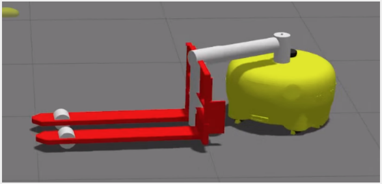
Autonomous Mobile Robotics

Motion Planning and Control

Geesara Kulathunga



January 31, 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

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

- 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

- 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

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

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

- 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

- 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 the direction in the space of its degree of freedom (**Omnidirectional** robot)

- 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 the 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

Several types of kinematic models exist

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

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

Several types of kinematic models exist

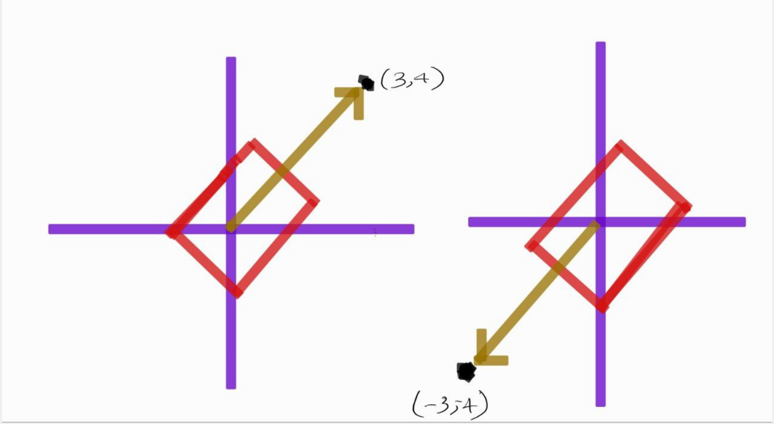
- Internal kinematics: consider internal variables (wheel rotation and robot motion)
- External kinematics: robot pose relative to a reference frame
- Direct kinematics: robot states as a function of its inputs (wheel speed and joints motions)

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 < α < π/2	+	+	+
II	π/2 < α < π	+	-	-
III	π < α < 3π/2	-	-	+
IV	3π/2 < α < 2π	-	+	-

- $|A \cdot B| = |A||B|COS(\theta)$ and $|A \times B| = |A||B|SIN(\theta)$

The difference between atan and atan2

Quadrant	Angle	sin	cos	tan
I	0 < α < π/2	+	+	+
II	π/2 < α < π	+	-	-
III	π < α < 3π/2	-	-	+
IV	3π/2 < α < 2π	-	+	-

- $|A \cdot B| = |A||B|COS(\theta)$ 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    < α < π/2      +     +     +
II          π/2  < α < π        +     -     -
III         π    < α < 3π/2     -     -     +
IV          3π/2 < α < 2π       -     +     -
```

- 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. $-\pi/2 <= atan() <= \pi/2$), regardless of the original input to the tangent

Quadrant	Angle	sin	cos	tan
I	0 < α < π/2	+	+	+
II	π/2 < α < π	+	-	-
III	π < α < 3π/2	-	-	+
IV	3π/2 < α < 2π	-	+	-

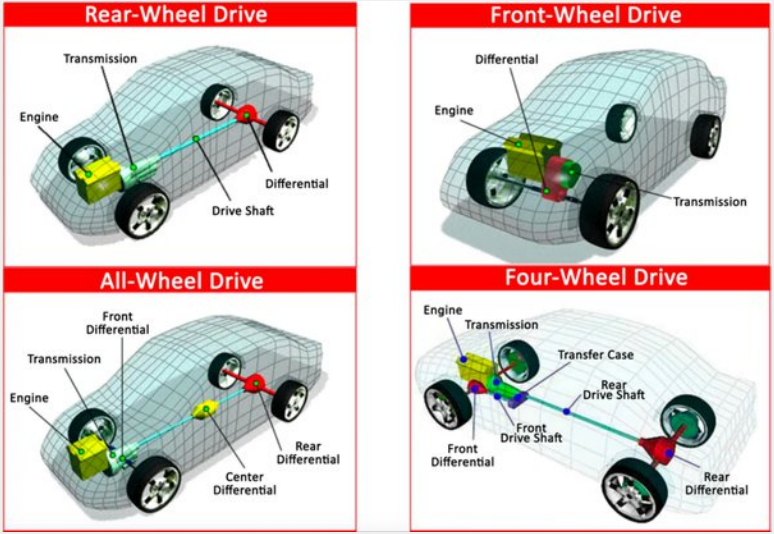
- If $tan(\alpha)$ is **positive**, it could come from either the **first** or **third** quadrant and if it is negative, it could come from either the second or fourth quadrant. Hence, atan() returns an angle from the first or fourth quadrant (i.e. $-\pi/2 <= atan() <= \pi/2$), regardless of the original 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 π 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	$3\pi/2 < \alpha < 2\pi$	-	+	-

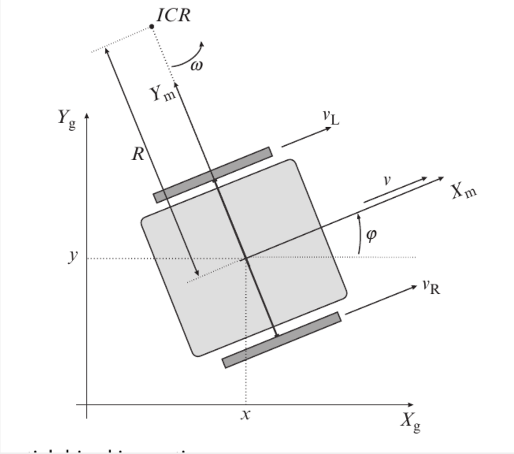
- If $tan(\alpha)$ is **positive**, it could come from either the **first** or **third** quadrant and if it is negative, it could come from either the second or fourth quadrant. Hence, atan() returns an angle from the first or fourth quadrant (i.e. $-\pi/2 <= atan() <= \pi/2$), regardless of the original 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 π to the result of atan() whenever the cosine is negative
- atan2: -pi < atan2(y,x) <pi and atan: -pi/2 < atan(y/x) < pi/2

Differential Drive Kinematics



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

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

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

- Well-fit for **smaller mobile robots**
- Usually have one or two castor wheels
- **Velocity** of **each wheel control** separately
- According to Fig. 10,
    - Terms $\mathbf{v}_R(t), \mathbf{v}_L(t)$, denoted velocity of right and left wheels, respectively

- Well-fit for **smaller mobile robots**
- Usually have one or two castor wheels
- **Velocity** of **each wheel control** separately
- According to Fig. 10,
  - Terms $\mathbf{v}_R(t), \mathbf{v}_L(t)$, 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**.

- Tangential velocity

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

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

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

- Tangential velocity

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

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

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

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

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

- Internal robot kinematics

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

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

- Internal robot kinematics

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

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

- External robot kinematics

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

Differential Drive Kinematics

- Internal robot kinematics

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

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

- External robot kinematics

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

- Discrete time dynamics using Euler integration

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

, where discrete time instance $t = kT_s$, k=0,1,2,.., for T_s
me

- 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,.., 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,.., for T_s sampling time

- We can also try trapezoidal numerical integration for better approximation

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

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

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

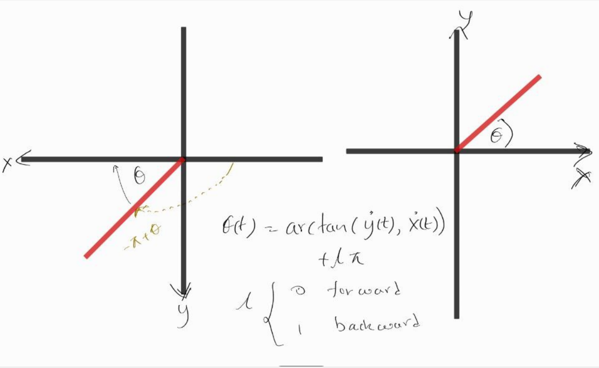
$$\begin{aligned} \mathbf{v}_R = \mathbf{v}_L = \mathbf{v}_R, \omega(t) = 0, \mathbf{v}(t) = \mathbf{v}_R // \textit{forward} \\ \mathbf{v}_R = -\mathbf{v}_L = \mathbf{v}_R, \omega(t) = 2\mathbf{v}_R / L, \mathbf{v}(t) = 0 // \textit{rotation} \end{aligned} \quad (9)$$

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

- 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., $x(t), y(t)$

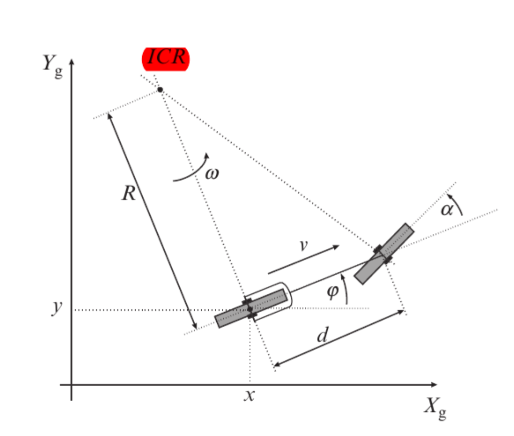
$$
\begin{aligned}
\mathbf{v}(t) &= \pm\sqrt{\dot{x}^2(t) + \dot{y}^2(t)} // \text{+ forward and - reverse} \\
\Phi(t) &= arctan2(\dot{y}(t), \dot{x}(t)) + l\pi, \quad l \in \{0, 1\} \\
&\quad // \text{ 0 forward and 1 reverse} \\
\omega(t) &= \frac{\dot{x}(t)\ddot{y}(t) - \dot{y}(t)\ddot{x}(t)}{\dot{x}^2(t) + \dot{y}^2(t)} = v(t)k(t)
\end{aligned} \tag{10}
$$

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


x
θ
$-\pi+\theta$
y
$\theta(t) = \arctan(\dot{y}(t), \dot{x}(t))$
$+ l\pi$
l
0 forward
1 backward
y
θ
x

Motion control of Bicycle mobile robots

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


ICR
Y_g
R
ω
α
v
φ
y
d
x
X_g

According to Fig. 20,

- Steering angle α, steering wheel angular velocity ω_S, ICR point is defined by intersection of both wheel axes, and distance between wheels d

According to Fig. 20,

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

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

According to Fig. 20,

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

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

- Angular velocity ω around ICR

$$\omega(t) = \dot{\Phi} = \frac{\mathbf{v}_S(t)}{\sqrt{d^2 + R^2}} = \frac{v_S(t)}{d} sin(\alpha(t)) \tag{12}$$

According to Fig. 20,

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

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

- Angular velocity ω around ICR

$$\omega(t) = \dot{\Phi} = \frac{\mathbf{v}_s(t)}{\sqrt{d^2 + R^2}} = \frac{v_S(t)}{d} sin(\alpha(t)) \tag{12}$$

- Steering wheel velocity

$$\mathbf{v}_S(t) = \omega_S(t) r \tag{13}$$

Bicycle mobile (front wheel drive)

- Internal robot kinematics

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

Bicycle mobile (front wheel drive)

- Internal robot kinematics

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

- 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

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

Motion control of Rear-Wheel Bicycle mobile robots

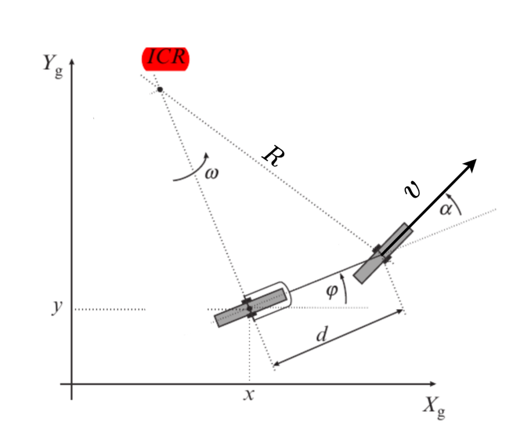
- Internal robot kinematics

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

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

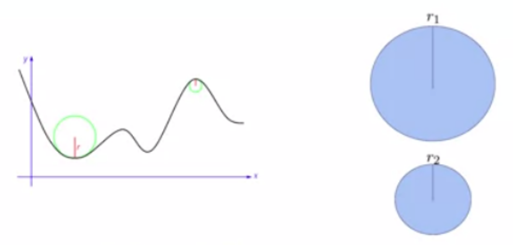


ICR
Y_g
R
ω
v
α
φ
y
d
x
X_g

- 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

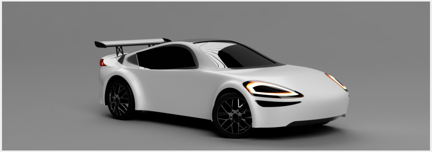


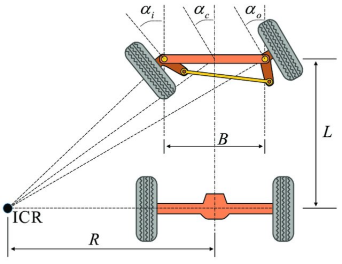
- Bicycle model imposes curvature constraint, where the curvature is defined by

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

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

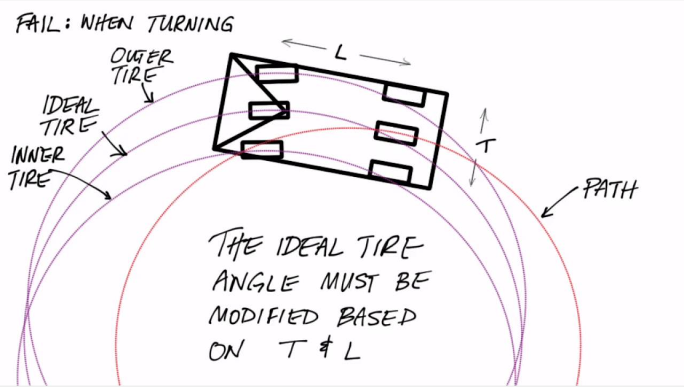
Motion control of Car(Ackermann) drive mobile robots





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

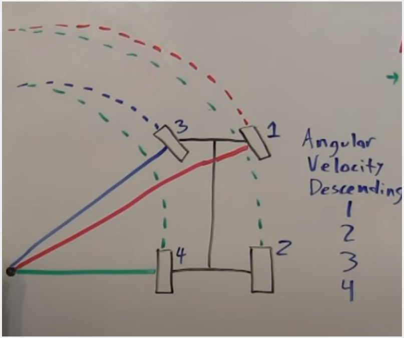


Figure: Angular velocity speed descending order

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

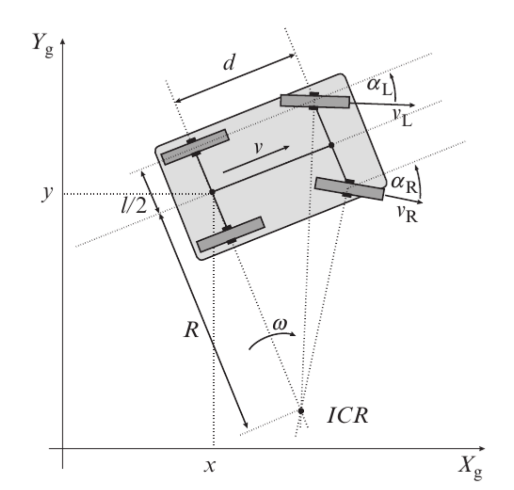
- **Ackermann geometry** is to **avoid** the need for tires to **slip sideways** when following the path around a curve which requires that the ICR point lies on a straight line defined by the rear wheels' axis
- **Ackermann geometry** can be seen as **two bicycles welded together** side by side
- For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

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

Motion control of Car(Ackermann) drive mobile robots

- Steering wheels orientations

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

- Back wheels (inner and outer) velocities

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

Motion control of Car(Ackermann) drive mobile robots

- Steering wheels orientations

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

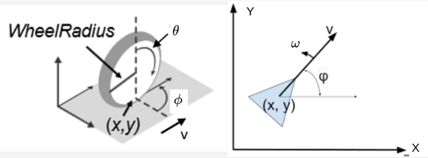
- Back wheels (inner and outer) velocities

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

- Inverse kinematics is quite complicated (TODO)

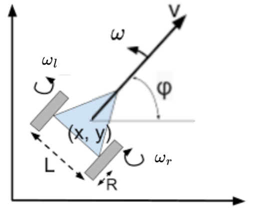
Define Mobile Robots with Kinematic Constraints

Unicycle kinematics



$$\begin{cases} \dot{x} = vcos(\phi) = rcos(\phi)\dot{\theta} \\ \dot{y} = vsin(\phi) = rsin(\phi)\dot{\theta} \\ \dot{\phi} = \omega \end{cases}$$

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 ω **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 ω **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

- **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

- 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

- 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

- 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**

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

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

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

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

- **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 $\Phi_{ref}(t)$

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

- **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

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

, where K is an arbitrary positive constant

Target (reference) Orientation Control

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

$$\begin{aligned} \alpha(t) &= K\left(\Phi_{ref}(t) - \Phi(t)\right) \\ \dot{\Phi}(t) &= \frac{\mathbf{v}_r}{d} tan(K\left(\Phi_{ref}(t) - \Phi(t)\right)) \end{aligned} \tag{24}$$

Target (reference) Orientation Control

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

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

- **For small angle** and constant velocity of rear wheels $\mathbf{v}_r(t) = V$, a linear approximation can be obtained,

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

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

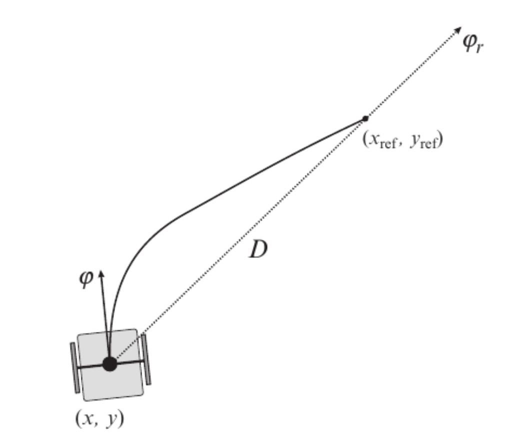
$$\mathbf{v}(\mathbf{t}) = K\sqrt{((x_{ref}(t) - x(t))^2 + (y_{ref}(t) - y(t))^2)} \tag{26}$$

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

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

- 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


φ_r
$(x_{\text{ref}}, y_{\text{ref}})$
D
φ
(x, y)

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

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

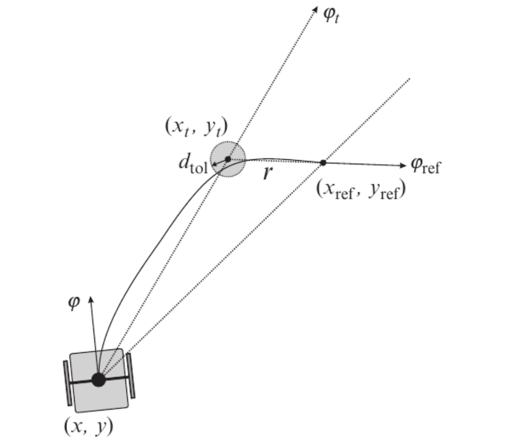
- It is required to reach the target position where the final orientation is not prescribed, hence the direction of the reference position

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

- 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

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

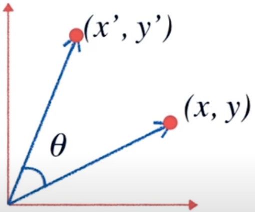
Control to reference pose via an intermediate point



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 intermediate point denoted r

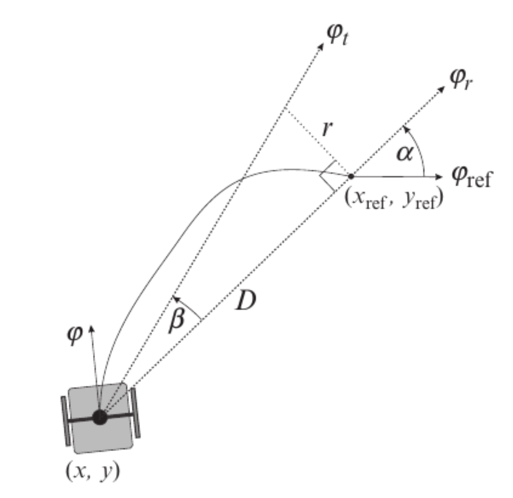
Control to reference pose via an intermediate point

- 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 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

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

Control to reference pose via an intermediate direction

- Distance between current pose and target pose

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

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

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

, 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, β increases. Thus, there are two phases:

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

$$\omega(t) = K e_\Phi(t)$$

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, β increases. Thus, there are two phases:

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

- 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 α and β 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, β increases. Thus, there are two phases:

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

- 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 α and β 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, β increases. Thus, there are two phases:

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

- 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 α and β 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