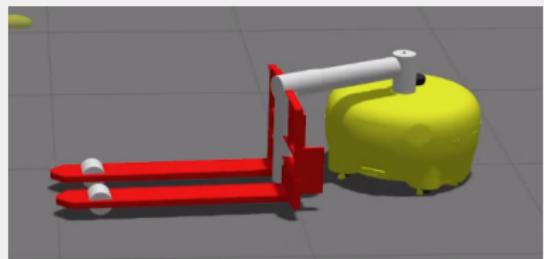


# **AUTONOMOUS MOBILE ROBOTICS**

## MOTION PLANNING AND CONTROL

GEESARA KULATHUNGA

FEBRUARY 6, 2023



# **CONTROL OF MOBILE ROBOTS**

# CONTENTS

- **Kinematics of wheeled mobile robots:** internal, external, direct, and inverse
  - ▶ Differential drive kinematics
  - ▶ Bicycle drive kinematics
  - ▶ Rear-wheel bicycle drive kinematics
  - ▶ Car(Ackermann) drive kinematics
- Wheeled Mobile System Control: **pose** and **orientation**
  - ▶ Control to reference pose
  - ▶ Control to reference pose via an intermediate point
  - ▶ Control to reference pose via an intermediate direction
  - ▶ Control by a straight line and a circular arc
  - ▶ Reference path control
- Dubins path planning

# KINEMATICS OF WHEELED MOBILE ROBOTS

- The process of moving an autonomous system from one place to another is called **Locomotion**



[www.proantic.com/en/display.php](http://www.proantic.com/en/display.php)

# KINEMATICS OF WHEELED MOBILE ROBOTS

- The process of moving an autonomous system from one place to another is called **Locomotion**
- **Kinematic model** describes **geometric relationship** of the system and the system velocities and is presented by **a set of first-order differential equations**



[www.proantic.com/en/display.php](http://www.proantic.com/en/display.php)

# KINEMATICS OF WHEELED MOBILE ROBOTS

- The process of moving an autonomous system from one place to another is called **Locomotion**
- **Kinematic model** describes **geometric relationship** of the system and the system velocities and is presented by **a set of first-order differential equations**
- **Dynamic models** describe a **system motion when forces are applied** to the system and the model is described by **a set of second-order differential equations**



[www.proantic.com/en/display.php](http://www.proantic.com/en/display.php)

# KINEMATICS OF WHEELED MOBILE ROBOTS

- The process of moving an autonomous system from one place to another is called **Locomotion**
- **Kinematic model** describes **geometric relationship** of the system and the system velocities and is presented by **a set of first-order differential equations**
- **Dynamic models** describe a **system motion when forces are applied** to the system and the model is described by **a set of second-order differential equations**
- For mobile robotics **kinematic model is sufficient**



[www.proantic.com/en/display.php](http://www.proantic.com/en/display.php)

# KINEMATICS OF WHEELED MOBILE ROBOTS

- The number of directions in which motion can be made is called DOF

# KINEMATICS OF WHEELED MOBILE ROBOTS

- The number of directions in which motion can be made is called DOF
- A car has 2 DOF

# KINEMATICS OF WHEELED MOBILE ROBOTS

- The number of directions in which motion can be made is called DOF
- A car has 2 DOF
- Highly efficient on hard surfaces compared to Legged locomotion

# KINEMATICS OF WHEELED MOBILE ROBOTS

- The number of directions in which motion can be made is called DOF
- A car has 2 DOF
- Highly efficient on hard surfaces compared to Legged locomotion
- There is no direct way to measure the current pose

# KINEMATICS OF WHEELED MOBILE ROBOTS

- The number of directions in which motion can be made is called DOF
- A car has 2 DOF
- Highly efficient on hard surfaces compared to Legged locomotion
- There is no direct way to measure the current pose
- **Holonomic Systems** - a robot is able to move instantaneously in any direction in the space of its degree of freedom (**Omnidirectional** robot)

# KINEMATICS OF WHEELED MOBILE ROBOTS

- The number of directions in which motion can be made is called DOF
- A car has 2 DOF
- Highly efficient on hard surfaces compared to Legged locomotion
- There is no direct way to measure the current pose
- **Holonomic Systems** - a robot is able to move instantaneously in any direction in the space of its degree of freedom (**Omnidirectional** robot)
- **Non-holonomic Systems** - a robot is not able to move instantaneously in any direction in the space of its degree of freedom

# KINEMATICS OF WHEELED MOBILE ROBOTS

Several types of kinematic models exist

- **Internal kinematics:** consider internal variables (wheel rotation and robot motion)

# KINEMATICS OF WHEELED MOBILE ROBOTS

Several types of kinematic models exist

- **Internal kinematics:** consider internal variables (wheel rotation and robot motion)
- **External kinematics:** robot pose relative to a reference frame

# KINEMATICS OF WHEELED MOBILE ROBOTS

Several types of kinematic models exist

- **Internal kinematics:** consider internal variables (wheel rotation and robot motion)
- **External kinematics:** robot pose relative to a reference frame
- **Direct kinematics:** robot states as a function of its inputs (wheel speed and joints motions)

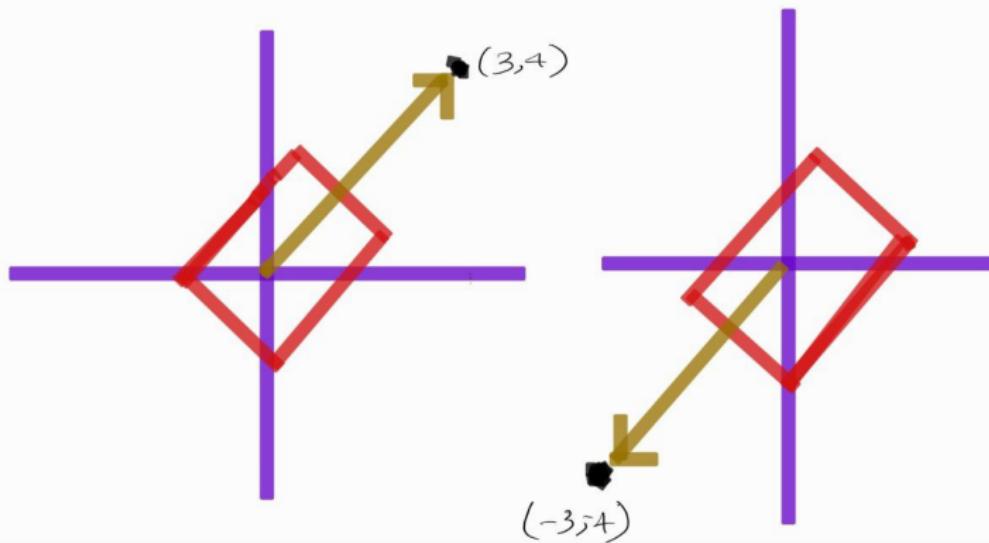
# KINEMATICS OF WHEELED MOBILE ROBOTS

Several types of kinematic models exist

- **Internal kinematics:** consider internal variables (wheel rotation and robot motion)
- **External kinematics:** robot pose relative to a reference frame
- **Direct kinematics:** robot states as a function of its inputs (wheel speed and joints motions)
- **Inverse kinematics:** robot inputs as a function of the desired robot pose

# THE DIFFERENCE BETWEEN ATAN AND ATAN2

Can you estimate the orientation of the robot?



# THE DIFFERENCE BETWEEN ATAN AND ATAN2

Quadrant	Angle	sin	cos	tan
I	$0 < \alpha < \pi/2$	+	+	+
II	$\pi/2 < \alpha < \pi$	+	-	-
III	$\pi < \alpha < 3\pi/2$	-	-	+
IV	$3\pi/2 < \alpha < 2\pi$	-	+	-

- $|A \cdot B| = |A||B|\cos(\theta)$  and  $|A \times B| = |A||B|\sin(\theta)$

# THE DIFFERENCE BETWEEN ATAN AND ATAN2

Quadrant	Angle	sin	cos	tan
I	$0 < \alpha < \pi/2$	+	+	+
II	$\pi/2 < \alpha < \pi$	+	-	-
III	$\pi < \alpha < 3\pi/2$	-	-	+
IV	$3\pi/2 < \alpha < 2\pi$	-	+	-

- $|A \cdot B| = |A||B|\cos(\theta)$  and  $|A \times B| = |A||B|\sin(\theta)$
- $\arctan2(|A \times B|/|A \cdot B|)$

# THE DIFFERENCE BETWEEN ATAN AND ATAN2

Quadrant	Angle	sin	cos	tan
I	$0 < \alpha < \pi/2$	+	+	+
II	$\pi/2 < \alpha < \pi$	+	-	-
III	$\pi < \alpha < 3\pi/2$	-	-	+
IV	$3\pi/2 < \alpha < 2\pi$	-	+	-

- If  $\tan(\alpha)$  is **positive**, it could come from either the **first** or **third** quadrant and if it is **negative**, it could come from either the **second** or **fourth** quadrant. Hence, atan() returns an angle from the first or fourth quadrant (i.e.  $-\pi/2 \leq \text{atan}() \leq \pi/2$ ), regardless of the original input to the tangent

# THE DIFFERENCE BETWEEN ATAN AND ATAN2

Quadrant	Angle	sin	cos	tan
I	$0 < \alpha < \pi/2$	+	+	+
II	$\pi/2 < \alpha < \pi$	+	-	-
III	$\pi < \alpha < 3\pi/2$	-	-	+
IV	$3\pi/2 < \alpha < 2\pi$	-	+	-

- If  $\tan(\alpha)$  is **positive**, it could come from either the **first** or **third** quadrant and if it is **negative**, it could come from either the **second** or **fourth** quadrant. Hence, `atan()` returns an angle from the first or fourth quadrant (i.e.  $-\pi/2 \leq \text{atan}() \leq \pi/2$ ), regardless of the original input to the tangent
- To **get full information**, the values of the **sine and cosine** are **considered separately**. And this is what `atan2()` does. It takes both, the  $\sin(\alpha)$  and  $\cos(\alpha)$  and **resolves all four quadrants** by adding  $\pi$  to the result of `atan()` whenever the **cosine is negative**

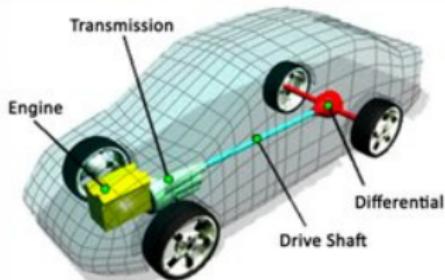
# THE DIFFERENCE BETWEEN ATAN AND ATAN2

Quadrant	Angle	sin	cos	tan
I	$0 < \alpha < \pi/2$	+	+	+
II	$\pi/2 < \alpha < \pi$	+	-	-
III	$\pi < \alpha < 3\pi/2$	-	-	+
IV	$3\pi/2 < \alpha < 2\pi$	-	+	-

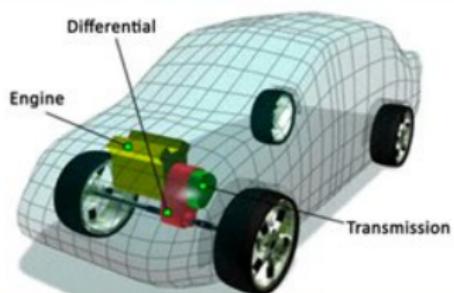
- If  $\tan(\alpha)$  is **positive**, it could come from either the **first** or **third** quadrant and if it is **negative**, it could come from either the **second** or **fourth** quadrant. Hence, `atan()` returns an angle from the first or fourth quadrant (i.e.  $-\pi/2 \leq \text{atan}() \leq \pi/2$ ), regardless of the original input to the tangent
- To **get full information**, the values of the **sine and cosine** are **considered separately**. And this is what `atan2()` does. It takes both, the  $\sin(\alpha)$  and  $\cos(\alpha)$  and **resolves all four quadrants** by adding  $\pi$  to the result of `atan()` whenever the **cosine is negative**
- $\text{atan2}: -\pi < \text{atan2}(y,x) < \pi$  and  $\text{atan}: -\pi/2 < \text{atan}(y/x) < \pi/2$

# DIFFERENTIAL DRIVE KINEMATICS

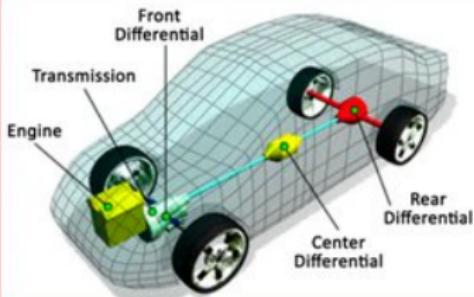
## Rear-Wheel Drive



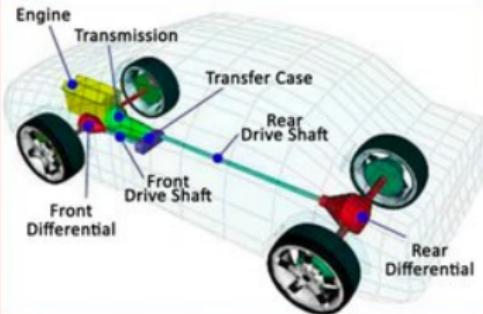
## Front-Wheel Drive



## All-Wheel Drive



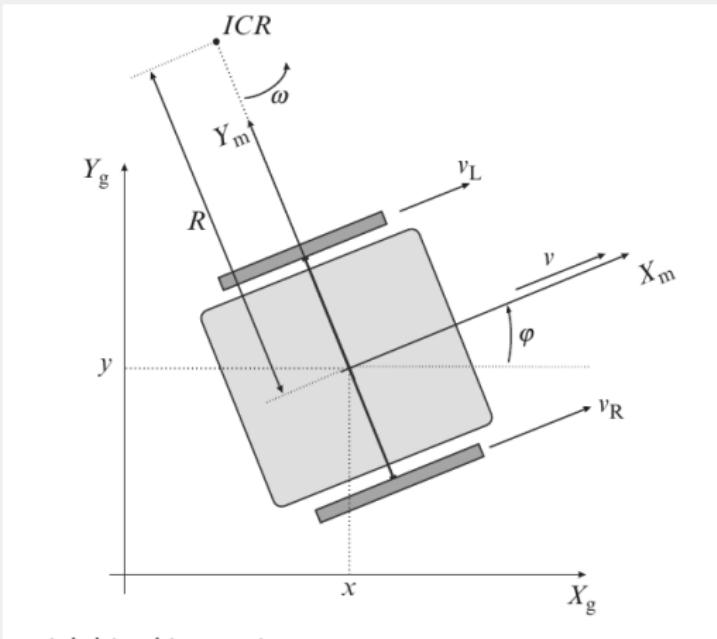
## Four-Wheel Drive



<https://cartreatments.com/types-of-differentials-how-they-work/>

# DIFFERENTIAL DRIVE KINEMATICS

# DIFFERENTIAL DRIVE KINEMATICS



# DIFFERENTIAL DRIVE KINEMATICS

- Well-fit for **smaller mobile robots**

# DIFFERENTIAL DRIVE KINEMATICS

- Well-fit for **smaller mobile robots**
- Usually have one or two castor wheels

# DIFFERENTIAL DRIVE KINEMATICS

- Well-fit for **smaller mobile robots**
- Usually have one or two castor wheels
- **Velocity of each wheel control** separately

# DIFFERENTIAL DRIVE KINEMATICS

- Well-fit for **smaller mobile robots**
- Usually have one or two castor wheels
- **Velocity of each wheel control** separately
- According to Fig. 10,

# DIFFERENTIAL DRIVE KINEMATICS

- Well-fit for **smaller mobile robots**
- Usually have one or two castor wheels
- **Velocity of each wheel control** separately
- According to Fig. 10,
  - ▶ Terms  $v_R(t)$ ,  $v_L(t)$ , denoted velocity of right and left wheels, respectively

# DIFFERENTIAL DRIVE KINEMATICS

- Well-fit for **smaller mobile robots**
- Usually have one or two castor wheels
- **Velocity of each wheel control** separately
- According to Fig. 10,
  - ▶ Terms  $v_R(t)$ ,  $v_L(t)$ , denoted velocity of right and left wheels, respectively
  - ▶ Wheel radius r, distance between wheels L, and term R(t) depicts the vehicle's instantaneous radios (ICR). **Angular velocity** is the **same** for **both left and right wheels around the ICR**.

# DIFFERENTIAL DRIVE KINEMATICS

## ■ Tangential velocity

$$\mathbf{v}(t) = \omega(t)R(t) = \frac{\mathbf{v}_R(t) + \mathbf{v}_L(t)}{2} \quad (1)$$

, where  $\omega = \mathbf{v}_L(t)/(R(t) - L/2) = \mathbf{v}_R(t)/(R(t) + L/2)$ . Hence,  $\omega$  and  $R(t)$  can be determined as follows:

$$\begin{aligned}\omega(t) &= \frac{\mathbf{v}_R(t) - \mathbf{v}_L(t)}{L} \\ R(t) &= \frac{L}{2} \frac{\mathbf{v}_R(t) + \mathbf{v}_L(t)}{\mathbf{v}_R(t) - \mathbf{v}_L(t)}\end{aligned} \quad (2)$$

# DIFFERENTIAL DRIVE KINEMATICS

## ■ Tangential velocity

$$\mathbf{v}(t) = \omega(t)R(t) = \frac{\mathbf{v}_R(t) + \mathbf{v}_L(t)}{2} \quad (1)$$

, where  $\omega = \mathbf{v}_L(t)/(R(t) - L/2) = \mathbf{v}_R(t)/(R(t) + L/2)$ . Hence,  $\omega$  and  $R(t)$  can be determined as follows:

$$\begin{aligned}\omega(t) &= \frac{\mathbf{v}_R(t) - \mathbf{v}_L(t)}{L} \\ R(t) &= \frac{L}{2} \frac{\mathbf{v}_R(t) + \mathbf{v}_L(t)}{\mathbf{v}_R(t) - \mathbf{v}_L(t)}\end{aligned} \quad (2)$$

## ■ Wheels tangential velocities (estimated **relative to the center of the respective wheel**)

$$\mathbf{v}_L = r\omega_L(t), \quad \mathbf{v}_R = r\omega_R(t) \quad (3)$$

# DIFFERENTIAL DRIVE KINEMATICS

## ■ Internal robot kinematics

$$\begin{bmatrix} \dot{x}_m(t) \\ \dot{y}_m(t) \\ \dot{\phi}(t) \end{bmatrix} = \begin{bmatrix} v_{X_m}(t) \\ v_{Y_m} \\ \omega(t) \end{bmatrix} = \begin{bmatrix} r/2 & r/2 \\ 0 & 0 \\ -r/L & r/L \end{bmatrix} \begin{bmatrix} \omega_L(t) \\ \omega_R(t) \end{bmatrix} \quad (4)$$

, where  $\omega(t)$  and  $\mathbf{v}(t)$  are the control variables

# DIFFERENTIAL DRIVE KINEMATICS

## ■ Internal robot kinematics

$$\begin{bmatrix} \dot{x}_m(t) \\ \dot{y}_m(t) \\ \dot{\phi}(t) \end{bmatrix} = \begin{bmatrix} v_{X_m}(t) \\ v_{Y_m} \\ \omega(t) \end{bmatrix} = \begin{bmatrix} r/2 & r/2 \\ 0 & 0 \\ -r/L & r/L \end{bmatrix} \begin{bmatrix} \omega_L(t) \\ \omega_R(t) \end{bmatrix} \quad (4)$$

, where  $\omega(t)$  and  $\mathbf{v}(t)$  are the control variables

## ■ External robot kinematics

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\phi}(t) \end{bmatrix} = \begin{bmatrix} \cos(\phi(t)) & 0 \\ \sin(\phi(t)) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{v}(t) \\ \omega(t) \end{bmatrix} \quad (5)$$

# DIFFERENTIAL DRIVE KINEMATICS

## ■ Internal robot kinematics

$$\begin{bmatrix} \dot{x}_m(t) \\ \dot{y}_m(t) \\ \dot{\Phi}(t) \end{bmatrix} = \begin{bmatrix} v_{X_m}(t) \\ v_{Y_m} \\ \omega(t) \end{bmatrix} = \begin{bmatrix} r/2 & r/2 \\ 0 & 0 \\ -r/L & r/L \end{bmatrix} \begin{bmatrix} \omega_L(t) \\ \omega_R(t) \end{bmatrix} \quad (4)$$

, where  $\omega(t)$  and  $\mathbf{v}(t)$  are the control variables

## ■ External robot kinematics

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\Phi}(t) \end{bmatrix} = \begin{bmatrix} \cos(\Phi(t)) & 0 \\ \sin(\Phi(t)) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{v}(t) \\ \omega(t) \end{bmatrix} \quad (5)$$

## ■ Discrete time dynamics using Euler integration

$$\begin{aligned} x(k+1) &= x(k) + v(k)T_s \cos(\Phi(k)) \\ y(k+1) &= y(k) + v(k)T_s \sin(\Phi(k)) \\ \Phi(k+1) &= \Phi(k) + \omega(k)T_s \end{aligned} \quad (6)$$

, where discrete time instance  $t = kT_s$ ,  $k=0,1,2,\dots$ , for  $T_s$

# DIFFERENTIAL DRIVE KINEMATICS

- Forward robot kinematics (given a set of wheel speeds, determine robot velocity)

$$\begin{aligned}x(k+1) &= x(k) + v(k)T_s \cos(\Phi(k)) \\y(k+1) &= y(k) + v(k)T_s \sin(\Phi(k)) \\ \Phi(k+1) &= \Phi(k) + \omega(k)T_s\end{aligned}\tag{7}$$

, where discrete time instance  $t = kT_s$ ,  $k=0,1,2,\dots$ , for  $T_s$  sampling time

# DIFFERENTIAL DRIVE KINEMATICS

- Forward robot kinematics (given a set of wheel speeds, determine robot velocity)

$$\begin{aligned}x(k+1) &= x(k) + v(k)T_s \cos(\Phi(k)) \\y(k+1) &= y(k) + v(k)T_s \sin(\Phi(k)) \\ \Phi(k+1) &= \Phi(k) + \omega(k)T_s\end{aligned}\tag{7}$$

, where discrete time instance  $t = kT_s$ ,  $k=0,1,2,\dots$ , for  $T_s$  sampling time

- We can also try trapezoidal numerical integration for better approximation

$$\begin{aligned}x(k+1) &= x(k) + v(k)T_s \cos(\Phi(k) + \omega(k)T_s/2) \\y(k+1) &= y(k) + v(k)T_s \sin(\Phi(k) + \omega(k)T_s/2) \\ \Phi(k+1) &= \Phi(k) + \omega(k)T_s\end{aligned}\tag{8}$$

# DIFFERENTIAL DRIVE KINEMATICS

- Inverse robot kinematics (given desired robot velocity, determine corresponding wheel velocities)

# DIFFERENTIAL DRIVE KINEMATICS

- Inverse robot kinematics (given desired robot velocity, determine corresponding wheel velocities)
  - ▶ The **most challenging case compared to direct or forward kinematics**

# DIFFERENTIAL DRIVE KINEMATICS

- Inverse robot kinematics (given desired robot velocity, determine corresponding wheel velocities)
  - ▶ The **most challenging case compared to direct or forward kinematics**
  - ▶ Given the target pose **how many possible ways to get there?**

# DIFFERENTIAL DRIVE KINEMATICS

- Inverse robot kinematics (given desired robot velocity, determine corresponding wheel velocities)
  - ▶ The **most challenging case compared to direct or forward kinematics**
  - ▶ Given the target pose **how many possible ways to get there?**
  - ▶ What if the **robot** goes can perform only **two types of motions: forward and rotations**

$$\begin{aligned}\mathbf{v}_R = \mathbf{v}_L = \mathbf{v}_R, \omega(t) = 0, \mathbf{v}(t) &= \mathbf{v}_R // \text{forward} \\ \mathbf{v}_R = -\mathbf{v}_L = \mathbf{v}_R, \omega(t) &= 2\mathbf{v}_R/L, \mathbf{v}(t) = 0 // \text{rotation}\end{aligned}\tag{9}$$

# DIFFERENTIAL DRIVE KINEMATICS

- Inverse robot kinematics (given desired robot velocity, determine corresponding wheel velocities)

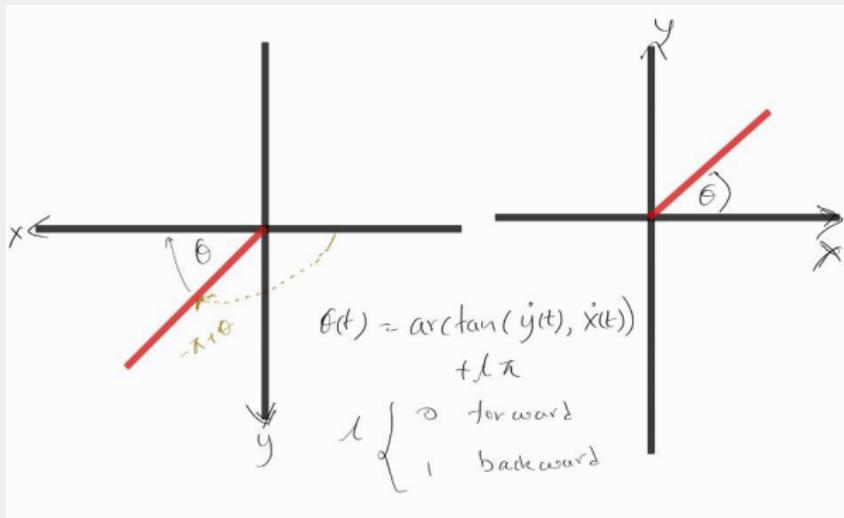
# DIFFERENTIAL DRIVE KINEMATICS

- Inverse robot kinematics (given desired robot velocity, determine corresponding wheel velocities)
  - ▶ If there is a disturbance in the trajectory and know the desired pose at time  $t$ , i.e.,  $x(t), y(t)$

$$\begin{aligned}\mathbf{v}(t) &= \pm \sqrt{\dot{x}^2(t) + \dot{y}^2(t)} // +\text{ forward and - reverse} \\ \Phi(t) &= \arctan 2(\dot{y}(t), \dot{x}(t)) + l\pi, \quad l \in \{0, 1\} \\ &\quad // 0 \text{ forward and } 1 \text{ reverse} \\ \omega(t) &= \frac{\dot{x}(t)\ddot{y}(t) - \dot{y}(t)\ddot{x}(t)}{\dot{x}^2(t) + \dot{y}^2(t)} = v(t)k(t)\end{aligned}\tag{10}$$

, where  $k(t)$  is the **path curvature** and  $\omega(t) = \dot{\Phi}(t)$

# DIFFERENTIAL DRIVE KINEMATICS



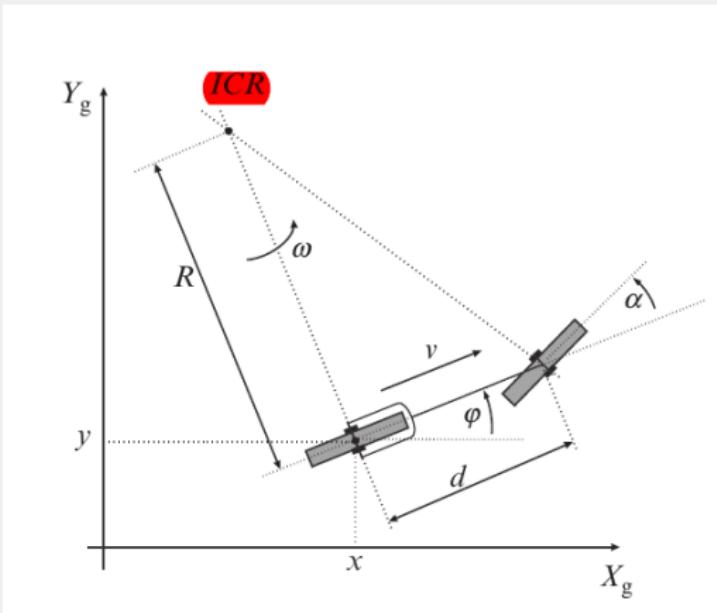
# MOTION CONTROL OF BICYCLE MOBILE ROBOTS



<https://helpfulcolin.com/bike-riding-robots-are-helped-by-gyroscopes-cameras/>

# MOTION CONTROL OF BICYCLE MOBILE ROBOTS

# MOTION CONTROL OF BICYCLE MOBILE ROBOTS



# MOTION CONTROL OF BICYCLE MOBILE ROBOTS

According to Fig. 20,

- Steering angle  $\alpha$ , steering wheel angular velocity  $\omega_S$ , ICR point is defined by intersection of both wheel axes, and distance between wheels  $d$

# MOTION CONTROL OF BICYCLE MOBILE ROBOTS

According to Fig. 20,

- Steering angle  $\alpha$ , steering wheel angular velocity  $\omega_S$ , ICR point is defined by intersection of both wheel axes, and distance between wheels  $d$
- We can define  $R(t)$

$$R(t) = d \tan\left(\frac{\pi}{2} - \alpha(t)\right) = \frac{d}{\tan(\alpha(t))} \quad (11)$$

# MOTION CONTROL OF BICYCLE MOBILE ROBOTS

According to Fig. 20,

- Steering angle  $\alpha$ , steering wheel angular velocity  $\omega_s$ , ICR point is defined by intersection of both wheel axes, and distance between wheels  $d$
- We can define  $R(t)$

$$R(t) = d \tan\left(\frac{\pi}{2} - \alpha(t)\right) = \frac{d}{\tan(\alpha(t))} \quad (11)$$

- Angular velocity  $\omega$  around ICR

$$\omega(t) = \dot{\phi} = \frac{\mathbf{v}_s(t)}{\sqrt{d^2 + R^2}} = \frac{v_s(t)}{d} \sin(\alpha(t)) \quad (12)$$

# MOTION CONTROL OF BICYCLE MOBILE ROBOTS

According to Fig. 20,

- Steering angle  $\alpha$ , steering wheel angular velocity  $\omega_S$ , ICR point is defined by intersection of both wheel axes, and distance between wheels  $d$
- We can define  $R(t)$

$$R(t) = d \tan\left(\frac{\pi}{2} - \alpha(t)\right) = \frac{d}{\tan(\alpha(t))} \quad (11)$$

- Angular velocity  $\omega$  around ICR

$$\omega(t) = \dot{\phi} = \frac{\mathbf{v}_s(t)}{\sqrt{d^2 + R^2}} = \frac{v_s(t)}{d} \sin(\alpha(t)) \quad (12)$$

- Steering wheel velocity

$$\mathbf{v}_s(t) = \omega_S(t)r \quad (13)$$

# BICYCLE MOBILE (FRONT WHEEL DRIVE)

## ■ Internal robot kinematics

$$\begin{aligned}\dot{x}_m(t) &= \mathbf{v}_S(t)\cos(\alpha(t)) \\ \dot{y}_m(t) &= 0\end{aligned}\tag{14}$$

$$\dot{\Phi}(t) = \frac{\mathbf{v}_S(t)}{d} \sin(\alpha(t))$$

# BICYCLE MOBILE (FRONT WHEEL DRIVE)

## ■ Internal robot kinematics

$$\begin{aligned}\dot{x}_m(t) &= \mathbf{v}_S(t)\cos(\alpha(t)) \\ \dot{y}_m(t) &= 0\end{aligned}\tag{14}$$

$$\dot{\Phi}(t) = \frac{\mathbf{v}_S(t)}{d} \sin(\alpha(t))$$

## ■ External robot kinematics

$$\begin{aligned}\dot{x}(t) &= \mathbf{v}_S(t)\cos(\alpha(t))\cos(\Phi(t)) \\ \dot{y}(t) &= \mathbf{v}_S(t)\cos(\alpha(t))\sin(\Phi(t)) \\ \dot{\Phi}(t) &= \frac{\mathbf{v}_S(t)}{d} \sin(\alpha(t))\end{aligned}\tag{15}$$

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\Phi}(t) \end{bmatrix} = \begin{bmatrix} \cos(\Phi(t)) & 0 \\ \sin(\Phi(t)) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{v}(t) \\ \omega(t) \end{bmatrix}\tag{16}$$

, where  $\mathbf{v}(t) = \mathbf{v}_S(t)\cos(\alpha(t))$  and  $\omega(t) = \frac{\mathbf{v}_S}{d} \sin(\alpha(t))$

# MOTION CONTROL OF REAR-WHEEL BICYCLE MOBILE ROBOTS

## ■ Internal robot kinematics

$$\begin{aligned}\dot{x}_m(t) &= \mathbf{v}_s(t)\cos(\alpha(t)) = \mathbf{v}_r(t) \\ \dot{y}_m(t) &= 0 \\ \dot{\phi}(t) &= \frac{\mathbf{v}_r(t)}{d} \tan(\alpha(t))\end{aligned}\tag{17}$$

# MOTION CONTROL OF REAR-WHEEL BICYCLE MOBILE ROBOTS

## ■ Internal robot kinematics

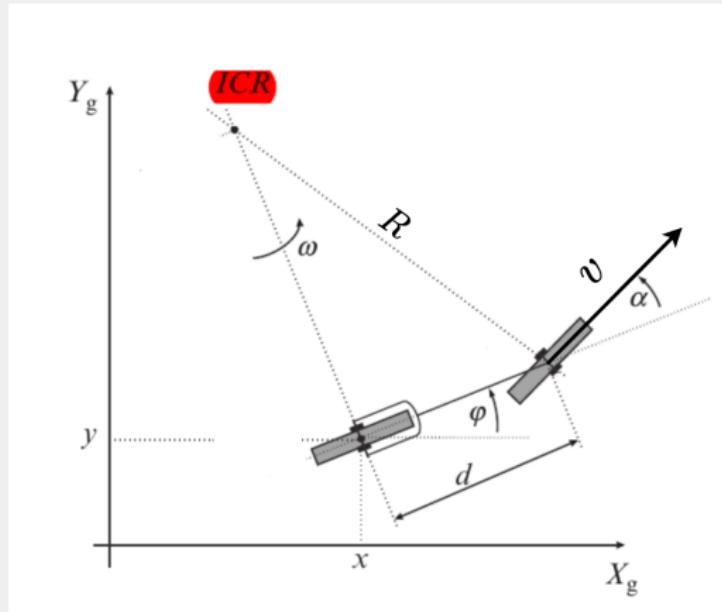
$$\begin{aligned}\dot{x}_m(t) &= \mathbf{v}_s(t)\cos(\alpha(t)) = \mathbf{v}_r(t) \\ \dot{y}_m(t) &= 0 \\ \dot{\Phi}(t) &= \frac{\mathbf{v}_r(t)}{d} \tan(\alpha(t))\end{aligned}\tag{17}$$

## ■ External robot kinematics

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\Phi}(t) \end{bmatrix} = \begin{bmatrix} \cos(\Phi(t)) & 0 \\ \sin(\Phi(t)) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{v}_r(t) \\ \omega(t) \end{bmatrix}\tag{18}$$

, where  $\omega(t) = \frac{\mathbf{v}_r}{d} \tan(\alpha(t))$

# MOTION CONTROL OF BICYCLE MOBILE ROBOTS



# MOTION CONTROL OF BICYCLE MOBILE ROBOTS

## ■ External robot kinematics

$$\begin{aligned}\dot{x}(t) &= v \cdot \cos(\phi(t) + \alpha(t)) \\ \dot{y}(t) &= v \cdot \sin(\phi(t) + \alpha(t)) \\ \dot{\phi}(t) &= v/R = v/(d/\sin(\alpha)) = v \cdot \sin(\alpha)/d \\ \dot{\alpha} &= \text{input (rate of change of steering angle)}\end{aligned}\tag{19}$$

# MOTION CONTROL OF REAR-WHEEL BICYCLE MOBILE ROBOTS

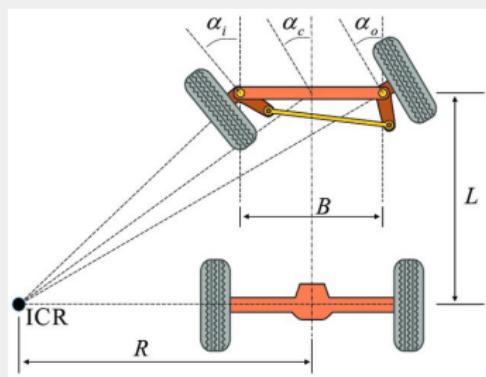


- Bicycle model imposes curvature constraint, where the curvature is defined by

$$k = \frac{\dot{x}(t)\ddot{y}(t) - \dot{y}(t)\ddot{x}(t)}{\left(\dot{x}^2(t) + \dot{y}^2(t)\right)^{3/2}}$$

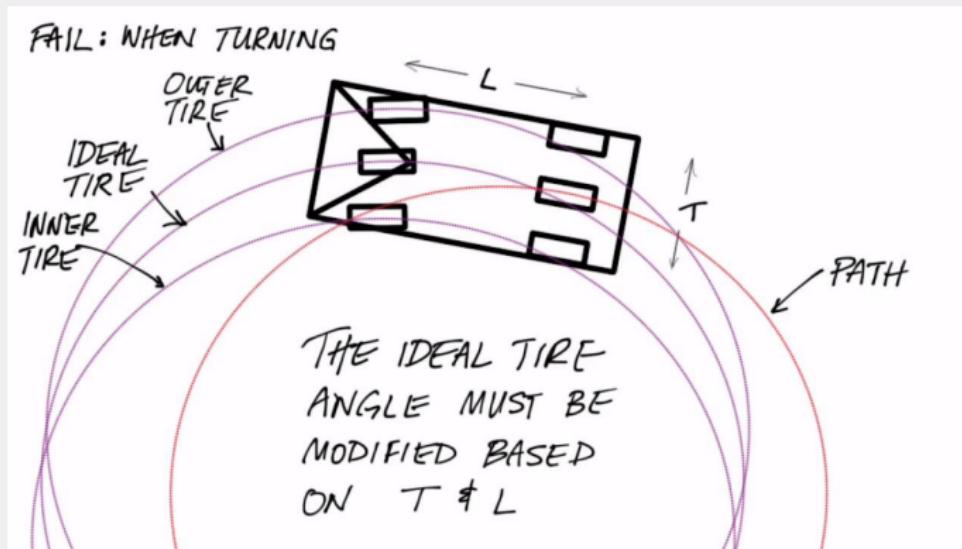
- Curvature constraint is non-holonomic  $v^2 \leq \frac{a_{lat}}{k}$ , where  $a_{lat} \leq a_{lat_{max}}$

# MOTION CONTROL OF CAR(ACKERMANN) DRIVE MOBILE ROBOTS



<https://github.com/winstxnhdw/AutoCarROS2>, <https://doi.org/10.3390/s19214816>

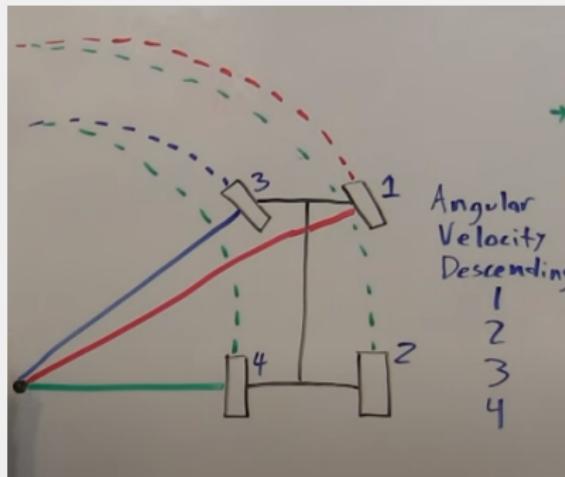
# MOTION CONTROL OF CAR(ACKERMANN) DRIVE MOBILE ROBOTS



<https://www.youtube.com/watch?v=i6uBwudwA5o>

# MOTION CONTROL OF CAR(ACKERMANN) DRIVE MOBILE ROBOTS

- Uses **steering principle**, i.e., the inner wheel, which is closer to its ICR, should steer for a bigger angle than the outer wheel. Consequently, the inner wheel travels at a slower speed than the outer wheel



**Figure:** Angular velocity speed descending order

# MOTION CONTROL OF CAR(ACKERMANN) DRIVE MOBILE ROBOTS

- Ackermann geometry is to **avoid** the need for tires to **slip sideways** when following the path around a curve which requires that the ICR point lies on a straight line defined by the rear wheels' axis

# MOTION CONTROL OF CAR(ACKERMANN) DRIVE MOBILE ROBOTS

- Ackermann geometry is to avoid the need for tires to slip sideways when following the path around a curve which requires that the ICR point lies on a straight line defined by the rear wheels' axis
- Ackermann geometry can be seen as two bicycles welded together side by side

# MOTION CONTROL OF CAR(ACKERMANN) DRIVE MOBILE ROBOTS

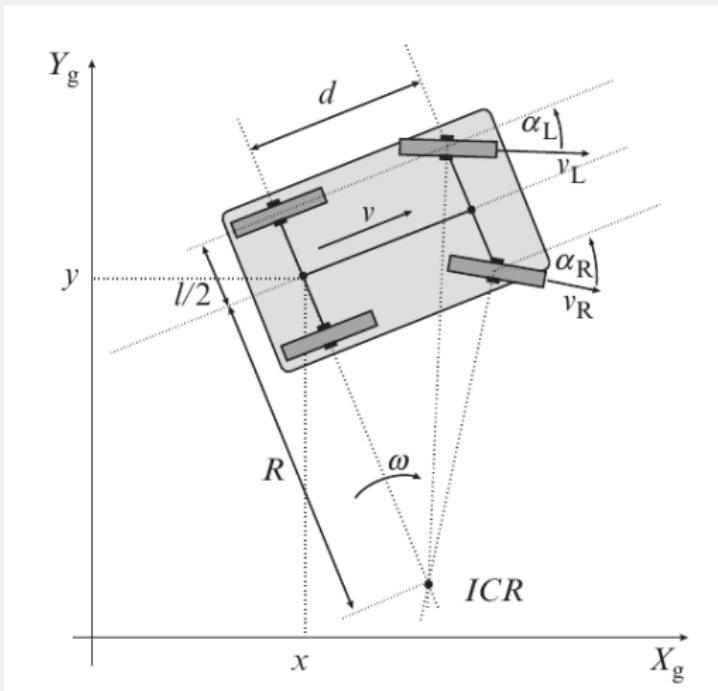
- Ackermann geometry is to avoid the need for tires to slip sideways when following the path around a curve which requires that the ICR point lies on a straight line defined by the rear wheels' axis
- Ackermann geometry can be seen as two bicycles welded together side by side
- For the differential drive it needs individual drives at each wheel which makes the system more complex

# MOTION CONTROL OF CAR(ACKERMANN) DRIVE MOBILE ROBOTS

- Ackermann geometry is to avoid the need for tires to slip sideways when following the path around a curve which requires that the ICR point lies on a straight line defined by the rear wheels' axis
- Ackermann geometry can be seen as two bicycles welded together side by side
- For the differential drive it needs individual drives at each wheel which makes the system more complex
- Ackerman steering adjusts the relative angles of the steerable wheels so they both run around a curve. Differentials allow the two driven wheels to run at different speeds around a curve, which is quite a different requirement

# MOTION CONTROL OF CAR(ACKERMANN) DRIVE MOBILE ROBOTS

# MOTION CONTROL OF CAR(ACKERMANN) DRIVE MOBILE ROBOTS



# MOTION CONTROL OF CAR(ACKERMANN) DRIVE MOBILE ROBOTS

## ■ Steering wheels orientations

$$\begin{aligned}\tan\left(\frac{\pi}{2} - \alpha_L\right) &= \frac{R + l/2}{d} \rightarrow \alpha_L = \frac{\pi}{2} - \arctan\left(\frac{R + l/2}{d}\right) \\ \tan\left(\frac{\pi}{2} - \alpha_R\right) &= \frac{R - l/2}{d} \rightarrow \alpha_R = \frac{\pi}{2} - \arctan\left(\frac{R - l/2}{d}\right)\end{aligned}\quad (20)$$

# MOTION CONTROL OF CAR(ACKERMANN) DRIVE MOBILE ROBOTS

## ■ Steering wheels orientations

$$\begin{aligned}\tan\left(\frac{\pi}{2} - \alpha_L\right) &= \frac{R + l/2}{d} \rightarrow \alpha_L = \frac{\pi}{2} - \arctan\left(\frac{R + l/2}{d}\right) \\ \tan\left(\frac{\pi}{2} - \alpha_R\right) &= \frac{R - l/2}{d} \rightarrow \alpha_R = \frac{\pi}{2} - \arctan\left(\frac{R - l/2}{d}\right)\end{aligned}\quad (20)$$

## ■ Back wheels (inner and outer) velocities

$$\begin{aligned}\mathbf{v}_L &= \omega\left(R + \frac{l}{2}\right) \\ \mathbf{v}_R &= \omega\left(R - \frac{l}{2}\right)\end{aligned}\quad (21)$$

# MOTION CONTROL OF CAR(ACKERMANN) DRIVE MOBILE ROBOTS

## ■ Steering wheels orientations

$$\begin{aligned}\tan\left(\frac{\pi}{2} - \alpha_L\right) &= \frac{R + l/2}{d} \rightarrow \alpha_L = \frac{\pi}{2} - \arctan\left(\frac{R + l/2}{d}\right) \\ \tan\left(\frac{\pi}{2} - \alpha_R\right) &= \frac{R - l/2}{d} \rightarrow \alpha_R = \frac{\pi}{2} - \arctan\left(\frac{R - l/2}{d}\right)\end{aligned}\quad (20)$$

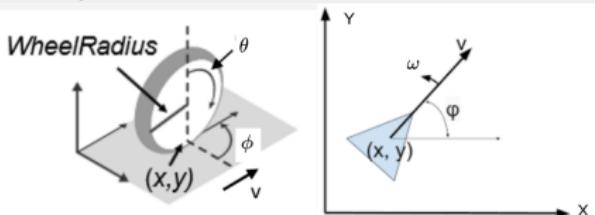
## ■ Back wheels (inner and outer) velocities

$$\begin{aligned}\mathbf{v}_L &= \omega\left(R + \frac{l}{2}\right) \\ \mathbf{v}_R &= \omega\left(R - \frac{l}{2}\right)\end{aligned}\quad (21)$$

## ■ Inverse kinematics is quite complicated (TODO)

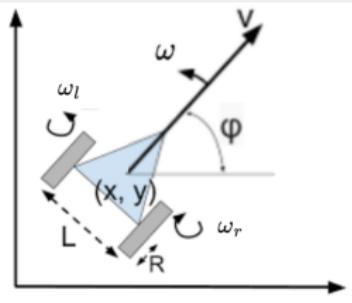
# DEFINE MOBILE ROBOTS WITH KINEMATIC CONSTRAINTS

## Unicycle kinematics



$$\begin{cases} \dot{x} = v\cos(\phi) = r\cos(\phi)\dot{\theta} \\ \dot{y} = v\sin(\phi) = r\sin(\phi)\dot{\theta} \\ \dot{\phi} = \omega \end{cases}$$

## Diffdrive kinematics



$$\begin{cases} \dot{x} = \frac{1}{2}(v_r + v_l)\cos(\phi) \\ \dot{y} = \frac{1}{2}(v_r + v_l)\sin(\phi) \\ \dot{\phi} = \frac{1}{L}(v_r - v_l) \end{cases}$$

After considering these listed models,

$$v_r = \frac{2v + \omega L}{2}, v_l = \frac{2v - \omega L}{2}$$

# DEFINE MOBILE ROBOTS WITH KINEMATIC CONSTRAINTS

- The **unicycle** and **differential drive** models share the generalized control inputs:  $v$  **vehicle speed** and  $\omega$  **vehicle angular velocity**

<https://nl.mathworks.com/help/robotics/ref/ackermannkinematics.html>

# DEFINE MOBILE ROBOTS WITH KINEMATIC CONSTRAINTS

- The **unicycle** and **differential drive** models share the generalized control inputs:  $v$  **vehicle speed** and  $\omega$  **vehicle angular velocity**
- **Unicycle Kinematic Model**  
The **simplest** way to represent **mobile robot vehicle kinematics** is with a unicycle model, which has **a wheel speed set by a rotation about a central axle** and can pivot about its z-axis. Both the **differential-drive** and **bicycle kinematic models reduce** down to **unicycle kinematics** when inputs are provided as vehicle speed and vehicle heading rate and **other constraints are not considered**.

<https://nl.mathworks.com/help/robotics/ref/ackermannkinematics.html>

# DEFINE MOBILE ROBOTS WITH KINEMATIC CONSTRAINTS

## ■ Differential-Drive Kinematic Model

**uses a rear driving axle to control both vehicle speed and heading rate.** The wheels on the driving axle can **spin in both directions.**

<https://nl.mathworks.com/help/robotics/ref/ackermannkinematics.html>

# DEFINE MOBILE ROBOTS WITH KINEMATIC CONSTRAINTS

## ■ Differential-Drive Kinematic Model

uses a rear driving axle to control both vehicle speed and heading rate. The wheels on the driving axle can spin in both directions.

## ■ Bicycle Kinematic Model

treats the robot as a car-like model with two axles: a rear driving axle, and a front axle that turns about the z-axis. The bicycle model assumes that wheels on each axle can be modelled as a single, centred wheel and that the front wheel heading can be directly set, like a bicycle.

<https://nl.mathworks.com/help/robotics/ref/ackermannkinematics.html>

# DEFINE MOBILE ROBOTS WITH KINEMATIC CONSTRAINTS

## ■ Ackermann Kinematic Model

is a modified **car-like model** that assumes Ackermann steering. In most car-like vehicles, the **front wheels do not turn about the same axis**, but instead, **turn on slightly different axes to ensure that they ride on concentric circles about the centre of the vehicle's turn**. This **difference** in turning angle is called **Ackermann steering** and is typically enforced by a mechanism in actual cars. From a vehicle and wheel kinematics standpoint, it can be enforced by treating the steering angle as a rated input.

<https://nl.mathworks.com/help/robotics/ref/ackermannkinematics.html>

# WHEELED MOBILE SYSTEM CONTROL

- Can navigate from a start pose to a goal pose by classical control, where intermediate state trajectory is not prescribed or reference trajectory tracking

# WHEELED MOBILE SYSTEM CONTROL

- Can navigate from a start pose to a goal pose by classical control, where intermediate state trajectory is not prescribed or reference trajectory tracking
- **Nonholonomic constraints** need to be considered, in such occasions, the controller is twofold: **feedforward** control and **feedback** control, namely **two-degree-of-freedom control**.

# WHEELED MOBILE SYSTEM CONTROL

- Can navigate from a start pose to a goal pose by classical control, where intermediate state trajectory is not prescribed or reference trajectory tracking
- **Nonholonomic constraints** need to be considered, in such occasions, the controller is twofold: **feedforward** control and **feedback** control, namely **two-degree-of-freedom control**.
- **Open-loop** control: **feedforward** control is calculated from the reference trajectory and those control actions are fed to the system

# WHEELED MOBILE SYSTEM CONTROL

- Can navigate from a start pose to a goal pose by classical control, where intermediate state trajectory is not prescribed or reference trajectory tracking
- **Nonholonomic constraints** need to be considered, in such occasions, the controller is twofold: **feedforward** control and **feedback** control, namely **two-degree-of-freedom control**.
- **Open-loop** control: **feedforward** control is calculated from the reference trajectory and those control actions are fed to the system
- However, **feedforward** control is **not practical as it is not robust to disturbance**, feedback needs to be applied

# WHEELED MOBILE SYSTEM CONTROL

- Can navigate from a start pose to a goal pose by classical control, where intermediate state trajectory is not prescribed or reference trajectory tracking
- **Nonholonomic constraints** need to be considered, in such occasions, the controller is twofold: **feedforward** control and **feedback** control, namely **two-degree-of-freedom control**.
- **Open-loop** control: **feedforward** control is calculated from the reference trajectory and those control actions are fed to the system
- However, **feedforward** control is **not practical as it is not robust to disturbance**, feedback needs to be applied
- Wheeled mobile robots are dynamic. Thus, the motion controlling system has to incorporate the dynamics of the system, in general, which systems are designed as **cascade control schemes**: **outer controller** for velocity control and **inner controller** to handle torque, force, etc.

# TARGET (REFERENCE) POSE CONTROL

- Pose = position + orientation

## TARGET (REFERENCE) POSE CONTROL

- Pose = position + orientation
- Feasible path, which can be **optimal**, should satisfy the **kinematic, dynamic, and other constraints including disturbances**, appropriately

## TARGET (REFERENCE) POSE CONTROL

- Pose = position + orientation
- Feasible path, which can be **optimal**, should satisfy the **kinematic, dynamic, and other constraints including disturbances**, appropriately
- Reference pose control, in general, is performed as two sub-controlling tasks: **orientation control** and **forward-motion control**. However, **these are interconnected** with each other

# TARGET (REFERENCE) ORIENTATION CONTROL

- Orientation control **cannot be performed** independently from the **forward-motion control**

# TARGET (REFERENCE) ORIENTATION CONTROL

- Orientation control **cannot be performed** independently from the **forward-motion control**
- Let robot orientation  $\Phi(t)$ , at time  $t$ , and reference orientation is  $\Phi_{ref}(t)$

$$e_\Phi(t) = \Phi_{ref}(t) - \Phi(t) \quad (22)$$

# TARGET (REFERENCE) ORIENTATION CONTROL

- Orientation control cannot be performed independently from the forward-motion control
- Let robot orientation  $\Phi(t)$ , at time t, and reference orientation is  $\Phi_{ref}(t)$

$$e_\Phi(t) = \Phi_{ref}(t) - \Phi(t) \quad (22)$$

- How fast can we drive the control error to zero? It depends on additional factors: energy consumption, actuator load, and robustness

# TARGET (REFERENCE) ORIENTATION CONTROL

- Orientation control cannot be performed independently from the forward-motion control
- Let robot orientation  $\Phi(t)$ , at time t, and reference orientation is  $\Phi_{ref}(t)$

$$e_\Phi(t) = \Phi_{ref}(t) - \Phi(t) \quad (22)$$

- How fast can we drive the control error to zero? It depends on additional factors: energy consumption, actuator load, and robustness
- Since  $\dot{\Phi}(t) = \omega(t)$  is the input for control for diff drive, a proportional controller is able to drive control error of an integral process to 0

$$\omega(t) = K(\Phi_{ref} - \Phi(t)) \quad (23)$$

, where K is an arbitrary positive constant

# TARGET (REFERENCE) ORIENTATION CONTROL

- $\dot{\phi}(t) = \frac{v_r}{d} \tan(\alpha(t))$  is the input for control for Ackermann drive. The control variable is  $\alpha$ , which can be chosen proportional to the orientation error:

$$\begin{aligned}\alpha(t) &= K (\Phi_{ref}(t) - \Phi(t)) \\ \dot{\phi}(t) &= \frac{v_r}{d} \tan(K (\Phi_{ref}(t) - \Phi(t)))\end{aligned}\tag{24}$$

# TARGET (REFERENCE) ORIENTATION CONTROL

- $\dot{\phi}(t) = \frac{v_r}{d} \tan(\alpha(t))$  is the input for control for Ackermann drive. The control variable is  $\alpha$ , which can be chosen proportional to the orientation error:

$$\begin{aligned}\alpha(t) &= K (\Phi_{ref}(t) - \Phi(t)) \\ \dot{\phi}(t) &= \frac{v_r}{d} \tan(K (\Phi_{ref}(t) - \Phi(t)))\end{aligned}\tag{24}$$

- **For small angle** and constant velocity of rear wheels  $v_r(t) = V$ , a linear approximation can be obtained,

$$\dot{\phi}(t) = \frac{V}{d} (K (\Phi_{ref}(t) - \Phi(t)))\tag{25}$$

## TARGET (REFERENCE) FORWARD-MOTION CONTROL

- Forward-motion control is inevitably interconnected with orientation control, i.e., **forward-motion alone can not drive to goal pose** without **correct orientation**

$$\mathbf{v(t)} = K \sqrt{((x_{ref}(t) - x(t))^2 + (y_{ref}(t) - y(t))^2)} \quad (26)$$

# TARGET (REFERENCE) FORWARD-MOTION CONTROL

- Forward-motion control is inevitably interconnected with orientation control, i.e., **forward-motion alone can not drive to goal pose** without **correct orientation**

$$\mathbf{v}(t) = K \sqrt{((x_{ref}(t) - x(t))^2 + (y_{ref}(t) - y(t))^2)} \quad (26)$$

- However,  $\mathbf{v}(t)$  has a maximum limit, which is due to actuator limitations driving surface conditions. On the other hand, when the robot gets **closer to the goal**, it might try to **overtake the reference pose**, which **eventually leads to acceleration**, which is not desired

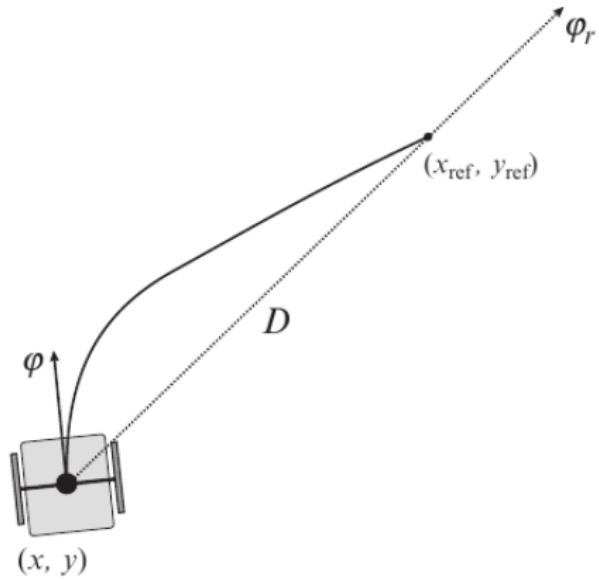
# CONTROL TO REFERENCE POSE

**What is the optimal way to navigate the robot from pose A to pose B?**



# CONTROL TO REFERENCE POSE

# CONTROL TO REFERENCE POSE



## CONTROL TO REFERENCE POSE

- It is required to reach the target position where the final orientation is not prescribed, hence the direction of the reference position

$$\begin{aligned}\Phi_r(t) &= \arctan \frac{y_{ref} - y(t)}{x_{ref} - x(t)}, \quad \omega(t) = K_1(\Phi_r(t) - \Phi(t)) \\ \mathbf{v}(t) &= K \sqrt{((x_{ref}(t) - x(t))^2 + (y_{ref}(t) - y(t))^2)}\end{aligned}\tag{27}$$

## CONTROL TO REFERENCE POSE

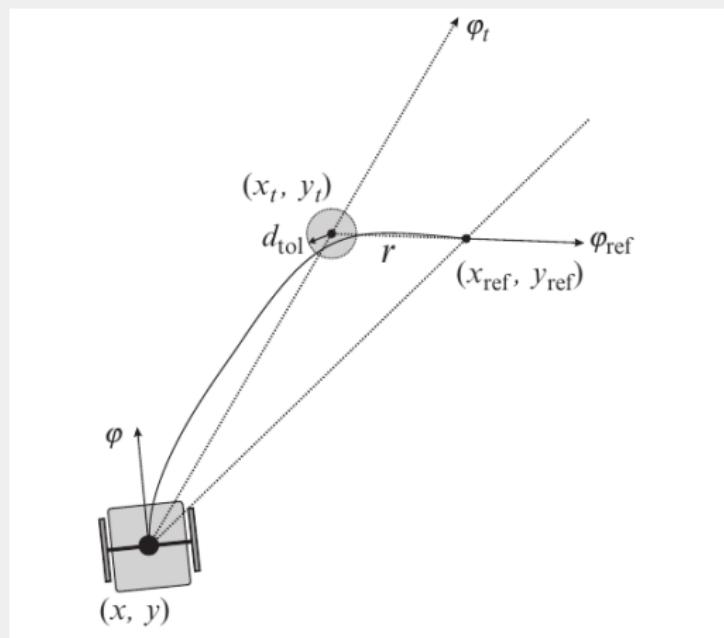
- It is required to reach the target position where the final orientation is not prescribed, hence the direction of the reference position

$$\Phi_r(t) = \arctan \frac{y_{ref} - y(t)}{x_{ref} - x(t)}, \omega(t) = K_1(\Phi_r(t) - \Phi(t))$$
$$\mathbf{v(t)} = K \sqrt{((x_{ref}(t) - x(t))^2 + (y_{ref}(t) - y(t))^2)}$$
(27)

- What will happen when the orientation error abruptly changes ( $\pm 180$  degrees)? if the absolute value of orientation error exceeds 90 degrees, orientation error increases or decreases by 180 degrees

$$e_\Phi(t) = \Phi_{ref}(t) - \Phi(t), \omega(t) = K_1 \arctan(\tan(e_\Phi(t)))$$
$$\mathbf{v(t)} = K \sqrt{((x_{ref}(t) - x(t))^2 + (y_{ref}(t) - y(t))^2)} . sgn(\cos(e_\Phi(t)))$$
(28)

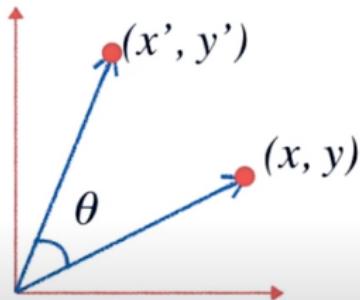
# CONTROL TO REFERENCE POSE VIA AN INTERMEDIATE POINT



# CONTROL TO REFERENCE POSE VIA AN INTERMEDIATE POINT

Rotation by a counterclockwise angle

2D Rotation



$$\begin{aligned}x' &= x \cos(\theta) - y \sin(\theta) \\y' &= x \sin(\theta) + y \cos(\theta)\end{aligned}$$

$$\mathbf{M} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

## CONTROL TO REFERENCE POSE VIA AN INTERMEDIATE POINT

- Idea is to shape in a way that the correct orientation is obtained

# CONTROL TO REFERENCE POSE VIA AN INTERMEDIATE POINT

- Idea is to shape in a way that the correct orientation is obtained
- Intermediate point is determined by

$$\begin{aligned}x_t &= x_{ref} - r \cos(\Phi_{ref}) \\y_t &= y_{ref} - r \sin(\Phi_{ref})\end{aligned}\tag{29}$$

, where the distance from the reference point to the intermediate point denoted  $r$

# CONTROL TO REFERENCE POSE VIA AN INTERMEDIATE POINT

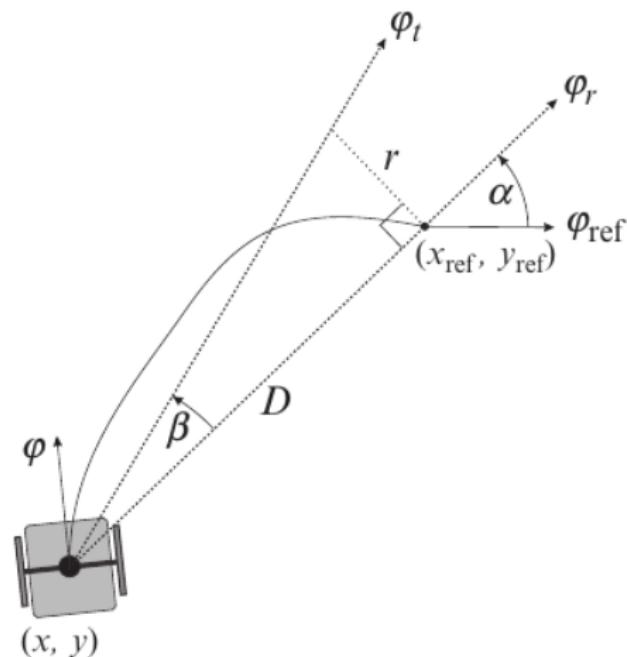
- Idea is to shape in a way that the correct orientation is obtained
- Intermediate point is determined by

$$\begin{aligned}x_t &= x_{ref} - r \cos(\Phi_{ref}) \\y_t &= y_{ref} - r \sin(\Phi_{ref})\end{aligned}\tag{29}$$

, where the distance from the reference point to the intermediate point denoted  $r$

- If distance between current and intermediate position  $\sqrt{(x - x_t)^2 + (y - y_t)^2} < d_{tol}$ , where term  $d_{tol}$  depicts threshold, robot starts controlling to reference point

# CONTROL TO REFERENCE POSE VIA AN INTERMEDIATE DIRECTION



# CONTROL TO REFERENCE POSE VIA AN INTERMEDIATE DIRECTION

- Distance between current pose and target pose

$$D = \sqrt{((x_{ref}(t) - x(t))^2 + (y_{ref}(t) - y(t))^2)} \quad (30)$$

# CONTROL TO REFERENCE POSE VIA AN INTERMEDIATE DIRECTION

- Distance between current pose and target pose

$$D = \sqrt{((x_{ref}(t) - x(t))^2 + (y_{ref}(t) - y(t))^2)} \quad (30)$$

- Let the perpendicular distance to D from the reference point be r. Then,

$$\begin{aligned} \alpha(t) &= \Phi_r(t) - \Phi_{ref} \\ \beta(t) &= \begin{cases} \arctan \frac{+r}{D} & \alpha(t) > 0 \\ -\arctan \frac{r}{D} & \text{otherwise} \end{cases} \end{aligned} \quad (31)$$

, where  $\alpha(t)$  and  $\beta(t)$  are always of the same sign unless  $\alpha = 0$

# CONTROL TO REFERENCE POSE VIA AN INTERMEDIATE DIRECTION

- To define the control law, these facts have to consider:  $\alpha(t)$  reduces when approaching the target, however,  $\beta$  increases. Thus, there are two phases:

$$e_{\Phi}(t) = \Phi_r(t) - \Phi(t) + \begin{cases} \alpha(t) & |\alpha(t)| < |\beta(t)| \\ \beta(t) & \text{otherwise} \end{cases} \quad (32)$$
$$\omega(t) = K e_{\Phi}(t)$$

# CONTROL TO REFERENCE POSE VIA AN INTERMEDIATE DIRECTION

- To define the control law, these facts have to consider:  $\alpha(t)$  reduces when approaching the target, however,  $\beta$  increases. Thus, there are two phases:

$$e_{\Phi}(t) = \Phi_r(t) - \Phi(t) + \begin{cases} \alpha(t) & |\alpha(t)| < |\beta(t)| \\ \beta(t) & \text{otherwise} \end{cases} \quad (32)$$
$$\omega(t) = K e_{\Phi}(t)$$

- In the first phase,  $|\alpha(t)|$  is large, and the robot's orientation is controlled toward the intermediate direction  $\Phi_t(t) = \Phi_r(t) + \beta(t)$ . When  $\alpha$  and  $\beta$  become the same, the current reference orientation switches to  $\Phi_t(t) = \Phi_r(t) + \alpha(t)$

# CONTROL TO REFERENCE POSE VIA AN INTERMEDIATE DIRECTION

- To define the control law, these facts have to consider:  $\alpha(t)$  reduces when approaching the target, however,  $\beta$  increases. Thus, there are two phases:

$$e_{\Phi}(t) = \Phi_r(t) - \Phi(t) + \begin{cases} \alpha(t) & |\alpha(t)| < |\beta(t)| \\ \beta(t) & \text{otherwise} \end{cases} \quad (32)$$
$$\omega(t) = K e_{\Phi}(t)$$

- In the first phase,  $|\alpha(t)|$  is large, and the robot's orientation is controlled toward the intermediate direction  $\Phi_t(t) = \Phi_r(t) + \beta(t)$ . When  $\alpha$  and  $\beta$  become the same, the current reference orientation switches to  $\Phi_t(t) = \Phi_r(t) + \alpha(t)$
- $e_{\Phi}(t)$  is not reducing to zero but is always slightly shifted

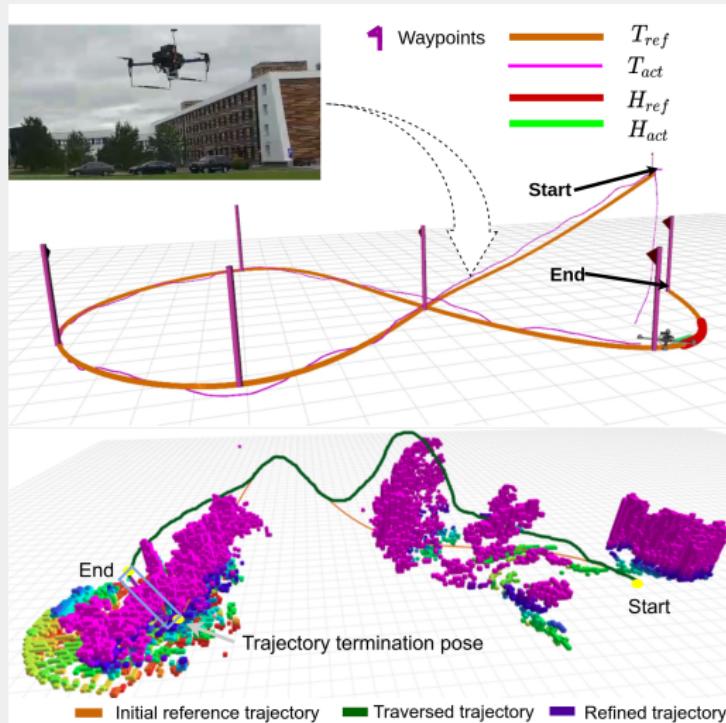
# CONTROL TO REFERENCE POSE VIA AN INTERMEDIATE DIRECTION

- To define the control law, these facts have to consider:  $\alpha(t)$  reduces when approaching the target, however,  $\beta$  increases. Thus, there are two phases:

$$e_{\Phi}(t) = \Phi_r(t) - \Phi(t) + \begin{cases} \alpha(t) & |\alpha(t)| < |\beta(t)| \\ \beta(t) & \text{otherwise} \end{cases} \quad (32)$$
$$\omega(t) = K e_{\Phi}(t)$$

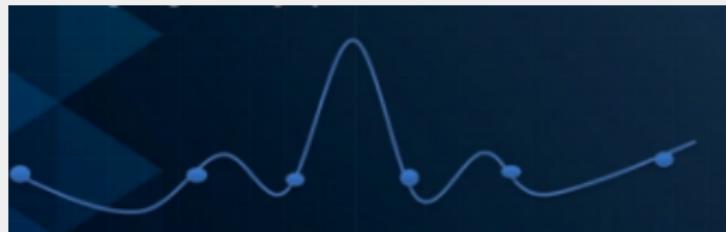
- In the first phase,  $|\alpha(t)|$  is large, and the robot's orientation is controlled toward the intermediate direction  $\Phi_t(t) = \Phi_r(t) + \beta(t)$ . When  $\alpha$  and  $\beta$  become the same, the current reference orientation switches to  $\Phi_t(t) = \Phi_r(t) + \alpha(t)$
- $e_{\Phi}(t)$  is not reducing to zero but is always slightly shifted
- Desired velocity is determined as  $\mathbf{v} = K_p D$ , where  $K_p \in \mathbb{R}^+$  is a constant

# REFERENCE TRAJECTORY GENERATION



<https://youtu.be/g6xHvkcrYcQ?t=21>

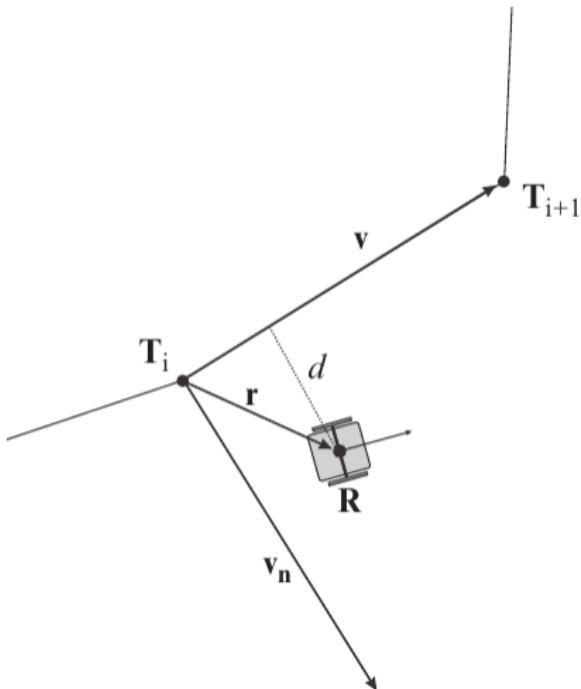
# TYPES OF CURVE FITTING



- B-Spline: can generate control commands without smoothing
- Bezier
- Minimum-span
- **Dubins curve**

# REFERENCE PATH CONTROL

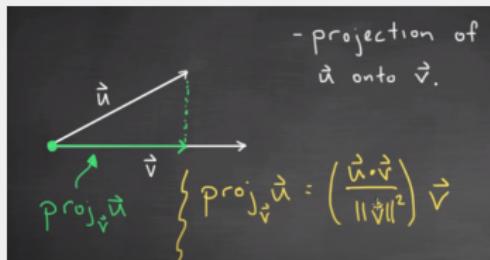
# REFERENCE PATH CONTROL



# REFERENCE PATH CONTROL

The projection has two parts:

- The **direction** of projecting onto. That's the unit vector in direction of  $\mathbf{v}$ , that is  $\frac{\mathbf{v}}{\|\mathbf{v}\|}$
- The **component** of  $\mathbf{u}$  in the direction of  $\mathbf{v}$ :  $\frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{v}\|}$ , because  $\mathbf{u} \cdot \mathbf{v} = \|\mathbf{u}\| \|\mathbf{v}\| \cos(\theta)$  Hence  $\|\mathbf{u}\| \cos(\theta) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{v}\|}$  and that gives the length of  $\mathbf{u}$ 's projection on the direction of  $\mathbf{v}$



<https://www.youtube.com/watch?v=fqPiDICPkj8>

The projection of  $\mathbf{u}$  onto  $\mathbf{v}$  is a vector of length  $\frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{v}\|}$  in the direction of  $\frac{\mathbf{v}}{\|\mathbf{v}\|}$ , i.e.

$$\frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{v}\|} \frac{\mathbf{v}}{\|\mathbf{v}\|}$$

## REFERENCE PATH CONTROL

- Often reference path is given by a set of control points. In such cases, control is driven to drive on a set of straight lines with proper orientation. However, this causes a nonsmooth transition between neighbouring line segments

## REFERENCE PATH CONTROL

- Often reference path is given by a set of control points. In such cases, control is driven to drive on a set of straight lines with proper orientation. However, this causes a nonsmooth transition between neighbouring line segments
- Consider the path given by a set of points  $\mathbf{T}_i = [x_i, y_i]^\top$ , where  $i \in 1, 2, \dots, n$  and  $n$  is the number of points. The orientation between two consecutive line segments is defined by taking the orientation of vector  $\mathbf{T}_{i+1} - \mathbf{T}_i$

## REFERENCE PATH CONTROL

- Often reference path is given by a set of control points. In such cases, control is driven to drive on a set of straight lines with proper orientation. However, this causes a nonsmooth transition between neighbouring line segments
- Consider the path given by a set of points  $\mathbf{T}_i = [x_i, y_i]^\top$ , where  $i \in 1, 2, \dots, n$  and  $n$  is the number of points. The orientation between two consecutive line segments is defined by taking the orientation of vector  $\mathbf{T}_{i+1} - \mathbf{T}_i$
- Let the direction vector be  $\mathbf{v} = [\Delta x, \Delta y]^\top$  along the  $\mathbf{T}_i$ . The vector  $\mathbf{v}_n = [\Delta y, -\Delta x]$  is orthogonal to the vector  $\mathbf{v}$

## REFERENCE PATH CONTROL

- Often reference path is given by a set of control points. In such cases, control is driven to drive on a set of straight lines with proper orientation. However, this causes a nonsmooth transition between neighbouring line segments
- Consider the path given by a set of points  $\mathbf{T}_i = [x_i, y_i]^\top$ , where  $i \in 1, 2, \dots, n$  and  $n$  is the number of points. The orientation between two consecutive line segments is defined by taking the orientation of vector  $\mathbf{T}_{i+1} - \mathbf{T}_i$
- Let the direction vector be  $\mathbf{v} = [\Delta x, \Delta y]^\top$  along the  $\mathbf{T}_i$ . The vector  $\mathbf{v}_n = [\Delta y, -\Delta x]$  is orthogonal to the vector  $\mathbf{v}$
- To check within which line segment robot is located at time  $t$ ,

$$u = \frac{\mathbf{v}^\top \mathbf{r}}{\mathbf{v}^\top \mathbf{v}} \begin{cases} \text{Follow the current segment}(\mathbf{T}_i, \mathbf{T}_{i+1}) & \text{if } 0 < u < 1 \\ \text{Follow the next segment}(\mathbf{T}_i, \mathbf{T}_{i+1}) & \text{if } u > 1 \end{cases} \quad (33)$$

## REFERENCE PATH CONTROL

- The normalized orthogonal distance between the current pose and the line segment that the robot should be

$$d = \frac{\mathbf{v}_n^\top \mathbf{r}}{\mathbf{v}_n^\top \mathbf{v}_n} \quad (34)$$

, where  $d$  is zero if the robot is on the line segment and positive if the robot is on the right side vice versa and  $\mathbf{r} = q - \mathbf{T}_i$ , where  $q$  is the current position of the robot

## REFERENCE PATH CONTROL

- The normalized orthogonal distance between the current pose and the line segment that the robot should be

$$d = \frac{\mathbf{v}_n^\top \mathbf{r}}{\mathbf{v}_n^\top \mathbf{v}_n} \quad (34)$$

, where  $d$  is zero if the robot is on the line segment and positive if the robot is on the right side vice versa and  $\mathbf{r} = q - \mathbf{T}_i$ , where  $q$  is the current position of the robot

- Orientation of line segment that the robot drives

$$\Phi_{lin} = \arctan2(\mathbf{v}_y, \mathbf{v}_x)$$

## REFERENCE PATH CONTROL

- The normalized orthogonal distance between the current pose and the line segment that the robot should be

$$d = \frac{\mathbf{v}_n^\top \mathbf{r}}{\mathbf{v}_n^\top \mathbf{v}_n} \quad (34)$$

, where  $d$  is zero if the robot is on the line segment and positive if the robot is on the right side vice versa and  $\mathbf{r} = \mathbf{q} - \mathbf{T}_i$ , where  $\mathbf{q}$  is the current position of the robot

- Orientation of line segment that the robot drives

$$\Phi_{lin} = \text{arctan2}(\mathbf{v}_y, \mathbf{v}_x)$$

- In case the robot is far from the line segment, it needs to drive perpendicularly to the line segment in order to reach the segment faster

$$\Phi_{rot} = \text{atan}(k_r \cdot d)$$

, where  $k_r \in \mathbb{R}^+$  is a small constant

# REFERENCE PATH CONTROL

- Reference orientation and orientation error

$$\begin{aligned}\Phi_{ref} &= \Phi_{lin} + \Phi_{rot}, \\ e_\Phi &= \Phi_{ref} - \Phi, \quad \omega = K_2 e_\Phi\end{aligned}\tag{35}$$

# REFERENCE PATH CONTROL

- Reference orientation and orientation error

$$\begin{aligned}\Phi_{ref} &= \Phi_{lin} + \Phi_{rot}, \\ e_\Phi &= \Phi_{ref} - \Phi, \quad \omega = K_2 e_\Phi\end{aligned}\tag{35}$$

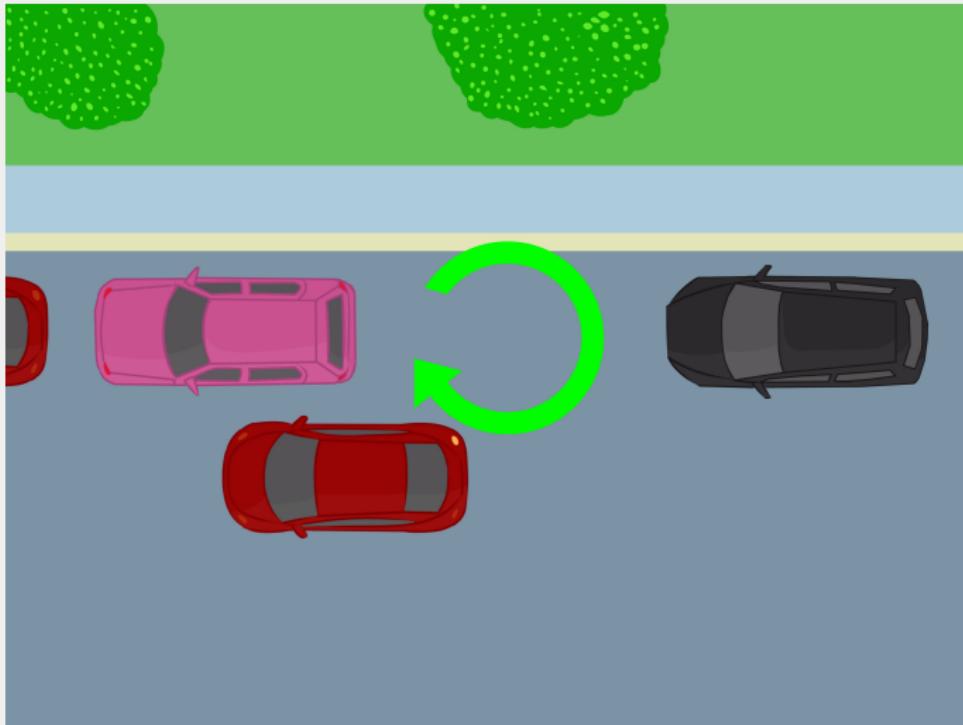
- Then the controller,

$$\begin{aligned}v &= k_p \cdot \cos(e_\Phi) \\ \omega &= k_\Phi \cdot e_\Phi\end{aligned}\tag{36}$$

, where  $k_\Phi, k_p \in \mathbb{R}^+$  are constants

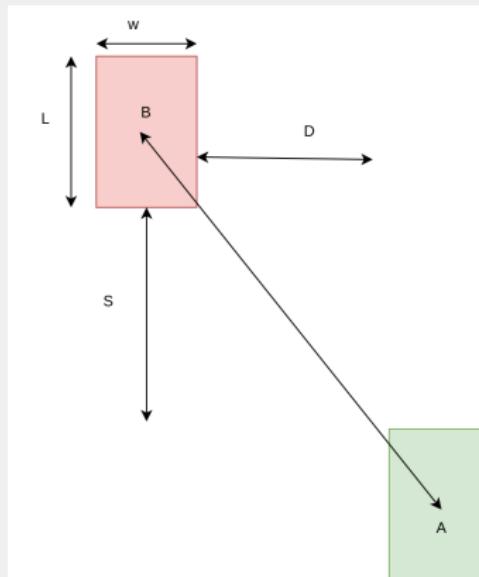
# CONTROL BY CIRCULAR ARCS

## Parallel Parking

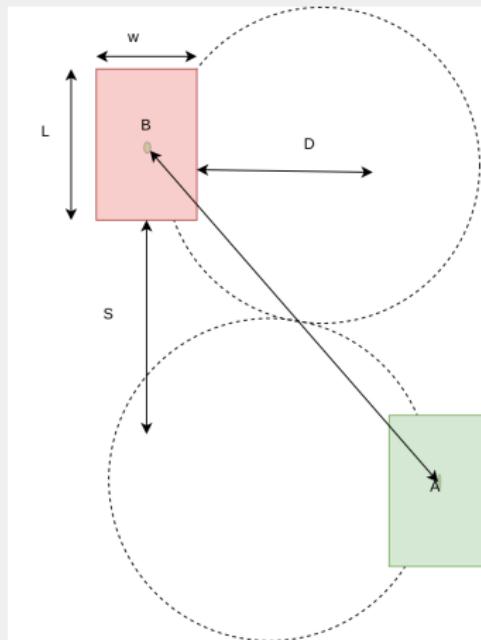


<https://www.wikihow.com/Parallel-Park>

# CONTROL BY CIRCULAR ARCS



# CONTROL BY CIRCULAR ARCS



## CONTROL BY CIRCULAR ARCS

Consider each circle's radius is given by  $r$  if the location of A and B is given by  $\langle x_a, y_a \rangle$  and  $\langle x_b, y_b \rangle$ , respectively

- Top circle:  $(x - (x_b + r))^2 + (y - y_b)^2 = r^2$
- Bottom circle:  $(x - (x_a + r))^2 + (y - y_a)^2 = r^2$

Can you try to define the control law for this scenario?

# LATERAL CONTROL AND LONGITUDINAL CONTROL

## ■ Lateral control

Define the **error relative to the desired reference path** by adjusting the steering angle and **minimize the error** satisfying **input constraints**

- ▶ **Geometric** controls: the **pure pursuit (or pure tracking controller)** and **Stanley**
- ▶ **Dynamic** controls: the model predictive (**MPC**)based-controller, the linear quadratic regulator-(**LQR**)-based

## ■ Longitudinal control

Keep the **vehicle** at the **desired speed** by **controlling** the **accelerator** and **brake**. The controller **minimizes** the **directional error** between the **current vehicle heading angle** and the **desired reference path**.

## LATERAL CONTROL: PURE PURSUIT

The **controller** is defined based on **the center position of the rear/front wheel** to **navigate** the center point to the **desired to track location** by **finding the curvature  $1/R$**  that will result in **finding the steering angle  $\delta$**



The forward-looking distance  $l_d$ , turning radius  $R$ , vehicle heading angle  $\delta$ , and required orientation change  $\alpha$

## LATERAL CONTROL: PURE PURSUIT

- Consider the geometric relationship between  $l_d$ ,  $R$ ,  $\delta$ , and  $\alpha$

$$\frac{l_d}{\sin(2\alpha)} = \frac{R}{\sin(\pi/2 - \alpha)} \Rightarrow R = \frac{l_d}{2\sin(\alpha)}$$

## LATERAL CONTROL: PURE PURSUIT

- Consider the geometric relationship between  $l_d$ ,  $R$ ,  $\delta$ , and  $\alpha$

$$\frac{l_d}{\sin(2\alpha)} = \frac{R}{\sin(\pi/2 - \alpha)} \Rightarrow R = \frac{l_d}{2\sin(\alpha)}$$

- According to the bicycle model

$$\delta = \arctan\left(\frac{L}{R}\right)$$

## LATERAL CONTROL: PURE PURSUIT

- Consider the geometric relationship between  $l_d$ ,  $R$ ,  $\delta$ , and  $\alpha$

$$\frac{l_d}{\sin(2\alpha)} = \frac{R}{\sin(\pi/2 - \alpha)} \Rightarrow R = \frac{l_d}{2\sin(\alpha)}$$

- According to the bicycle model

$$\delta = \arctan\left(\frac{L}{R}\right)$$

- The front wheel rotation angle

$$\delta = \arctan\left(\frac{2L\sin(\alpha)}{l_d}\right), \quad \delta \in [\delta_{min}, \delta_{max}]$$

## LATERAL CONTROL: PURE PURSUIT

- Consider the geometric relationship between  $l_d$ ,  $R$ ,  $\delta$ , and  $\alpha$

$$\frac{l_d}{\sin(2\alpha)} = \frac{R}{\sin(\pi/2 - \alpha)} \Rightarrow R = \frac{l_d}{2\sin(\alpha)}$$

- According to the bicycle model

$$\delta = \arctan\left(\frac{L}{R}\right)$$

- The front wheel rotation angle

$$\delta = \arctan\left(\frac{2L\sin(\alpha)}{l_d}\right), \quad \delta \in [\delta_{min}, \delta_{max}]$$

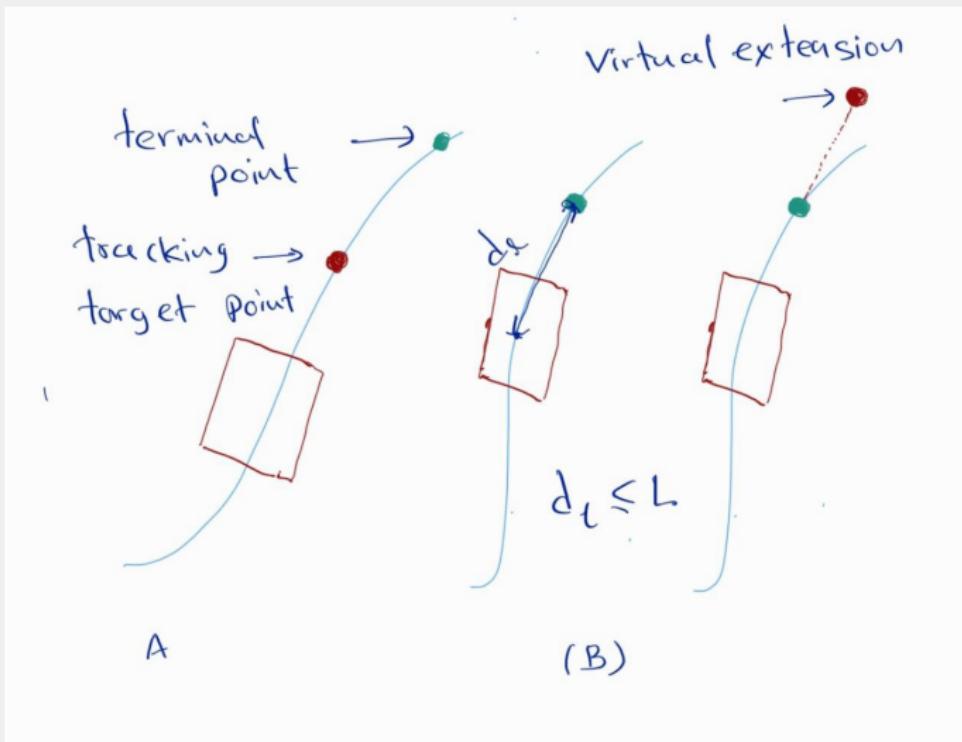
- The **lateral error**  $e_y = l_d \sin(\alpha)$ , however, when the lateral error is small

$$e_y = l_d \sin(\alpha) \approx \frac{l_d^2 \delta}{2L} \Rightarrow \delta = \frac{2L}{l_d^2} e_y$$

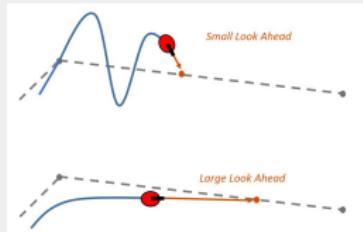
This can be consider as a **P controller**

# LATERAL CONTROL: PURE PURSUIT

When  $l_d \leq L$ , virtual point is considered



## LATERAL CONTROL: LOOK AHEAD DISTANCE



Also, in[1], a non-linear distance estimation approach is expressed in the following form:

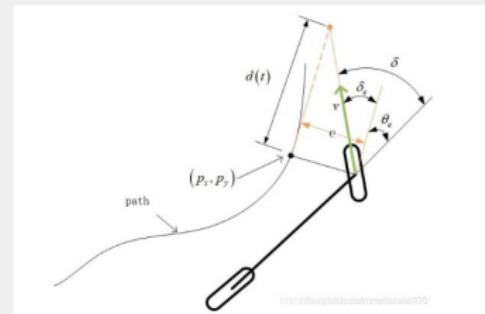
$$l_d = Av^2 + Bv + C$$

Term  $A = \frac{1}{2a_{max}}$ , where  $a_{max}$  is allowed maximum acceleration.  $Av^2$  indicates the shortest braking distance,  $B$  indicates the reaction time required by the vehicle when it encounters an abnormality, and  $Bv$  is the corresponding reaction distance, which  $C$  indicates the minimum turning radius of the vehicle.

[1] J. Duan., et al. Path Tracking based on Pure Pursuit Algorithm for Intelligent Vehicles. IEEE Transactions on Intelligent Transportation Systems, 2018,19(2): 613-626.

# LATERAL CONTROL: STANLEY CONTROL

The **controller** is defined based on **the center position of the front wheel** to **navigate** the center point to the **desired to track location** by **minimizing**: heading error  $\delta_e$  and steer to eliminate cross track error (or lateral tracking error)  $e$



Snider, J. M. (2009). Automatic steering methods for autonomous automobile path tracking. Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RITR-09-08.

## LATERAL CONTROL: STANLEY CONTROL

- Assuming  $d(t) \approx \frac{1}{k}v(t)$ , where k is a gain parameter

$$\delta_e(t) = \arctan\left(\frac{e(t)}{d(t)}\right) = \arctan\left(\frac{ke(t)}{v(t)}\right)$$

- Steering angle

$$\delta(t) = \delta_e(t) + \delta_{\theta_e}(t) \Rightarrow \delta(t) = \delta_{\theta_e}(t) + \arctan\left(\frac{ke(t)}{v(t)}\right), \delta(t) \in [\delta_{min}, \delta_{max}]$$

- For larger positive cross-track error

$$\arctan\left(\frac{ke(t)}{v(t)}\right) \approx \frac{\pi}{2} \Rightarrow \delta(t) = \delta_e(t) + \frac{\pi}{2}$$

# LATERAL CONTROL: STANLEY CONTROL

- The **change of lateral error** by using a bicycle model

$$\dot{e}(t) = -v(t)\sin(\delta_e(t)), \quad \sin(\delta_e(t)) = \frac{e(t)}{\sqrt{d(t)^2 + e(t)^2}}$$

lateral error becomes smaller and smaller, therefore,  
has the negative sign

$$\dot{e}(t) = \frac{-ke(t)}{\sqrt{1 + \left(\frac{ke(t)}{v(t)}\right)^2}}, \quad \left(\frac{ke(t)}{v(t)}\right)^2 \approx 0 \text{ for small } e$$

$$\Rightarrow \dot{e}(t) = -ke(t) \Rightarrow e(t) = e(0) \times \exp^{-kt} \text{ (solution of the first-order linear differential equation/ integration)}$$

## LATERAL CONTROL

- Pure pursuit

Depending on the way of selection of the target point, the **shorter the distance higher the accuracy with some vibrations, higher the distance smoother the control effect with fewer vibrations**

- Stanley control

Depends on the **control gain**. In the case of **unsatisfactory road curvature and smoothness**, the controller is easy to cause excessive vehicle response **overshoot**.

# REFERENCES

-  GREGOR KLANCAR, ANDREJ ZDESAR, SASO BLAZIC, AND IGOR SKRJANC.  
**WHEELED MOBILE ROBOTICS: FROM FUNDAMENTALS TOWARDS AUTONOMOUS SYSTEMS.**  
Butterworth-Heinemann, 2017.
-  ROLAND SIEGWART, ILLAH REZA NOURBAKHSH, AND DAVIDE SCARAMUZZA.  
**INTRODUCTION TO AUTONOMOUS MOBILE ROBOTS.**  
MIT press, 2011.
-  SEBASTIAN THRUN.  
**PROBABILISTIC ROBOTICS.**  
*Communications of the ACM*, 45(3):52–57, 2002.