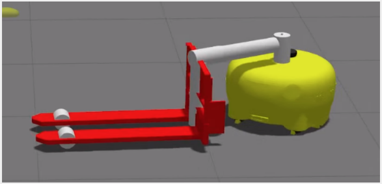
Autonomous Mobile Robotics

Motion Planning and Control

Geesara Kulathunga



September 29, 2022

Control of Mobile Robots

Contents

- 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

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 describes a system motion when forces are applied to the system and and model is described by a set of second order differential equations

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 describes a system motion when forces are applied to the system and and 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 3 DOF (Translation(2) + Rotation(1))

Kinematics of wheeled mobile robots

- The number of directions in which motion can be made is called DOF
- A car has 3 DOF (Translation(2) + Rotation(1))
- Highly efficient on hard surfaces compared to Legged locomotion

- The number of directions in which motion can be made is called DOF
- A car has 3 DOF (Translation(2) + Rotation(1))
- 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 3 DOF (Translation(2) + Rotation(1))
- Highly efficient on hard surfaces compared to Legged locomotion
- There is no direct way to measure the current pose
- Holonomic Systems - 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 3 DOF (Translation(2) + Rotation(1))
- Highly efficient on hard surfaces compared to Legged locomotion
- There is no direct way to measure the current pose
- Holonomic Systems - robot is able to move instantaneously in any direction in the space of its degree of freedom (Omnidirectional robot)
- Non-holonomic Systems - robot is not able to move instantaneously in any direction in the space of its degree of freedom

Several types of kinematics models exist

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

Several types of kinematics 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

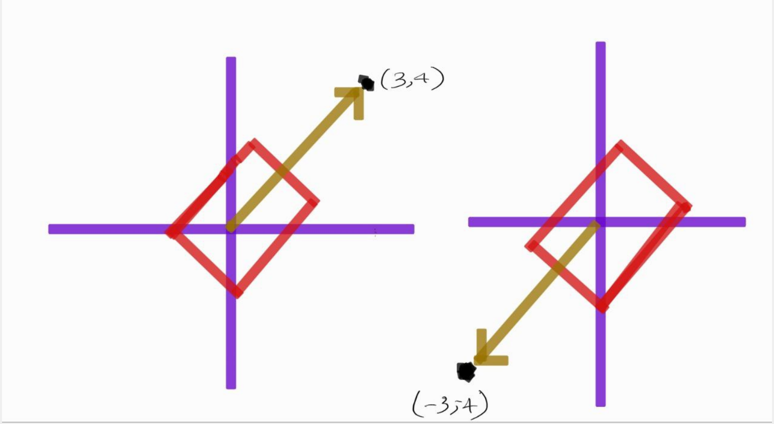
Several types of kinematics models exist

- 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 kinematics 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 desired robot pose

The difference between atan and atan2



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       -     +     -
```

- Given that the value of $tan(\alpha)$ is positive, we cannot distinguish, whether the angle was from the first or third quadrant and if it is negative, it could come from the second or fourth quadrant. So by convention, 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

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       -     +     -
```

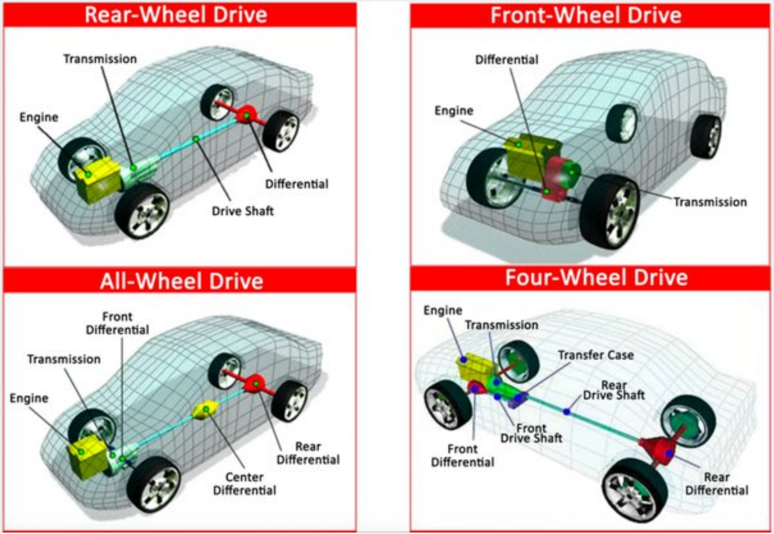
- Given that the value of $tan(\alpha)$ is positive, we cannot distinguish, whether the angle was from the first or third quadrant and if it is negative, it could come from the second or fourth quadrant. So by convention, atan() returns an angle from the first or fourth quadrant (i.e. $-\pi/2 <= atan() <= \pi/2$), regardless of the original input to the tangent
- To get full information, we must not use the result of the division $sin(\alpha)/cos(\alpha)$ but we have to look at the values of the sine and cosine 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

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	-	+	-

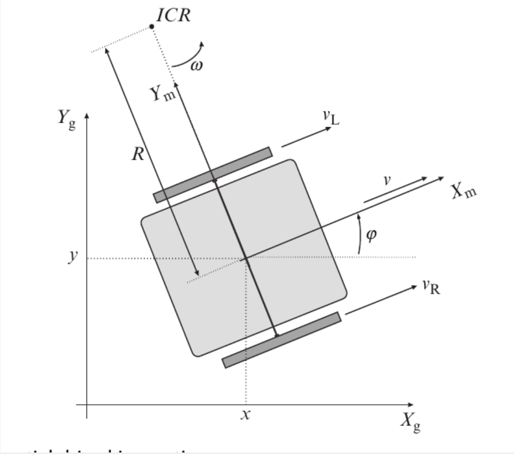
- Given that the value of $tan(\alpha)$ is positive, we cannot distinguish, whether the angle was from the first or third quadrant and if it is negative, it could come from the second or fourth quadrant. So by convention, atan() returns an angle from the first or fourth quadrant (i.e. $-\pi/2 <= atan() <= \pi/2$), regardless of the original input to the tangent
- To get full information, we must not use the result of the division $sin(\alpha)/cos(\alpha)$ but we have to look at the values of the sine and cosine 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/


ICR
ω
Y_m
Y_g
v_L
R
v
X_m
φ
y
v_R
x
X_g

- Well-fit for smaller mobile robots

- 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. 9,

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. 9,
  - Terms $\mathbf{v}_R(t), \mathbf{v}_L(t)$ 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. 9,
  - 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 instantaneous radios (ICR) of the vehicle. Angular velocity is 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

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

- 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

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

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

- 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 ...ling time

- 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

- 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 target pose how many possible ways to get there?

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

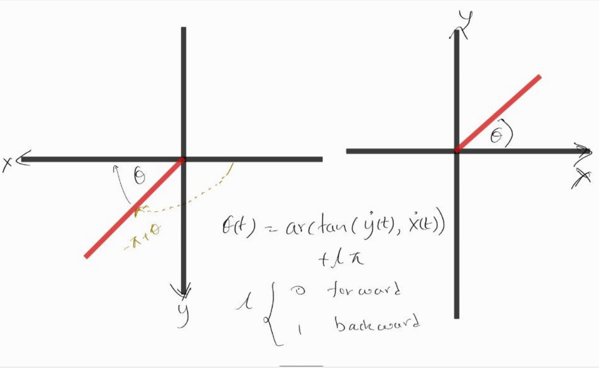
$$
\begin{aligned}
\mathbf{v}_R = \mathbf{v}_L = \mathbf{v}_R, \omega(t) = 0, \mathbf{v}(t) = \mathbf{v}_R // \textit{forward} \\
\mathbf{v}_R = -\mathbf{v}_L = \mathbf{v}_R, \omega(t) = 2\mathbf{v}_R/L, \mathbf{v}(t) = 0 // \textit{rotation}
\end{aligned}
\tag{9}
$$

- 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 disturbance in the trajectory and know the desired pose at time t, i.e., $x(t), y(t)$

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

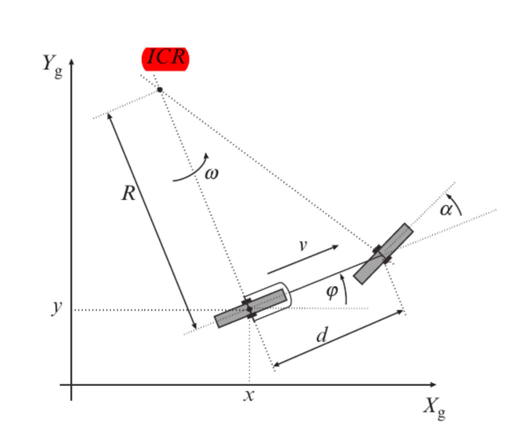
, where k(t) is the path curvature and $\omega(t) = \dot{\Phi(t)}$


x
y
θ
−π+θ
θ(t) = arctan(ẏ(t), ẋ(t))
+ l π
l
0 forward
1 backward
y
x
θ

Motion control of Bicycle mobile robots

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

Motion control of Bicycle mobile robots



ICR
Y_g
ω
R
v
α
φ
y
d
x
X_g

According to Fig. **??**,

- 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. **??**,

- 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. **??**,

- 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. **??**,

- 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

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

- 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

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

- External robot kinematics

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

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

, 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

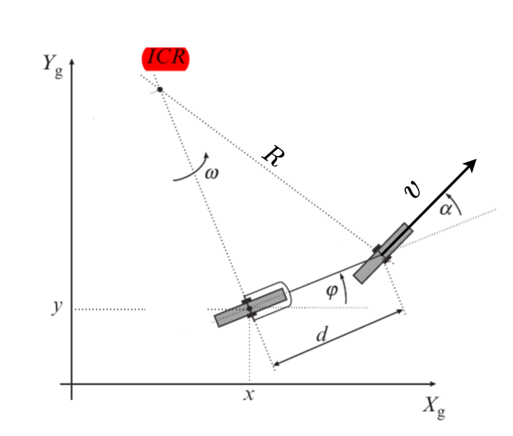
$$\begin{aligned} \dot{x}_m(t) = \mathbf{v}_S(t)cos(\alpha(t)) = \mathbf{v}_r(t) \\ \dot{y}_m(t) = 0 \\ \dot{\Phi(t)} = \frac{\mathbf{v}_r(t)}{d} tan(\alpha(t) \end{aligned} \tag{17}$$

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

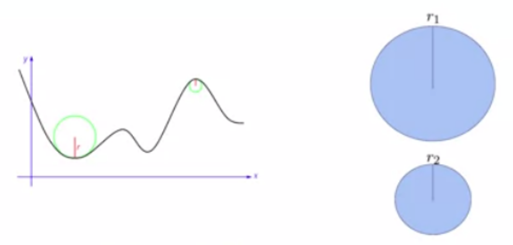
Motion control of Bicycle mobile robots



- 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

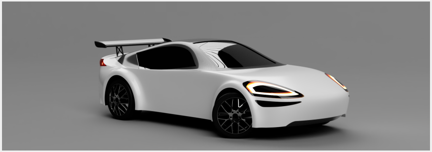


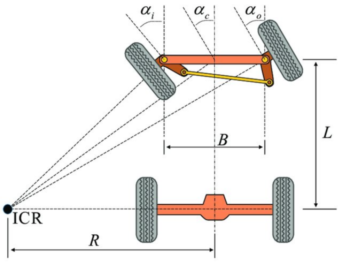
- Bicycle model imposes curvature constraint, where curvature is defined by

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

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

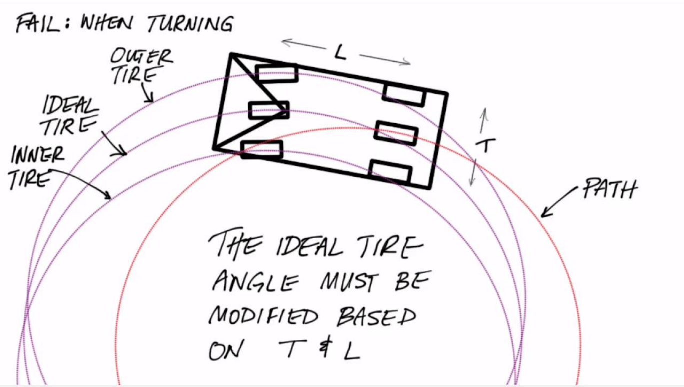
Motion control of Car(Ackermann) drive mobile robots





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

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

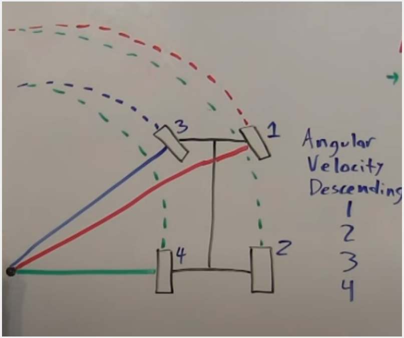


Figure: Angular velocity speed descending order

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

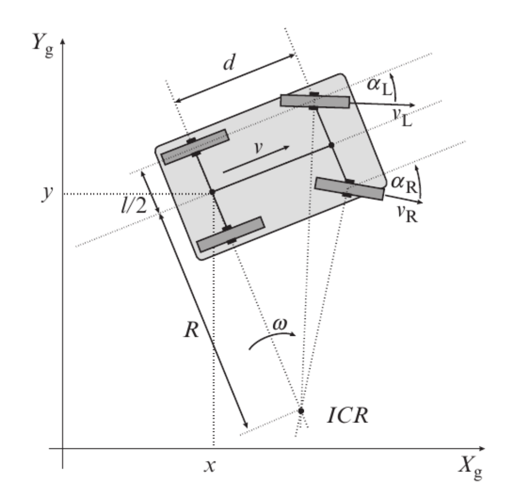
- Ackermann geometry is to avoid the need for tires to slip sideways when following the path around a curve which requires that the ICR point lies on a straight line defined by the rear wheels' axis
- Ackermann geometry can be seen as two bicycles welded together side by side
- For differential drive 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 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 true around a curve. Differentials allow the two driven wheels to run at different speeds around a curve, 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

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

- 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 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

- **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.
- **Differential-Drive Kinematic Model** uses a rear driving axle to control both vehicle speed and head rate. The wheels on the driving axle can spin in both directions. Since most mobile robots have some interface to the low-level wheel commands, this model will again use vehicle speed and heading rate as input to simplify the vehicle control.

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

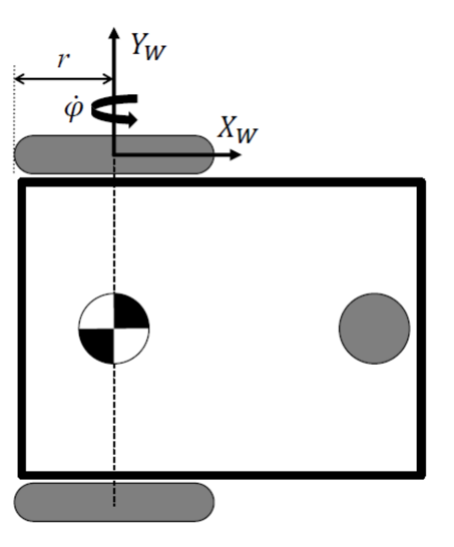
Define Mobile Robots with Kinematic Constraints

- **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 works under the assumption that wheels on each axle can be modeled as a single, centered wheel, and that the front wheel heading can be directly set, like a bicycle.

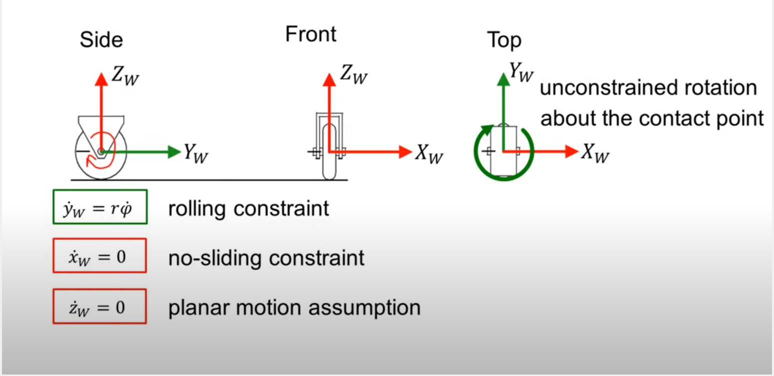
Define Mobile Robots with Kinematic Constraints

- **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 works under the assumption that wheels on each axle can be modeled as a single, centered wheel, and that the front wheel heading can be directly set, like a bicycle.
- **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 center of the vehicle's turn. This difference in turning angle is called Ackermann steering, and is typically enforced by a mechanism in actual vehicles. From a vehicle and wheel kinematics standpoint, it can be enforced by treating the steering angle as a rate

Wheel kinematics constraints

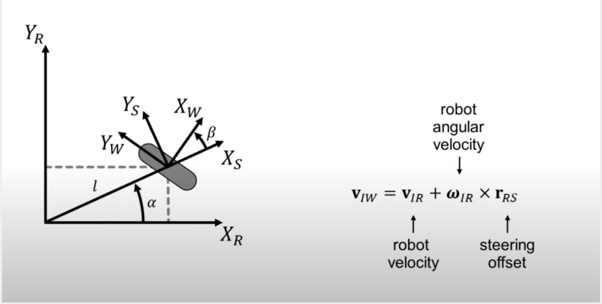


Wheel kinematics constraints



https://www.youtube.com/watch?v=hu__jYsN6mw

Wheel kinematics constraints



Coordinate frames

- I – inertial
- R – robot
- S – steering
- W – wheel

https://www.youtube.com/watch?v=hu__jYsN6mw

Wheel kinematics constraints

There are different types of wheel types, each of which has own constraints. For this course we only focus on standard wheel type. The following important assumptions are made

- Plane of wheel always remains vertical, where only one single point of contact between the and ground plane
- No sliding at this single point of contact

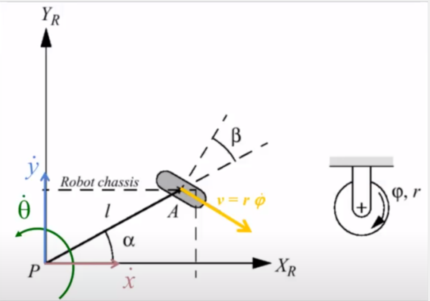
Rolling constraint

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \dot{\varphi} r \\ 0 \end{bmatrix}$$

No-sliding constraint

Wheel kinematics constraints: Fixed standard wheel

- α, β, and l locate the relative to the robot internal (local) frame
- θ is the angle between inertial x-axis and X_R (global frame)
- What differential constraints on velocity does the wheel impose on the chassis?



https://asl.ethz.ch/education/lectures/autonomous_mobile_rob

Wheel kinematics constraints: Fixed standard wheel

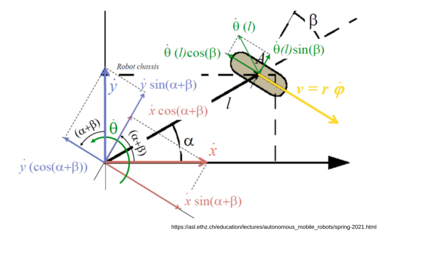
Two constraints can be derived based on those assumptions. The position A is expressed in polar coordinates by distance l and angle α

- rolling contact

$$[sin(\alpha+\beta) \quad -cos(\alpha+\beta) \quad (-l)cos(\beta)][\dot{x}\ \dot{y}\ \dot{\theta}]^{\top} - r\dot{\Phi} = 0 \tag{22}$$

- no lateral slippage

$$[cos(\alpha+\beta) \quad sin(\alpha+\beta) \quad (l)sin(\beta)][\dot{x}\ \dot{y}\ \dot{\theta}]^{\top} = 0 \tag{23}$$



Wheeled Mobile System Control

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

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

Wheeled Mobile System Control

- Can control start pose to goal pose by classical control, where intermediate state trajectory is not prescribed or reference trajectory tracking
- Nonholonomic constraints need to be considered, in such occasion, the controller is is twofold: feedforward control and feedback control, namely two-dgree-of-freedom control.
- Openloop control: feedforward control is calculated from the reference trajectory and those control action are fed to system

Wheeled Mobile System Control

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

Wheeled Mobile System Control

- Can control start pose to goal pose by classical control, where intermediate state trajectory is not prescribed or reference trajectory tracking
- Nonholonomic constraints need to be considered, in such occasion, the controller is is twofold: feedforward control and feedback control, namely two-dgree-of-freedom control.
- Openloop control: feedforward control is calculated from the reference trajectory and those control action are fed to 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, motion controlling system has to incorporate 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 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 $\Phi_{ref}(t)$

$$e_{\Phi}(t) = \Phi_{ref}(t) - \Phi(t) \tag{24}$$

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 $\Phi_{ref}(t)$

$$e_{\Phi}(t) = \Phi_{ref}(t) - \Phi(t) \tag{24}$$

- 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 $\Phi_{ref}(t)$

$$e_{\Phi}(t) = \Phi_{ref}(t) - \Phi(t) \tag{24}$$

- 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

$$\omega(t) = K(\Phi_{ref} - \Phi(t)) \tag{25}$$

, where K is an arbitrary positive constant

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

$$
\begin{aligned}
\alpha(t) &= K\left(\Phi_{ref}(t) - \Phi(t)\right) \\
\dot{\Phi}(t) &= \frac{\mathbf{v}_r}{d} tan(K\left(\Phi_{ref}(t) - \Phi(t)\right))
\end{aligned}
\tag{26}
$$

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

$$\begin{aligned} \alpha(t) &= K\,(\Phi_{ref}(t) - \Phi(t)) \\ \dot{\Phi}(t) &= \frac{\mathbf{v}_r}{d} tan(K\,(\Phi_{ref}(t) - \Phi(t))) \end{aligned} \tag{26}$$

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

$$\dot{\Phi}(t) = \frac{V}{d}(K\,(\Phi_{ref}(t) - \Phi(t))) \tag{27}$$

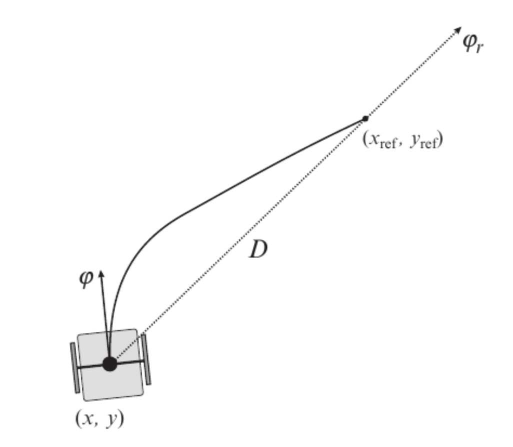
- Forward-motion control is inevitably interconnected with orientation control, i.e., forward-motion alone can not drive to goal pose without correct orientation

$$\mathbf{v}(\mathbf{t}) = K\sqrt{((x_{ref}(t) - x(t))^2 + (y_{ref}(t) - y(t))^2)} \tag{28}$$

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

$$\mathbf{v}(\mathbf{t}) = K\sqrt{((x_{ref}(t) - x(t))^2 + (y_{ref}(t) - y(t))^2)} \qquad (28)$$

- However, $\mathbf{v}(t)$ should have maximum limits, which is due to actuator limitations driving surface conditions. On the other hand, when robot get closer to goal, it might try to over take the reference pose, which is eventually lead to accelerate, which is not desired


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

Control to reference pose

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

$$
\begin{aligned}
\Phi_r(t) = arctan\frac{y_{ref} - y(t)}{x_{ref} - x(t)}, \omega(t) = K_1(\Phi_r(t) - \Phi(t)) \\
\mathbf{v}(\mathbf{t}) = K\sqrt{((x_{ref}(t) - x(t))^2 + (y_{ref}(t) - y(t))^2)}
\end{aligned}
\tag{29}
$$

Control to reference pose

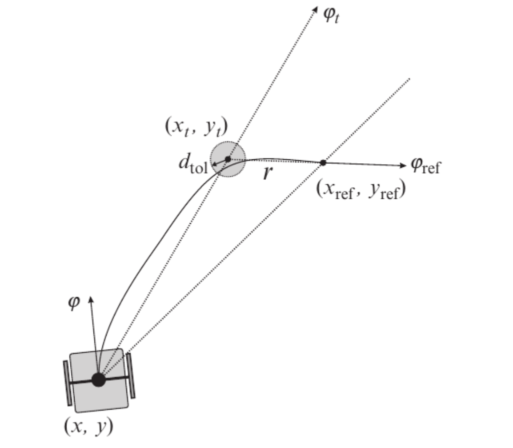
- It is required to reach to the target position where the final orientation is not prescribed, hence direction of reference position

$$\Phi_r(t) = arctan\frac{y_{ref} - y(t)}{x_{ref} - x(t)}, \omega(t) = K_1(\Phi_r(t) - \Phi(t))$$
$$\mathbf{v}(\mathbf{t}) = K\sqrt{((x_{ref}(t) - x(t))^2 + (y_{ref}(t) - y(t))^2)} \tag{29}$$

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

$$e_{\Phi}(t) = \Phi_{ref}(t) - \Phi(t), \omega(t) = K_1 arctan(tan(e_{\Phi}(t)))$$
$$\mathbf{v}(\mathbf{t}) = K\sqrt{((x_{ref}(t) - x(t))^2 + (y_{ref}(t) - y(t))^2)}.sgn(cos(e_{\Phi}(t))) \tag{30}$$

Control to reference pose via an intermediate point



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

$$\begin{aligned} x_t &= x_{ref} - r\,cos(\Phi_{ref}) \\ y_t &= y_{ref} - r\,sin(\Phi_{ref}) \end{aligned} \tag{31}$$

, where distance from reference point to intermediate point denoted r

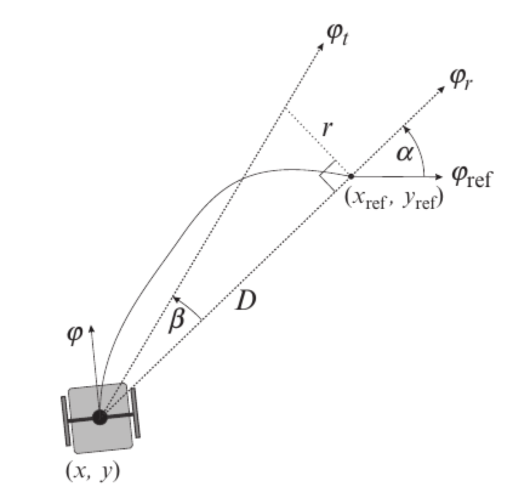
Control to reference pose via an intermediate point

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

$$
\begin{aligned}
x_t &= x_{ref} - r\,cos(\Phi_{ref}) \\
y_t &= y_{ref} - r\,sin(\Phi_{ref})
\end{aligned}
\tag{31}
$$

  , where distance from 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

$$D = \sqrt{((x_{ref}(t) - x(t))^2 + (y_{ref}(t) - y(t))^2)} \tag{32}$$

Control to reference pose via an intermediate direction

- Distance between current pose and target pose

$$D = \sqrt{((x_{ref}(t) - x(t))^2 + (y_{ref}(t) - y(t))^2)} \tag{32}$$

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

$$\begin{aligned} \alpha(t) &= \Phi_r(t) - \Phi_{ref} \\ \beta(t) &= \begin{cases} arctan\,\frac{+r}{D} & \alpha(t) > 0 \\ -arctan\frac{r}{D} & otherwise \end{cases} \end{aligned} \tag{33}$$

, 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:

$$e_{\Phi}(t) = \Phi_r(t) - \Phi(t) + \begin{cases} \alpha(t) & |\alpha(t)| < |\beta(t)| \\ \beta(t) & otherwise \end{cases} \qquad (34)$$

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

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:

$$e_\Phi(t) = \Phi_r(t) - \Phi(t) + \begin{cases} \alpha(t) & |\alpha(t)| < |\beta(t)| \\ \beta(t) & \textit{otherwise} \end{cases} \qquad (34)$$

$$\omega(t) = Ke_\Phi(t)$$

- In the first phase, $|\alpha(t)|$ is large, 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:

$$
\begin{aligned}
e_\Phi(t) = \Phi_r(t) - \Phi(t) + \begin{cases} \alpha(t) & |\alpha(t)| < |\beta(t)| \\ \beta(t) & otherwise \end{cases} \\
\omega(t) = Ke_\Phi(t)
\end{aligned}
\tag{34}
$$

- In the first phase, $|\alpha(t)|$ is large, 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

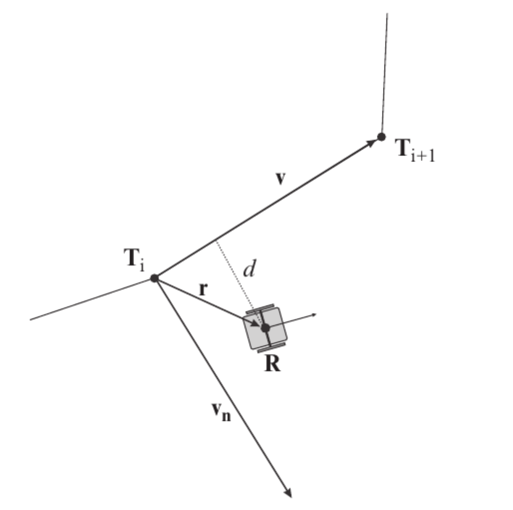
Control to reference pose via an intermediate direction

- 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:

$$e_\Phi(t) = \Phi_r(t) - \Phi(t) + \begin{cases} \alpha(t) & |\alpha(t)| < |\beta(t)| \\ \beta(t) & otherwise \end{cases} \qquad (34)$$
$$\omega(t) = Ke_\Phi(t)$$

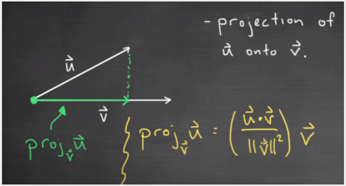
- In the first phase, $|\alpha(t)|$ is large, the robot's orientation is controlled toward the intermediate direction $\Phi_t(t) = \Phi_r(t) + \beta(t)$. When α and β become the same, the current reference orientation switches to $\Phi_t(t) = \Phi_r(t) + \alpha(t)$
- $e_\Phi(t)$ is not reducing to zero,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

Reference path control



Projection has two parts:

- The direction of projecting onto. That's the unit vector in direction of **v**, that is $\frac{\mathbf{v}}{||\mathbf{v}||}$
- The component of **u** in the direction of **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 **u**'s projection on the direction of **v**



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

The projection of **u** onto **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. Such cases, control is driven to drive on a set of straight lines with proper orientation. However, this causes nonsmooth transition between neighboring line segments

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

Reference path control

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

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

$$u = \frac{\mathbf{v}^\top \mathbf{r}}{\mathbf{v}^\top \mathbf{v}} \begin{cases} \text{Follow the current segment}(\mathbf{T}_i, \mathbf{T}_{i+1}) & \text{if } 0 < u < 1 \\ \text{Follow the next segment}(\mathbf{T}_i, \mathbf{T}_{i+1}) & \text{if } u > 1 \end{cases} \tag{35}$$

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

$$d = \frac{\mathbf{v}_n^\top \mathbf{r}}{\mathbf{v}_n^\top \mathbf{v}_n} \tag{36}$$

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

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

$$d = \frac{\mathbf{v}_n^\top \mathbf{r}}{\mathbf{v}_n^\top \mathbf{v}_n} \tag{36}$$

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

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

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

$$d = \frac{\mathbf{v}_n^\top \mathbf{r}}{\mathbf{v}_n^\top \mathbf{v}_n} \tag{36}$$

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

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

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

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

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

- Reference orientation and orientation error

$$\begin{aligned} \Phi_{ref} &= \Phi_{lin} + \Phi_{rot}, \\ e_{\Phi} = \Phi_{ref} - \Phi,\ \omega &= K_2 e_{\Phi} \end{aligned} \tag{37}$$

- Reference orientation and orientation error

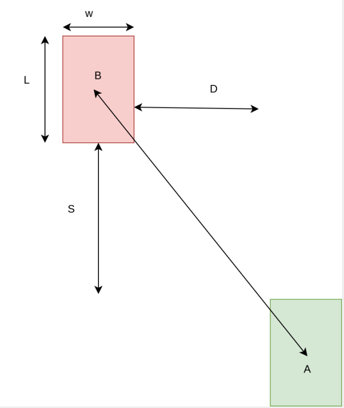
$$\begin{aligned} \Phi_{ref} &= \Phi_{lin} + \Phi_{rot}, \\ e_\Phi &= \Phi_{ref} - \Phi,\ \omega = K_2 e_\Phi \end{aligned} \tag{37}$$

- Then the controller,

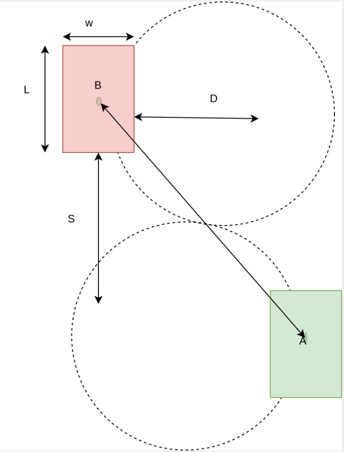
$$\begin{aligned} v &= k_p \cdot cos(e_\Phi) \\ \omega &= k_\Phi \cdot e_\Phi \end{aligned} \tag{38}$$

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

Control by circular arcs



Control by circular arcs

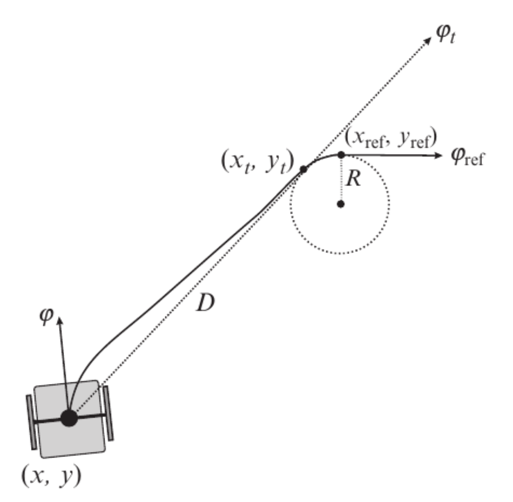


Consider each circle's radius is given by r, if the location of A and B are given by $< x_a, y_a >$ and $< x_b, y_b >$, respectively

- 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?

Control by a straight line and a circular arc



Multiplication of 2D Vectors

Quick calculation:

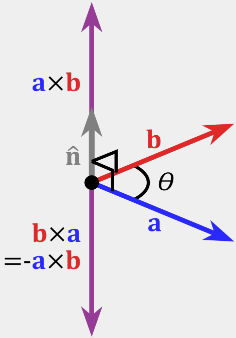
- Dot Product:

$$A \cdot B = x_a x_b + y_a y_b$$

- Cross Product:

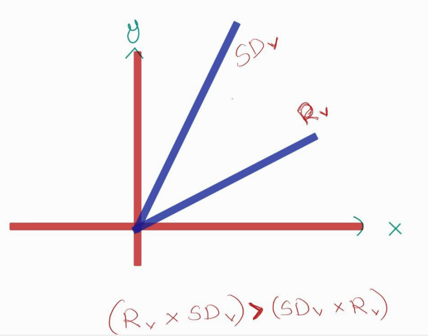
$$A \times B = (x_a y_b - y_a x_b)\,\hat{N}$$

***N** is the Normal Vector to A and B*

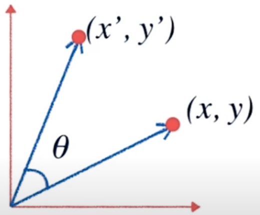


How do you define right hand side and left hand side cross product?

Control by a straight line and a circular arc



2D Rotation



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

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

rotation by a counterclockwise angle

- Circle's radius is the minimal turning radius of the vehicle

Control by a straight line and a circular arc

- Circle's radius is the minimal turning radius of the vehicle
- Path with lines and circular arcs are shortest path for Ackermann drive, however, for Diff drive, minimum arc length is zero

- Circle's radius is the minimal turning radius of the vehicle
- Path with lines and circular arcs are shortest path for Ackermann drive, however, for Diff drive, minimum arc length is zero
- The center of the circle Ps is determined by

$$Ps = \begin{cases} D + rX \cdot \mathbf{Rv}, & \mathbf{Rv} \times \mathbf{SDv} > \mathbf{SDv} \times \mathbf{Rv} \\ D + rX^{\top} \cdot \mathbf{Rv}, & \text{otherwise}, \end{cases} \tag{39}$$

  where term **SDv** is depicted the direction vector between the start S and the reference D points, matrix $X = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$, and direction vector of reference point is given by $\mathbf{Rv} = [cos(\Phi_{ref})\ sin(\Phi_{ref})]$

- Intermediate point (or temporally reference point) varies due to relative orientation

$$\begin{aligned}\mathbf{u}_1 &= S + d[cos(\Phi + \alpha])\ sin(\Phi + \alpha])] \\ \mathbf{u}_2 &= S + d[cos(\Phi - \alpha])\ sin(\Phi - \alpha])],\end{aligned} \tag{40}$$

where the distance between current pose S and intermediate point is given by $d = \sqrt{|S - Ps|^2 - r^2}$, $\alpha = atan(r/d)$ and $\Phi = atan2(Ps - S)$, respectively.

- Intermediate point (or temporally reference point) varies due to relative orientation

$$\begin{aligned} \mathbf{u}_1 &= S + d[cos(\Phi + \alpha]) \; sin(\Phi + \alpha])] \\ \mathbf{u}_2 &= S + d[cos(\Phi - \alpha]) \; sin(\Phi - \alpha])], \end{aligned} \tag{40}$$

  where the distance between current pose S and intermediate point is given by $d = \sqrt{|S - Ps|^2 - r^2}$, $\alpha = atan(r/d)$ and $\Phi = atan2(Ps - S)$, respectively.
- When moving towards intermediate point, the intermediate point D changes from the reference point (initially, D depicts the distance between S and R), hence, D is calculated as follows:

$$D = \begin{cases} \mathbf{u}_1, & \Big((\mathbf{u}_1 - S) \times (Ps - \mathbf{u}_1)\Big) \cdot \Big(\mathbf{Rv} \times (Ps - D)\Big) \geq 0 \\ \mathbf{u}_2, & \text{otherwise} \end{cases} \tag{41}$$

- Depends on the distance to intermediate point, i.e., $\|Ps - S\|_{l2} < r$, robot direction vector **Dv** is determined as follows:

$$\mathbf{Dv} = \begin{cases} \mathbf{Rv}, & \|Ps - S\|_{l2} < r \\ \mathbf{SDv} / |\mathbf{SDv}| + eps, & \text{otherwise} \end{cases} \tag{42}$$

, where $\mathbf{SDv} = D - S$

- If $|\mathbf{DTv} \times \mathbf{Ev}| < 0.0001$, where $\mathbf{Ev} = (D - S),\ \mathbf{DTv} = X \cdot \mathbf{Dv}$, drives on a straight line. Thus, robot direction vector

$$\begin{aligned} \mathbf{Sv} &= \mathbf{SDv}/(|\mathbf{SDv}| + eps) \\ \gamma &= 0 \\ l &= |Ps - S| \end{aligned} \tag{43}$$

- If $|\mathbf{DTv} \times \mathbf{Ev}| > 0.0001$, where $\mathbf{Ev} = (D - S),\ \mathbf{DTv} = \mathbf{Dv}$, drives on a circle. Thus, robot direction vector

$$\begin{aligned} \mathbf{Sv} = a \cdot X \cdot (C - S), \quad \mathbf{Sv} = \mathbf{Sv}/(|\mathbf{Sv}| + eps) \\ \gamma = a \cdot acos(\mathbf{Dv} \cdot Sv) \\ \gamma = \begin{cases} a \cdot 2\pi - \gamma, & a \cdot \mathbf{Sv} \times Dv < 0 \\ \gamma, & \text{otherwise} \end{cases} \end{aligned} \tag{44}$$

, where

$$\begin{aligned} a = \begin{cases} 1, & \mathbf{SDv} \times \mathbf{Dv} > 0 \\ -1, & \text{otherwise} \end{cases} \\ C = \frac{\mathbf{DTv} \cdot \mathbf{Ev} \times (D - M)}{(\mathbf{DTv} \times \mathbf{Ev}) + D} \\ l = \left|\gamma \cdot |S - C|_{l2}\right|, \quad M = (D + S)/2 \end{aligned} \tag{45}$$

- If $v > eps$

$$
\begin{aligned}
vDir &= \begin{cases} -1, & \mathbf{Ov} \cdot \mathbf{Sv} < 0 \\ 1, & \text{otherwise} \end{cases} \\
e_\Phi &= acos(vDir \cdot \mathbf{Sv} \cdot \mathbf{Ov}) \\
e_\Phi &= \begin{cases} -e_\Phi, & vDir \cdot \mathbf{Ov} \cdot \mathbf{XSv} < 0 \\ e_\Phi, & \text{otherwise} \end{cases} \\
v &= v + v_{Max} \cdot Ts \\
dt &= l/v \\
\omega &= \gamma/dt + e_\Phi/\Big(dt \cdot 10 \cdot (1 - exp(-l2/0.1))\Big)
\end{aligned} \tag{46}
$$

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.