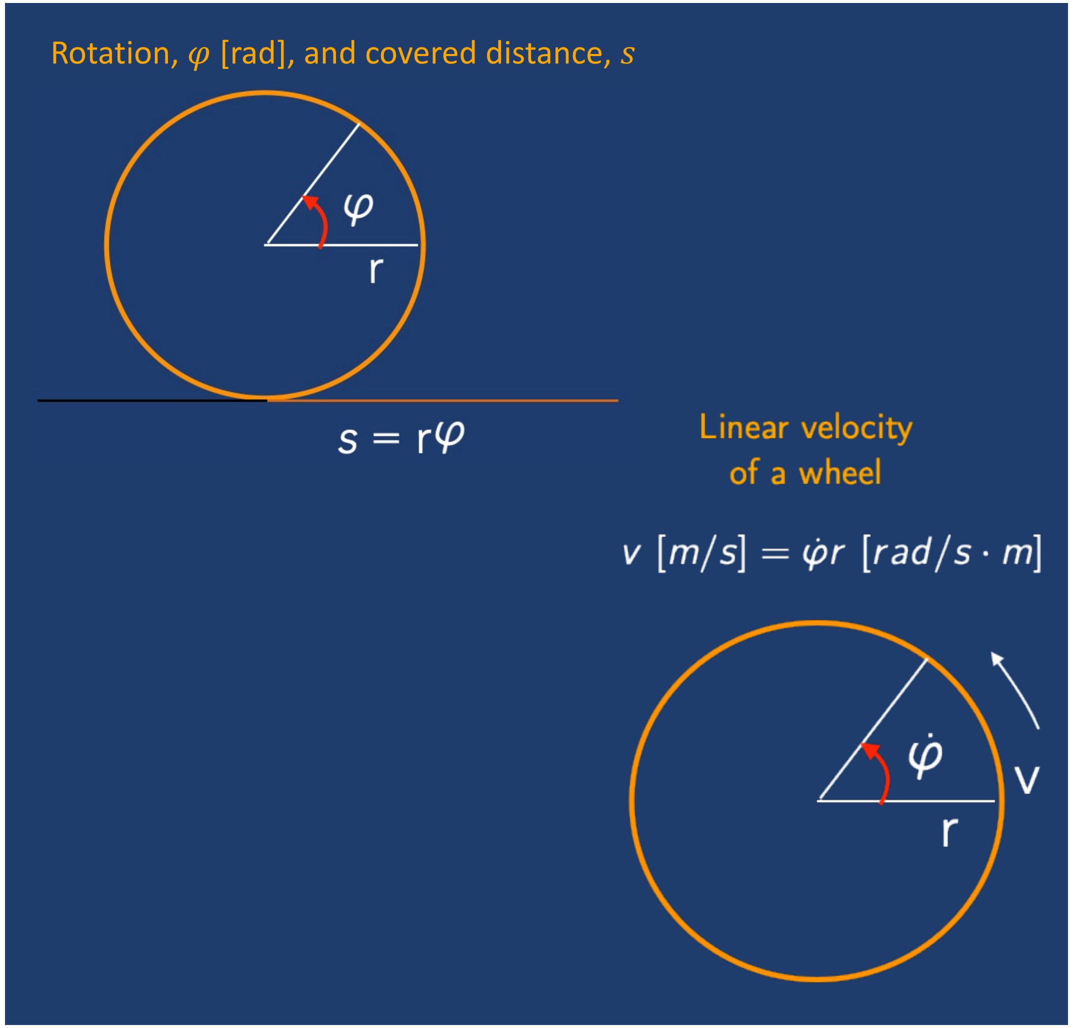
(Computing pose) Mobile robots: Modeling a differential robot





(Computing pose) Mobile robots: Modeling a differential robot





Computing pose of mobile robots: Let's go back to it!

- **Configuration variables**: $q = [\varphi_1\ \varphi_2]$ the rotation angles of the wheels

✓ From $q = [\varphi_1\ \varphi_2]$ we can measure total distance space S traveled by each wheel and therefore by robot i as:

$$S_{iR} = \varphi_{iR} r$$

$$S_{iL} = \varphi_{iL} r$$

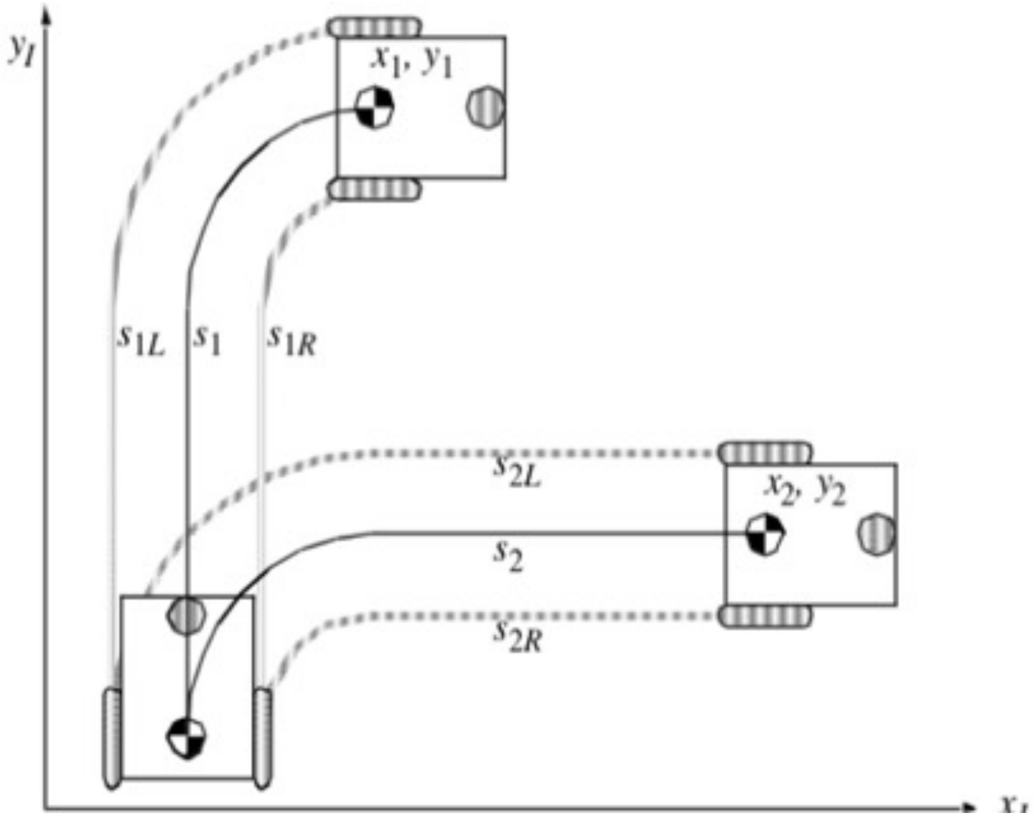
$$S_i = \frac{S_{iR} + S_{iL}}{2}$$

(S of an ideal middle wheel)

$$S_{1R} = S_{2R}$$

$$S_{1L} = S_{2L}$$

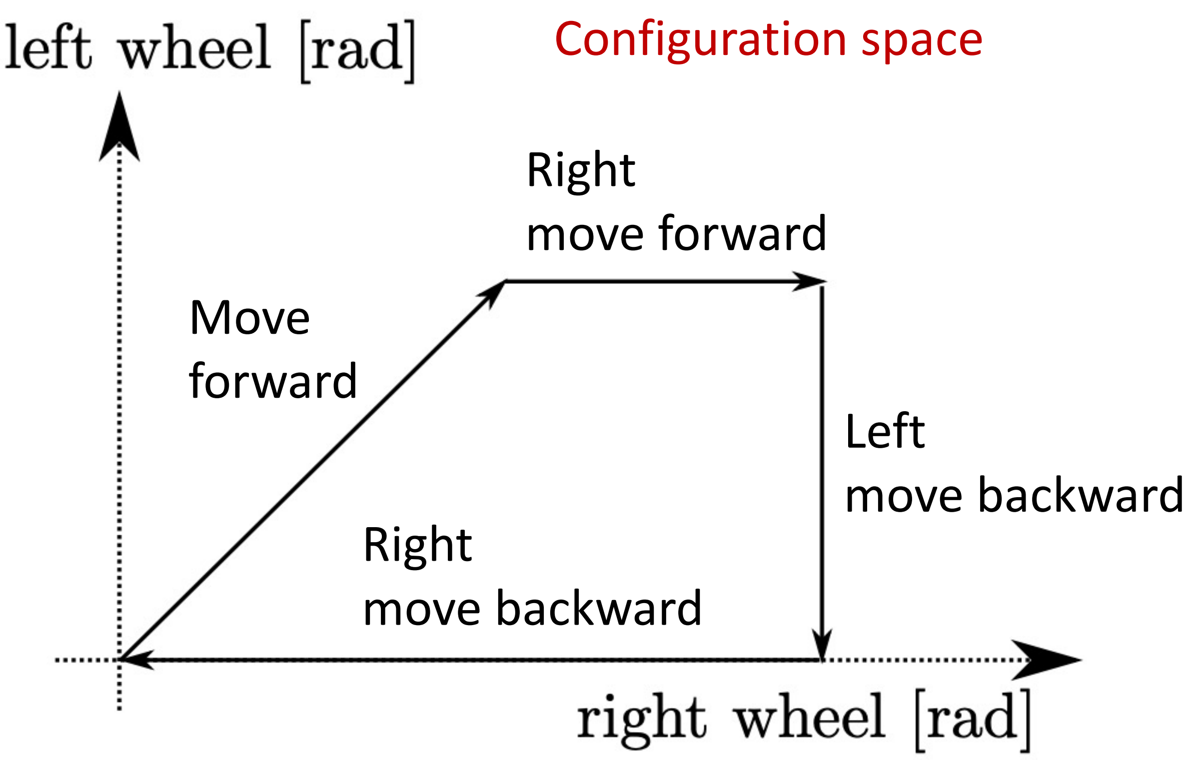
$S_1 = S_2$ but $(x_1, y_1) \neq (x_2, y_2)$

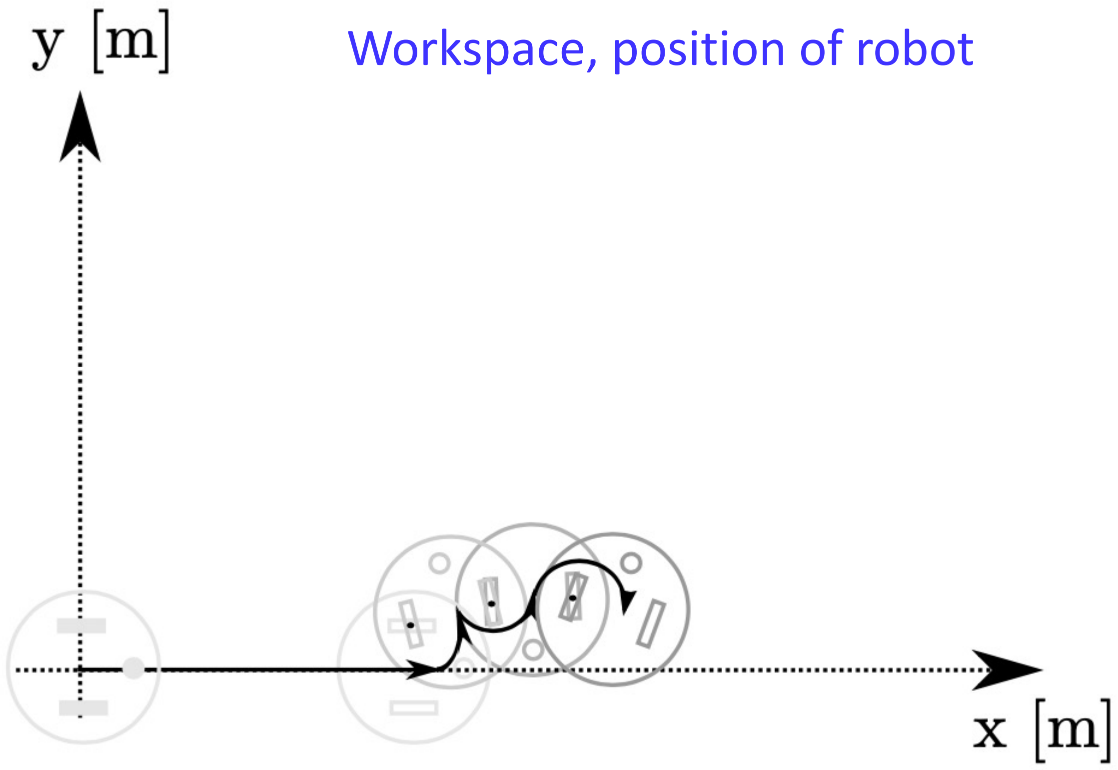


- ○ The measure of the traveled distance S_* of each wheel is not sufficient to calculate the final position of the robot: value of $q(t)$ as **positional variable** is not enough

➢ It is necessary to know how motion was executed as a function of time, **velocity profile** $\dot{q}(t)$
→ **Differential forward kinematics**!

Computing pose of mobile robots: Non-holonomicity





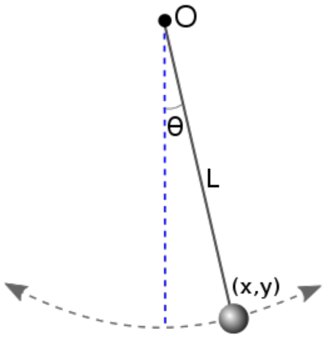
- Closed trajectories in configuration space do not (necessarily) result in closed trajectories in the workspace → Robot’s kinematics is **non-holonomic**

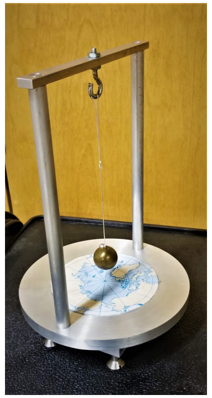
Recap: Holonomic constraints

A **geometric constraint** imposes restrictions on the achievable configurations of the robot. It is based on a functional relation among (some subset of) the configuration variables q:

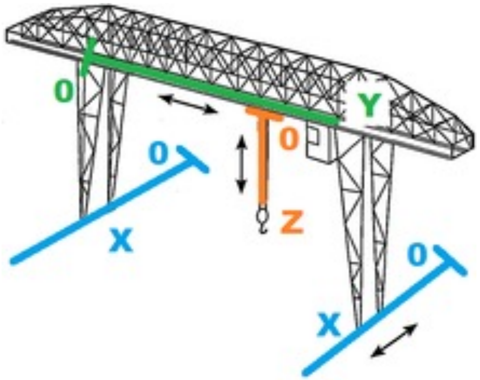
$$f(q) = 0$$

Train





Pendulum, robotic arms



Gantry crane

Geometric constraint → **Holonomic constraint**

Recap: Holonomic constraints

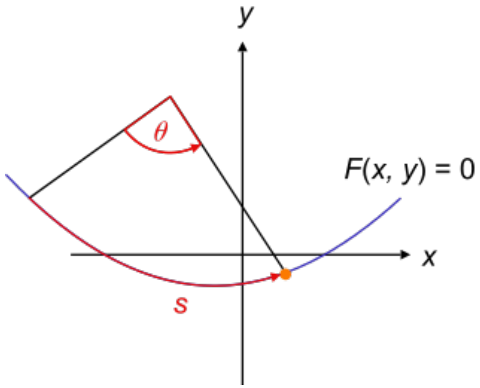
✓ A *geometric / holonomic constraint* is expressed through **positional** variables, e.g., $(\theta_1, \theta_2, \varphi_1, \varphi_2, x, y, z, \theta, \ldots)$, it **only** involves generalized coordinates q, not their derivatives:

$$f(\mathbf{q}) = 0$$

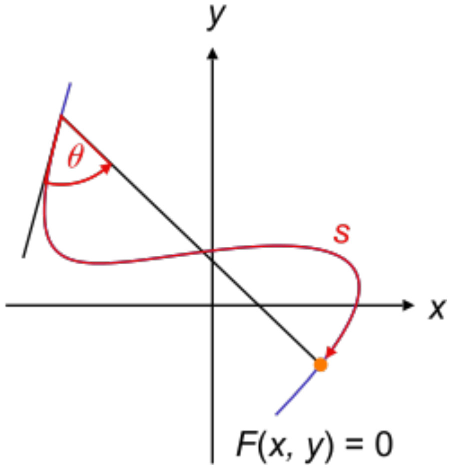
➢ A holonomic constraint limits *the motion of the system to a **manifold** of the configuration space,* depending also on the initial conditions.

$$f(\mathbf{q}, t) = 0$$

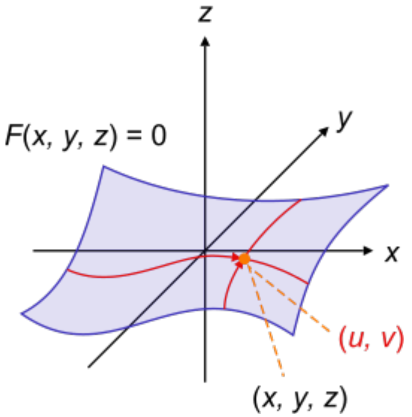
- A constraint *not* depending on time is said scleronomic, rheonomic otherwise. We focus on holonomic constraints that are scleronomic, $f(q) = 0$

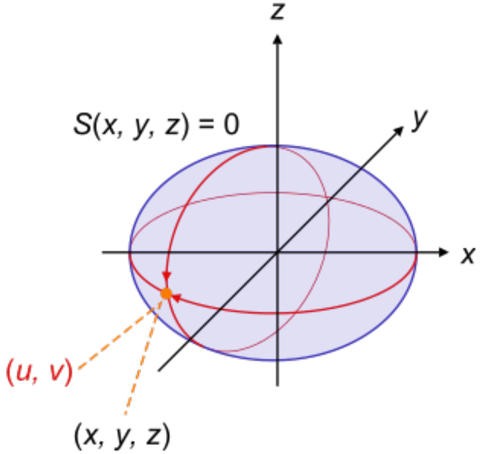


$q = (s)$ or $q = (\theta)$



$q = (s, \theta)$





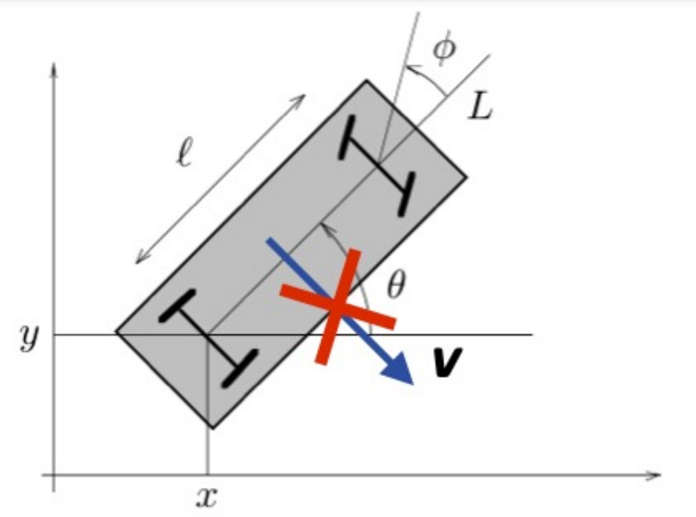
$q = (u, v)$

Need two variables to uniquely identify position in (x, y)

Non-holonomic constraints

A ***kinematic constraint*** imposes restrictions on the achievable velocities of the robot.
It is based on a functional relation among configuration variables q and their derivatives, $\dot{q}$

$$f(q,\ \dot{q},\ t) = 0$$



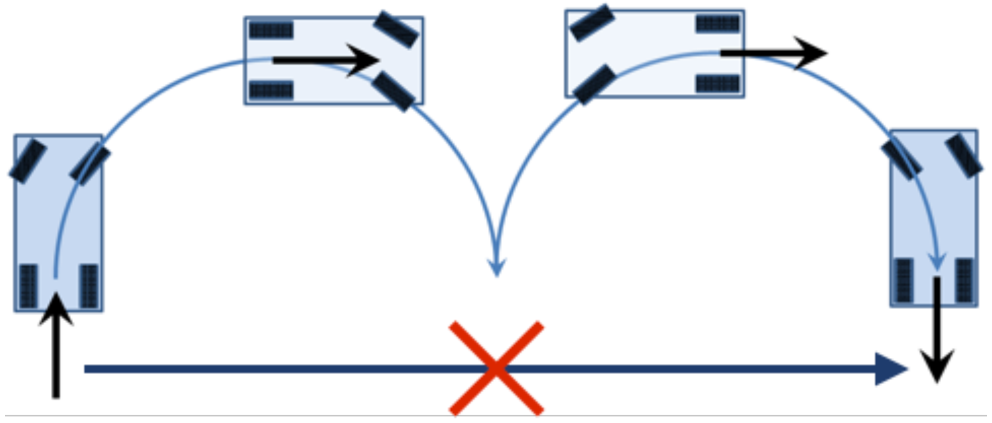
A kinematic constraint is **integrable** if it can be expressed in a form:

$$f(q, t) = 0$$

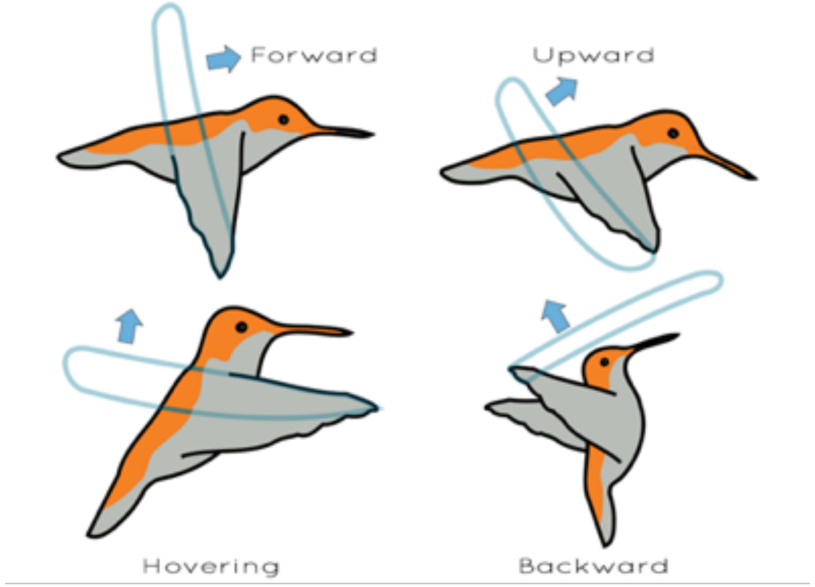
becoming a *holonomic* constraint, being expressed only through positional variables and not their derivatives

A kinematic constraint which is not integrable is a **non holonomic constraint**,
which does not limit the accessible configurations but limits the paths that can be followed to reach them.

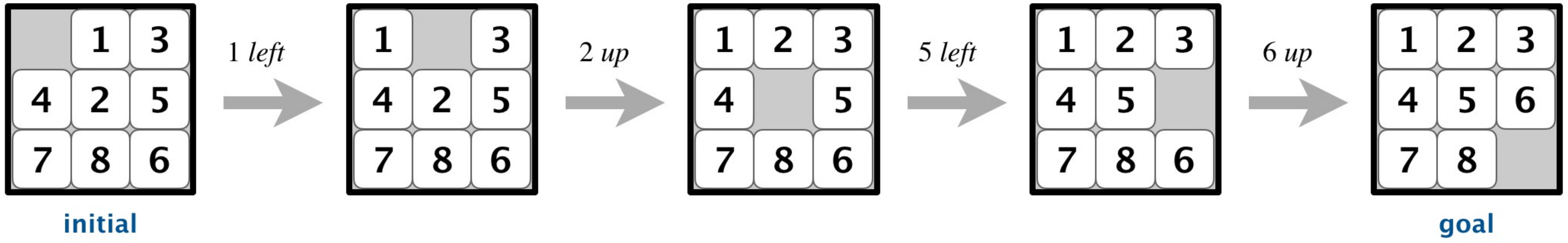
Motion actuators and structure can determine non-holonomic constraints



Two-moves car parking:
no side-way motion

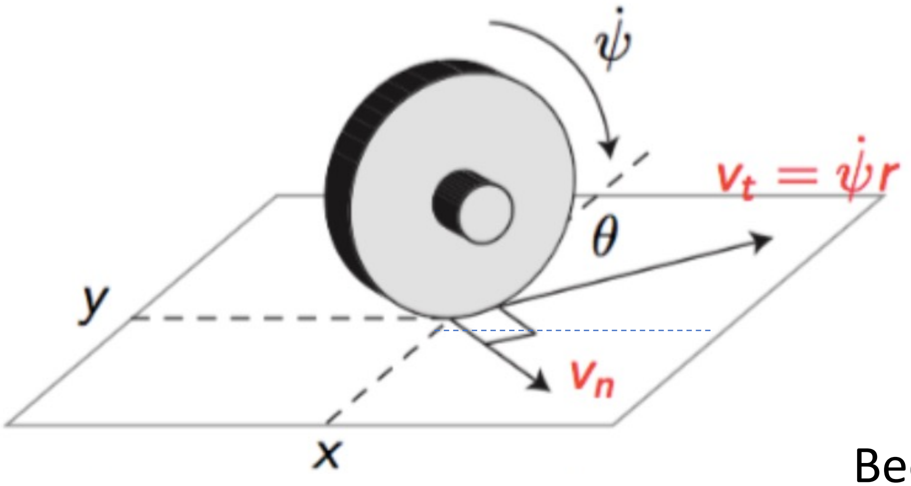


No easy side-way motion in 3D



Motion actuators and structure can determine non-holonomic constraints

- Each standard wheel introduces in the system a non-holonomic constraint since it doesn't allow motion in the direction normal to the rolling direction: the wheel constrains the achievable velocities without typically limiting the achievable configurations



Without constraints, two velocity components, v_t, v_n:

$$\begin{cases} \dot{x} = v_t \cos\theta + v_n \cos(\theta - 90) \\ \dot{y} = v_t \sin\theta + v_n \sin(\theta - 90) \end{cases}$$

Because of the constraint of no slipping in the normal direction, $v_n = 0$

$$\begin{cases} \dot{x} = v_t \cos\theta \\ \dot{y} = v_t \sin\theta \end{cases} \quad \Leftrightarrow \quad \frac{\sin\theta}{\cos\theta} = \tan\theta = \frac{\dot{y}}{\dot{x}} \quad \Leftrightarrow \quad \boxed{\dot{x}\sin\theta - \dot{y}\cos\theta = 0}$$

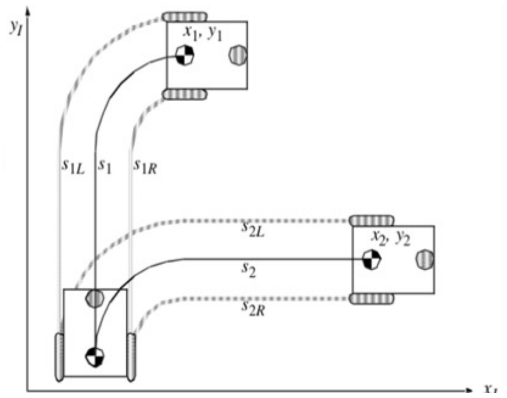
If we explicit w.r.t v_t

Non-holonomic / kinematic constraint

Need for working in the velocity space → Differential kinematics

The presence of non holonomic constraints forces to work in the terms of transformations on velocities rather than on positions

In presence of non holonomic constraints, the differential equations of motion are *not integrable to the final position.*

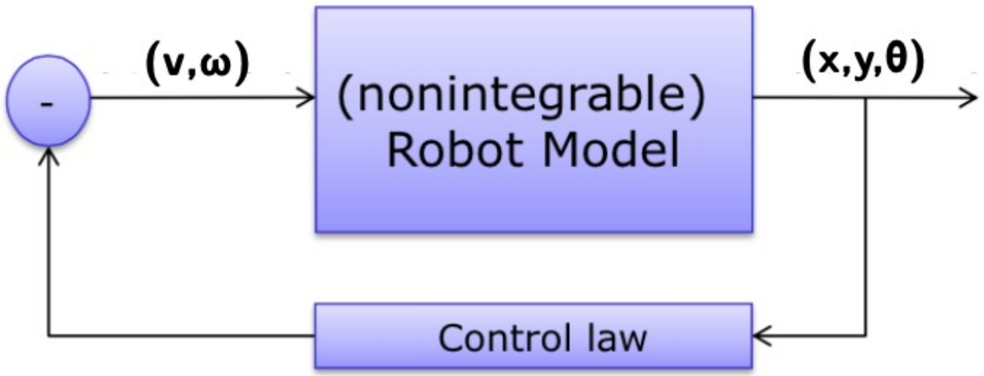


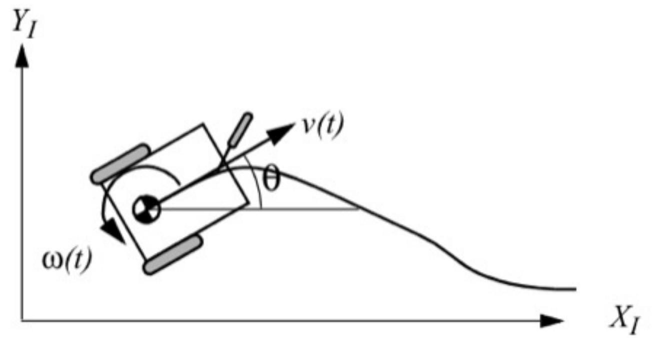
Forward kinematics: Transformation from configuration space to physical space

Inverse kinematics: Transformation from physical space to configuration space

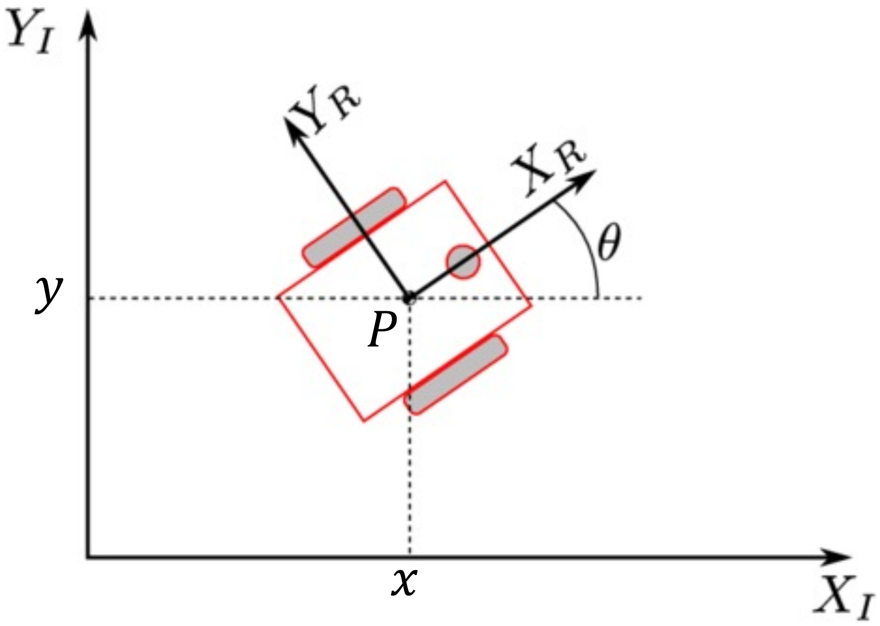
In mobile robotics, due to (pervasive presence of) non holonomic constraints, usually we need to work with ***differential (inverse) kinematics***:

Transformation between velocities instead of positions

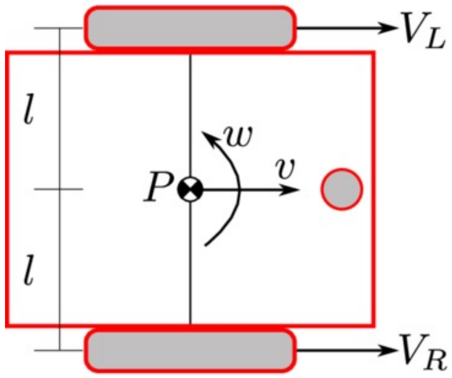




Differential kinematics for differential robot: Poses



- Inertial frame: $\{X_I, Y_I\}$
- Robot frame: $\{X_R, Y_R\}$



Pose in I

$$\xi_I(t) = \begin{bmatrix} x(t) \\ y(t) \\ \theta(t) \end{bmatrix}$$

Pose velocity in I

$$\dot{\xi}_I = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}$$

Pose in R

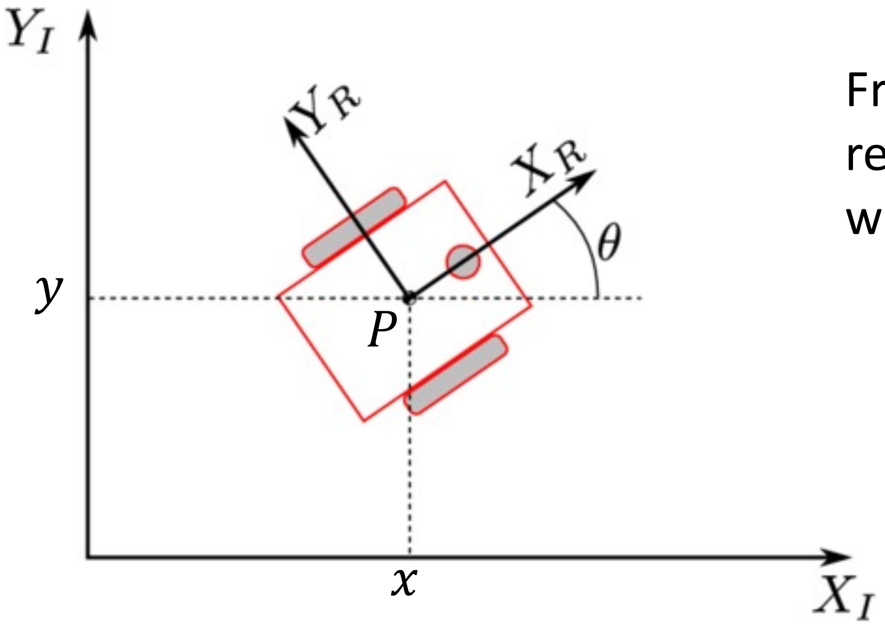
$$\xi_R(t) = \begin{bmatrix} x_R(t) \\ y_R(t) \\ \theta_R(t) \end{bmatrix}$$

Pose velocity in R

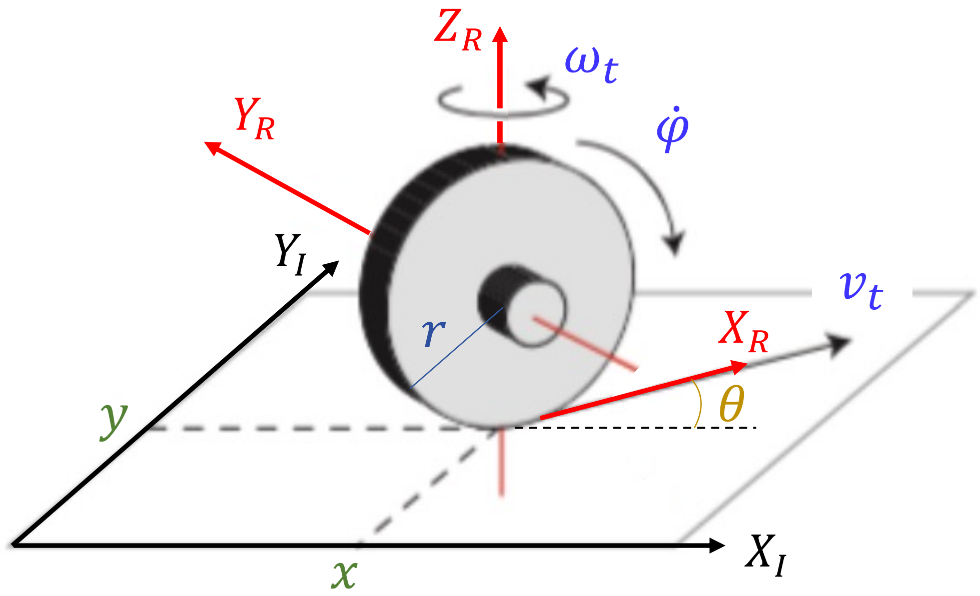
$$\dot{\xi}_R = \begin{bmatrix} \dot{x}_R \\ \dot{y}_R \\ \dot{\theta}_R \end{bmatrix}$$

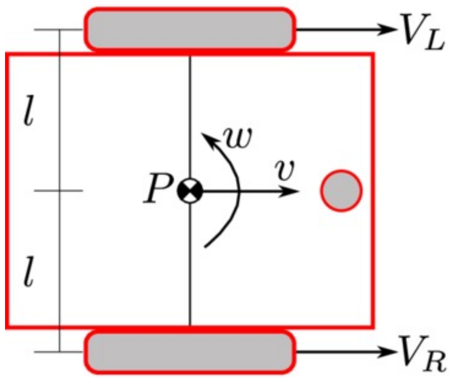
- ξ_I represents the pose of the robot w.r.t. the **inertial, global reference frame** I
- ξ_R is the pose in the **local robot reference frame** R
- The robot is always in $(0,0,0)$ in R, but it has velocity components in X_R (linear) and about Z_R (angular)

Differential kinematics for differential robot: unicycle model



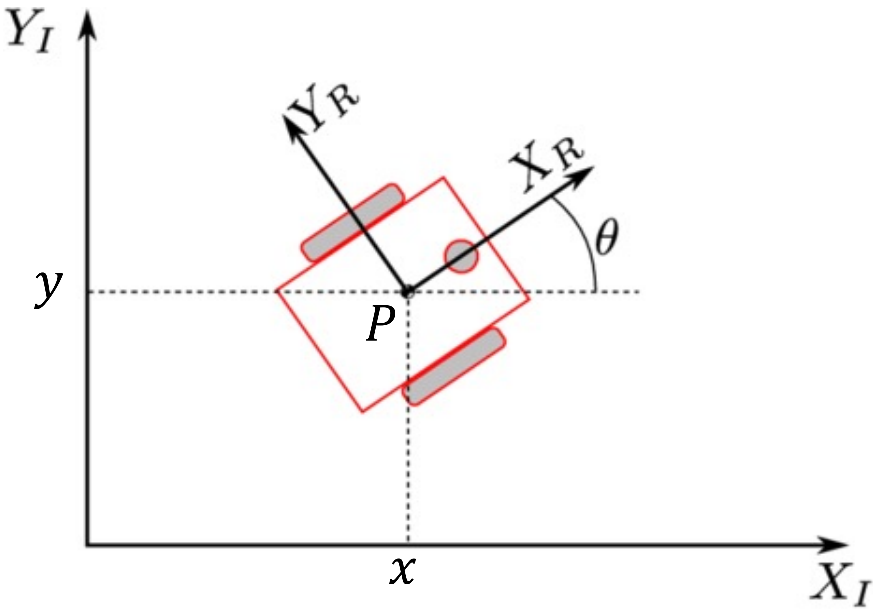
From a purely kinematic perspective, the motion of the robot's reference frame R centered in P is equivalent to a **unicycle** model, which abstracts the (ideal) motion of a **standard steering wheel**





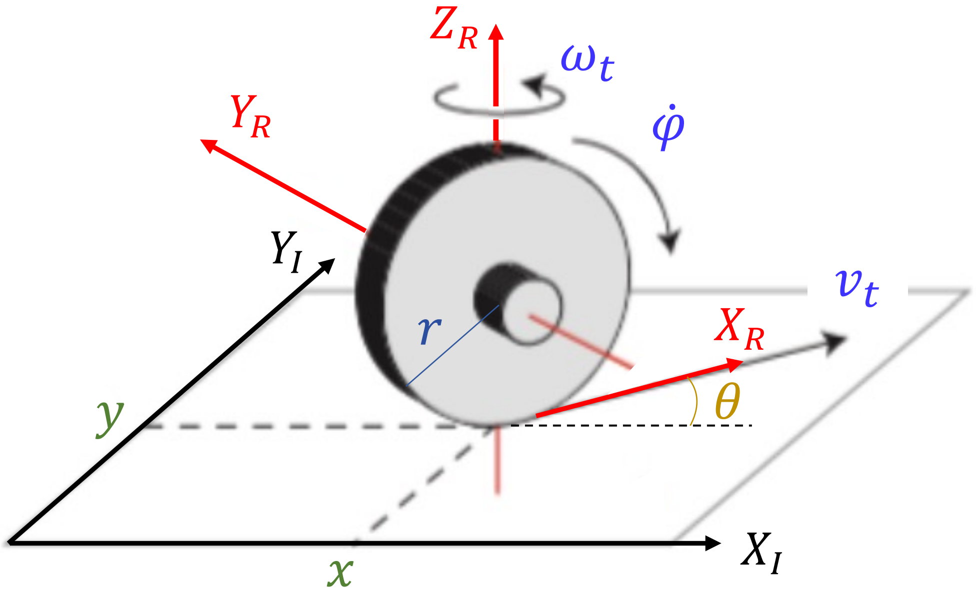
✓ We can **reuse** the equations found before for the one standard wheel model, and add the angular velocity ω

Differential kinematics equations relating velocities



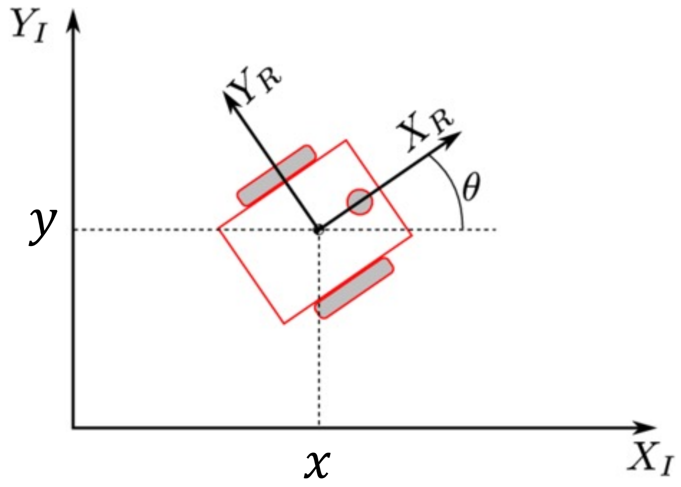
$$\dot{\xi}_I = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v(t)\cos\theta \\ v(t)\sin\theta \\ \omega(t) \end{bmatrix}$$

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



- **Differential kinematic equations** relating robot's velocity pose in I to the twist velocity controls issued to the robot
- $v(t)$ = **linear** / translational velocity
- $\omega(t)$ = **angular** / rotational velocity

Differential kinematics equations using rotation matrix



- $\dot{\xi}_I$ and $\dot{\xi}_R$ represent pose velocities: rate of change of the pose in the respective reference frames

- Robot's frame R is instantaneously roto-translated w.r.t I

✓ Pose transformations apply to both pose and velocity of pose

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

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

Rotation of R w.r.t. I

Instantaneous rotation matrix
(in the dt for pose change / calculations)

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

❖ **Differential kinematic equations** directly relating pose velocity vectors of the two frames, $\dot{\xi}_I$ and $\dot{\xi}_R$

Relating different forms of differential kinematics equations

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

- In R, velocity long the Y_R axis is zero (pure rolling constraint): $\dot{y}_R = 0$
- In R, the linear velocity is $v(t)$, therefore: $\dot{x}_R = v(t)$
- In R, the angular velocity is $\omega(t)$, therefore: $\dot{\theta}_R = \omega(t)$

Substituting:

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

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

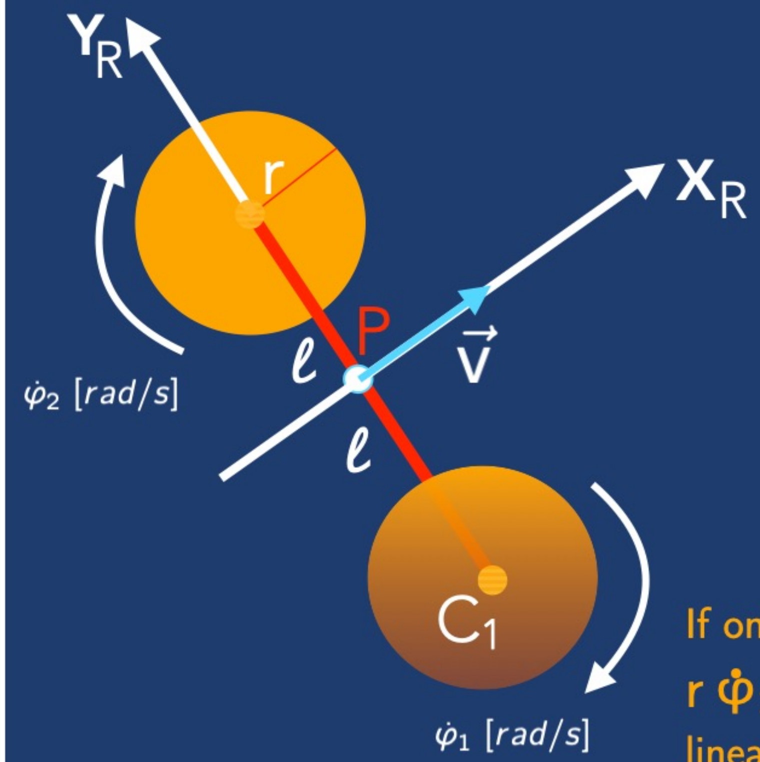
Which is the same as:

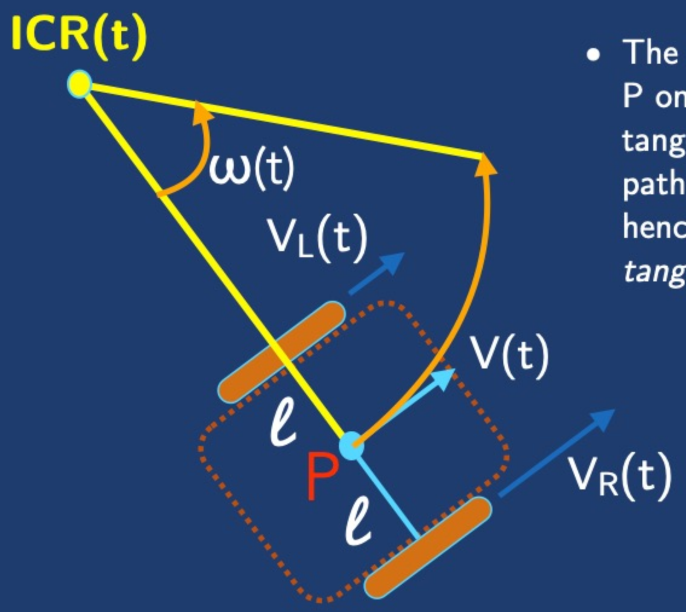
$$\dot{\xi}_I = \begin{bmatrix}\dot{x} \\ \dot{y} \\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}v(t)\cos\theta \\ v(t)\sin\theta \\ \omega(t)\end{bmatrix}$$

Can we express $\dot{\xi}_R$ (and therefore in $\dot{\xi}_I$) terms of **configuration coordinates**?

→ Need to relate $\dot{\varphi}_{1,2}$ to $v(t), \omega(t)$

Differential robot: composition of linear velocities





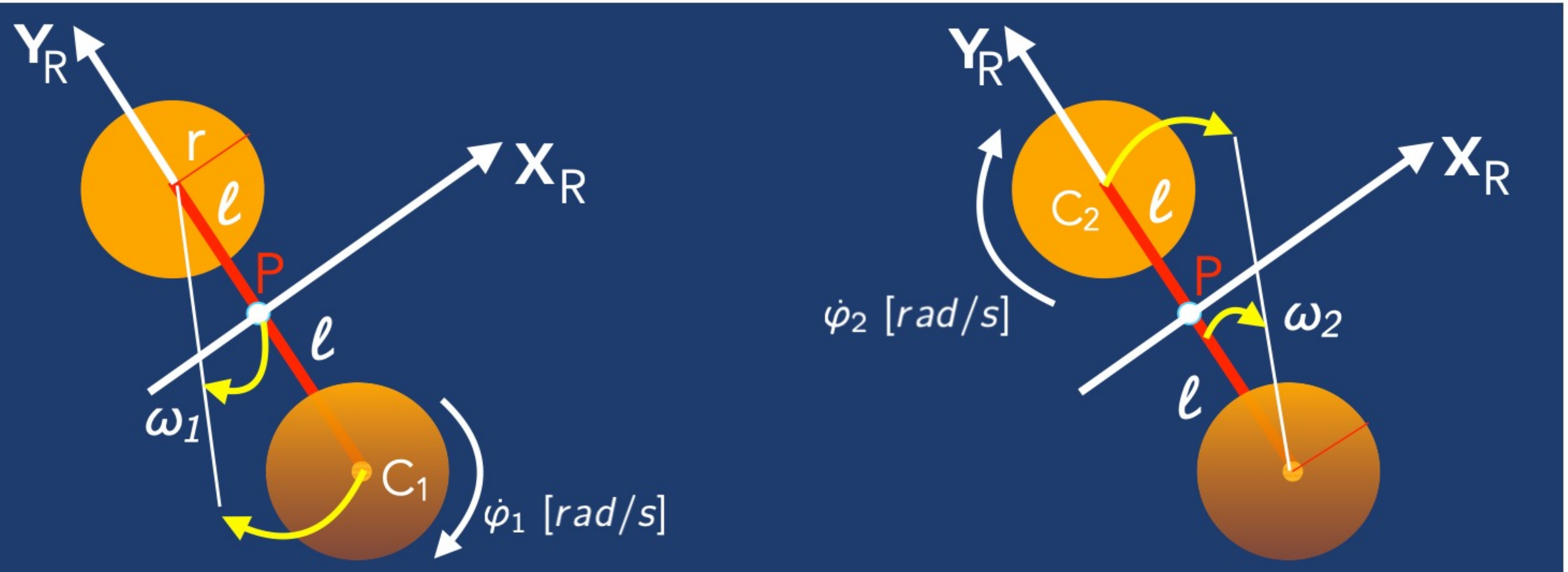
- The linear velocity of a point P on the rigid chassis is tangential to the circular path followed by the point, hence, is also termed a *tangential* velocity

If only the right wheel spins (forward), the linear velocity of C_1 is $r\,\dot{\varphi}_1$, that of C_2 is 0, and that of P is half of that of C_1, since linear velocity scales linearly with the radius (centered in C_2). An analogous reasoning applies when C_2 is the only spinning wheel

The contributions of each wheel to the tangential velocity in P can be computed *independently and added up, each divided by 2*

$$v_P = \frac{r\dot{\varphi}_1 + r\dot{\varphi}_2}{2}$$

Differential robot: composition of angular velocities



If only the right, C_1 wheel spins (forward), the contribution to the angular velocity of P:

$$\omega_1 = \frac{r\dot{\varphi}_1}{2l}$$

If only the left, C_2 wheel spins (forward), the contribution to the angular velocity of P:

$$\omega_2 = -\frac{r\dot{\varphi}_2}{2l}$$

The contributions of each wheel to the angular velocity in P can be *computed independently and added up (signed)*

$$\omega_P = \frac{r\dot{\varphi}_1 - r\dot{\varphi}_2}{2\ell}$$

Differential kinematics equations in configuration variables, I frame

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

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

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

Analogous expressions can be derived for the other form:

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

Summary of differential kinematic equations for the pose velocity $\dot{\xi}_I$ computed in the inertial frame I

Differential kinematics equations in configuration variables, R frame

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

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

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

Summary of differential kinematic equations for the pose velocity $\dot{\xi}_R$ computed in the robot's frame R

Differential forward kinematics and Jacobian

- What about the Jacobian?

Indeed, we HAVE computed the Jacobian, which is the matrix of partial derivatives!

$$\dot{\xi}_R = \begin{bmatrix} \dot{x}_R \\ \dot{y}_R \\ \dot{\theta}_R \end{bmatrix} = \begin{bmatrix} v(t) \\ 0 \\ \omega(t) \end{bmatrix} = J(\varphi_R, \varphi_L) \begin{bmatrix} \dot{\varphi}_R(t) \\ \dot{\varphi}_L(t) \end{bmatrix} = \begin{bmatrix} \frac{\partial x_R}{\partial \varphi_R} & \frac{\partial x_R}{\partial \varphi_L} \\ \frac{\partial y_R}{\partial \varphi_R} & \frac{\partial y_R}{\partial \varphi_L} \\ \frac{\partial \theta_R}{\partial \varphi_R} & \frac{\partial \theta_R}{\partial \varphi_L} \end{bmatrix} \begin{bmatrix} \dot{\varphi}_R(t) \\ \dot{\varphi}_L(t) \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ 0 & 0 \\ \frac{r}{2\ell} & -\frac{r}{2\ell} \end{bmatrix} \begin{bmatrix} \dot{\varphi}_R(t) \\ \dot{\varphi}_L(t) \end{bmatrix}$$

Pose for an (instantaneous) rotation (φ_R, φ_L)

$$x_R = \frac{r\varphi_R + r\varphi_L}{2}$$

$$y_R = 0$$

$$\theta_R = \frac{r\varphi_R - r\varphi_L}{2\ell}$$

$$\rightarrow \quad \begin{bmatrix} \frac{\partial x_R}{\partial \varphi_R} & \frac{\partial x_R}{\partial \varphi_L} \\ \frac{\partial y_R}{\partial \varphi_R} & \frac{\partial y_R}{\partial \varphi_L} \\ \frac{\partial \theta_R}{\partial \varphi_R} & \frac{\partial \theta_R}{\partial \varphi_L} \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ 0 & 0 \\ \frac{r}{2\ell} & -\frac{r}{2\ell} \end{bmatrix}$$

Pose prediction? → Compute integrals of differential kinematic eqs.

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

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

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

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

For a 2-wheeled differential robot

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