深蓝学院
shenlanxueyuan.com

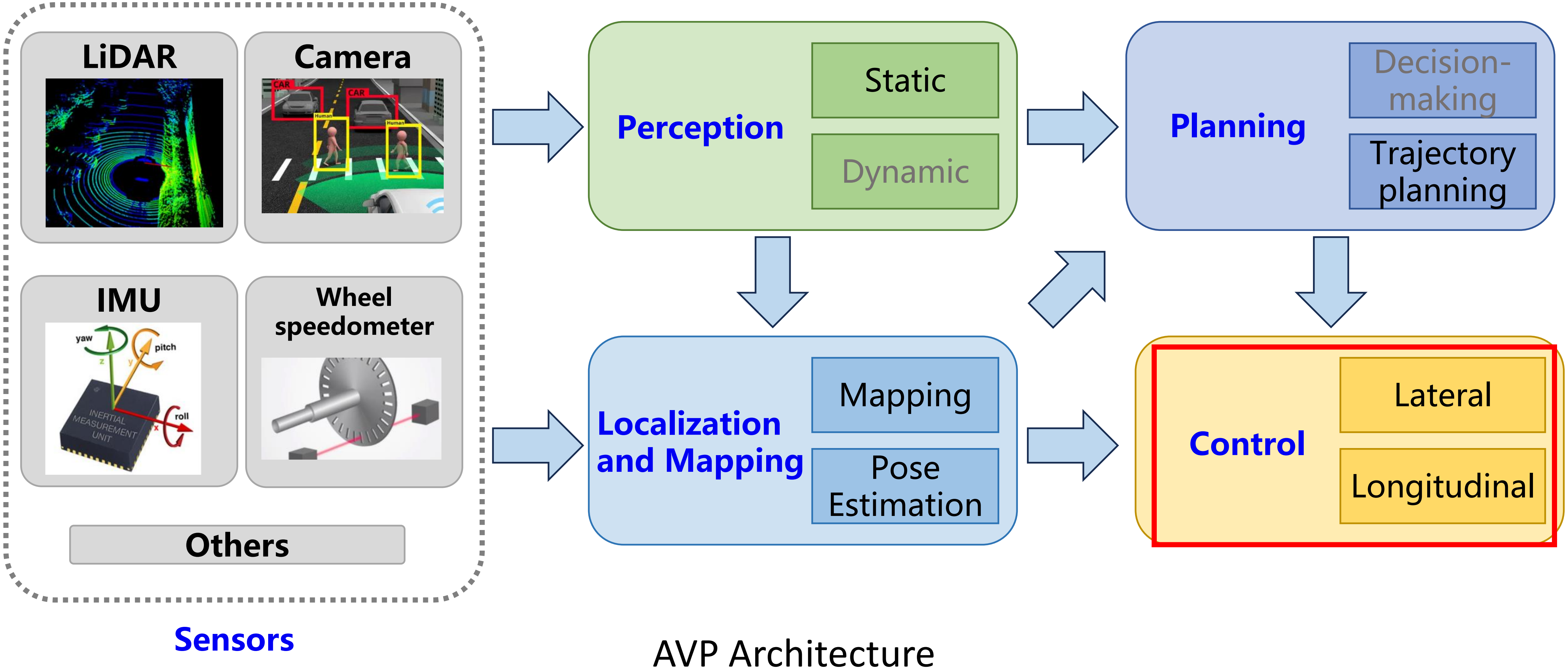
Autonomous Valet Parking（AVP）Theory and Practice
自主代客泊车理论与实践

Lecture 8：Vehicle Control

Lecturer Tong QIN

Associate Professor
Shanghai Jiao Tong University
Global Institute of Future Technology
qintong@sjtu.edu.cn

Content



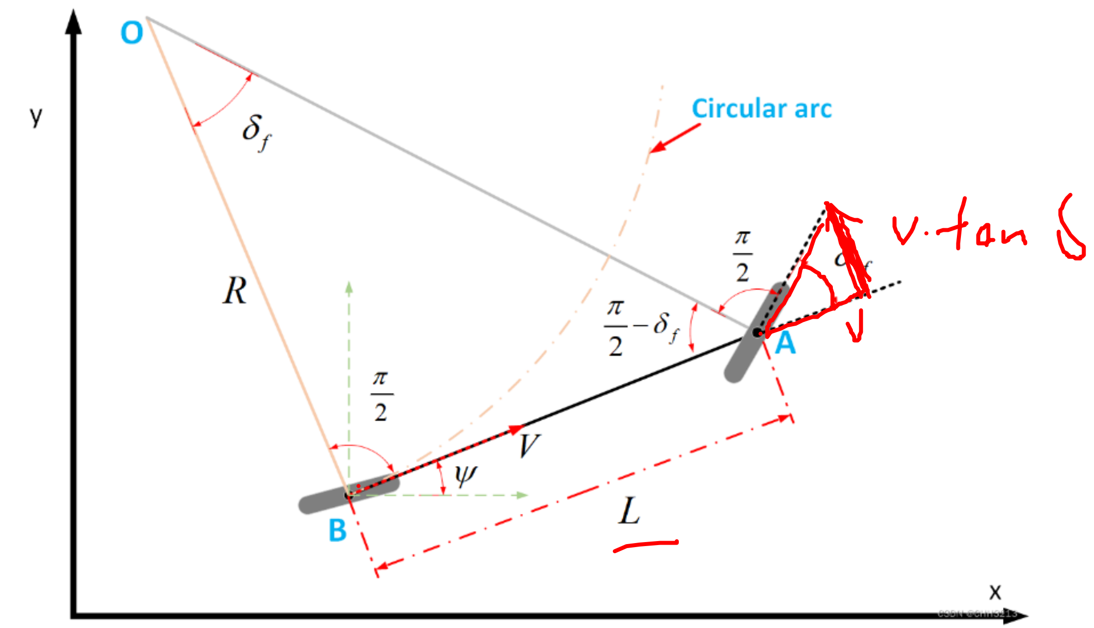
AVP Architecture

Content

Vehicle Kinematic Model

The bicycle model

- the rear axle center is the vehicle center.



State: $\boldsymbol{\chi} = [x, y, \psi]^T$

Control input : $\mathbf{u} = [v, \delta]^T$

Longitudinal control

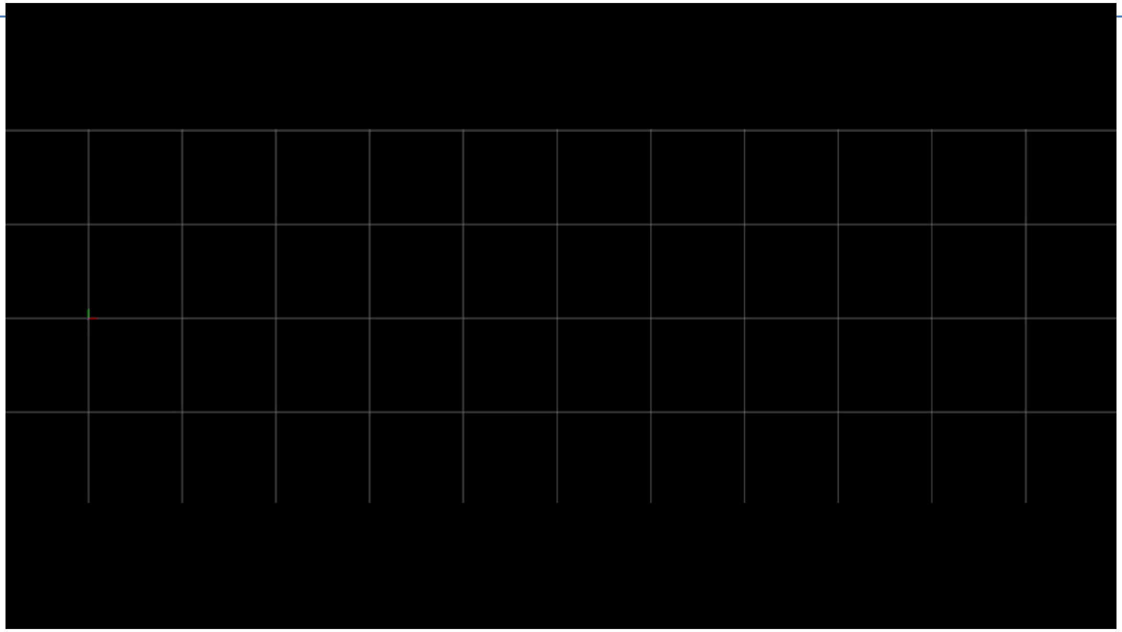
Lateral control

$$\dot{\chi} = f(\chi, u) \Rightarrow \begin{cases} \dot{x} = v\cos(\psi) \\ \dot{y} = v\sin(\psi) \\ \dot{\psi} = \frac{v}{L}\tan\delta_f \end{cases}$$

Trajectory tracking

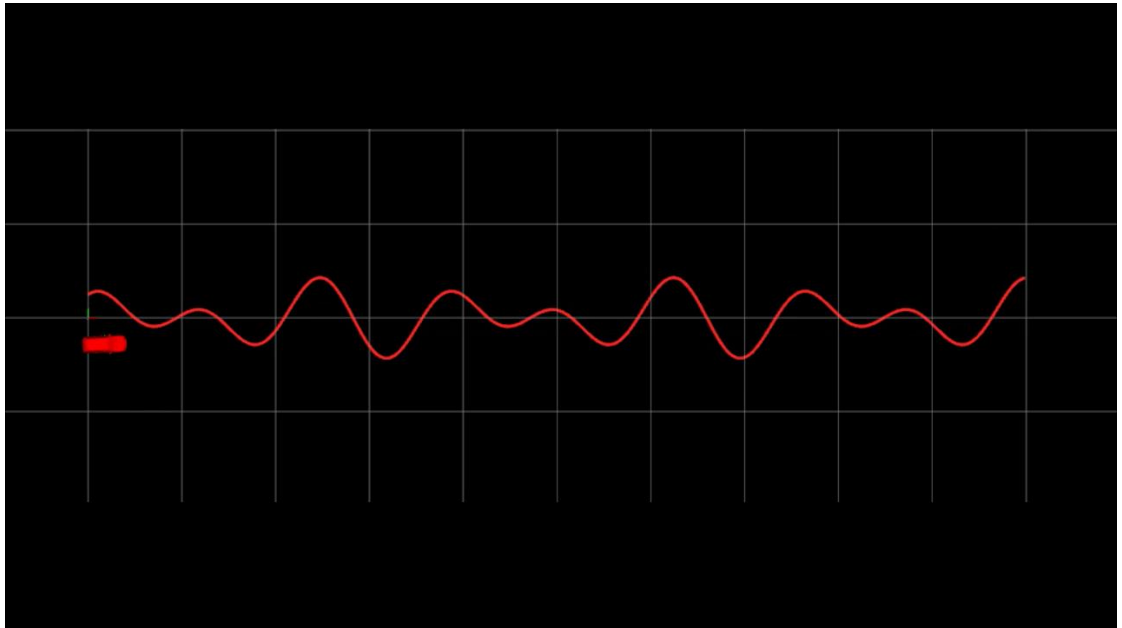
PID

Controller



LQR

Controller

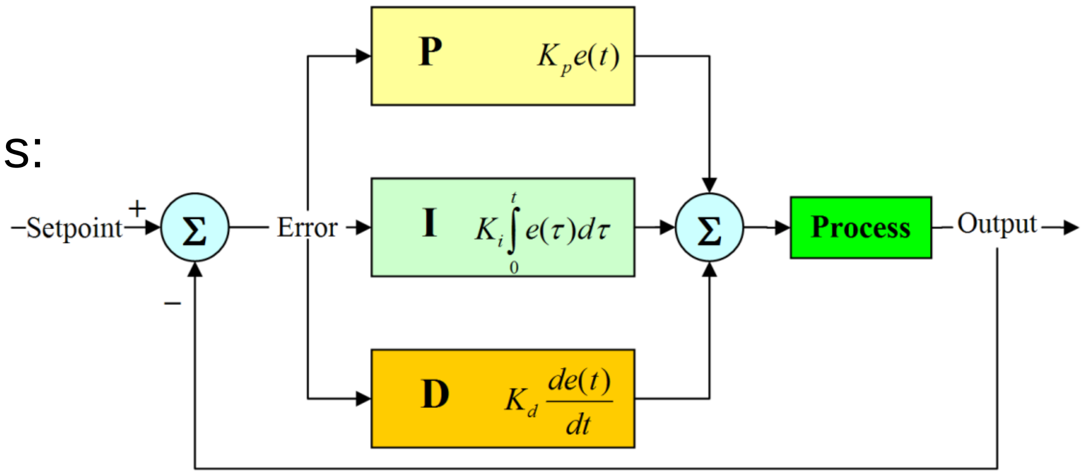


PID Controller

PID Controller

PID (Proportional-Integral-Derivative) controller is the most widely used controller in industrial applications:

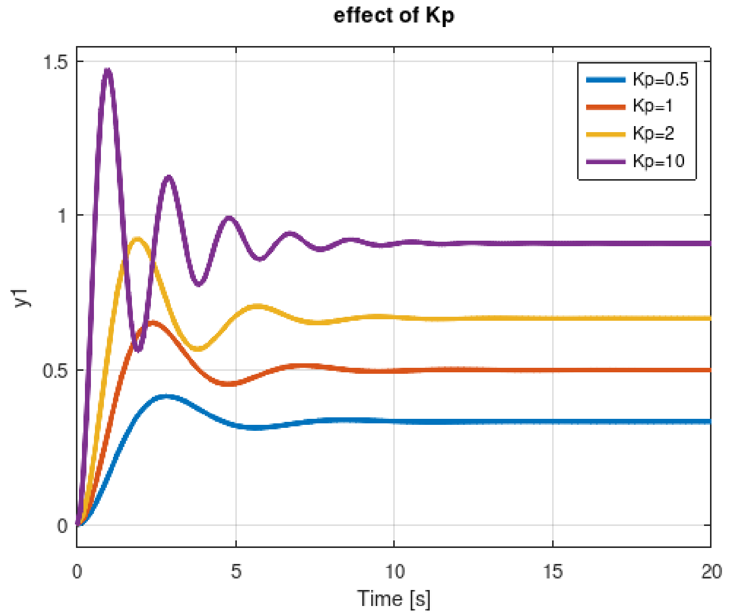
- proportional unit
- integral unit
- derivative unit



$$u(t) = K_P e(t) + K_I \int_0^t e(\tau)\, d\tau + K_D \frac{de(t)}{dt}$$

K_P: Proportional gain
K_I: Integral gain
K_D: Derivative gain
e: Error = Setpoint − Current value
t : Current time
τ: Integral variable, ranging from 0 to the current time t

PID Controller



The proportional (P), integral (I), and derivative (D) units of a PID controller correspond to the current error, the accumulated past error, and the future error, respectively.

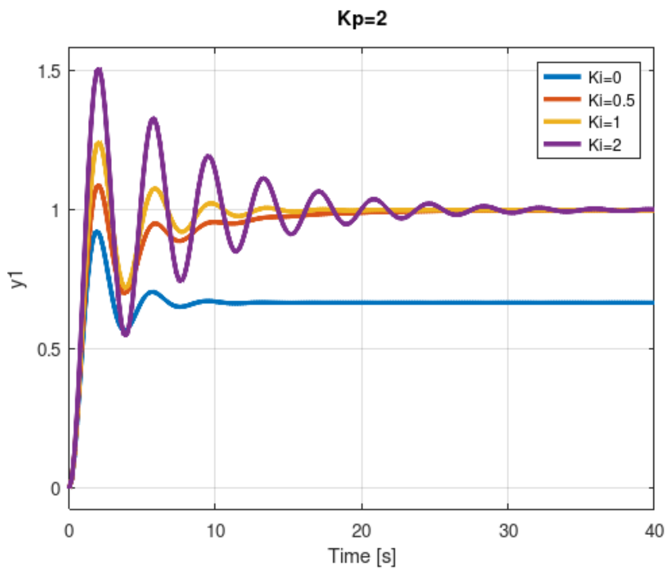
$$u(t) = \boxed{K_P e(t)} + K_I \int_0^t e(\tau)\, d\tau + K_D \frac{de(t)}{dt}$$

Proportional (P)

- Increasing the proportional term will accelerate the system's response.
- It acts on the output value quickly, but it cannot stabilize the system at an ideal value, it may lead to a steady-state error.
- A very large proportional gain can cause significant overshoot and oscillations, which can degrade system stability.

PID Controller

$$u(t) = K_P e(t) + \boxed{K_I \int_0^t e(\tau)\, d\tau} + K_D \frac{de(t)}{dt}$$



Discrete system: $\int_0^t e(\tau)\, d\tau \approx T \sum_{j=0}^{k} e_j$

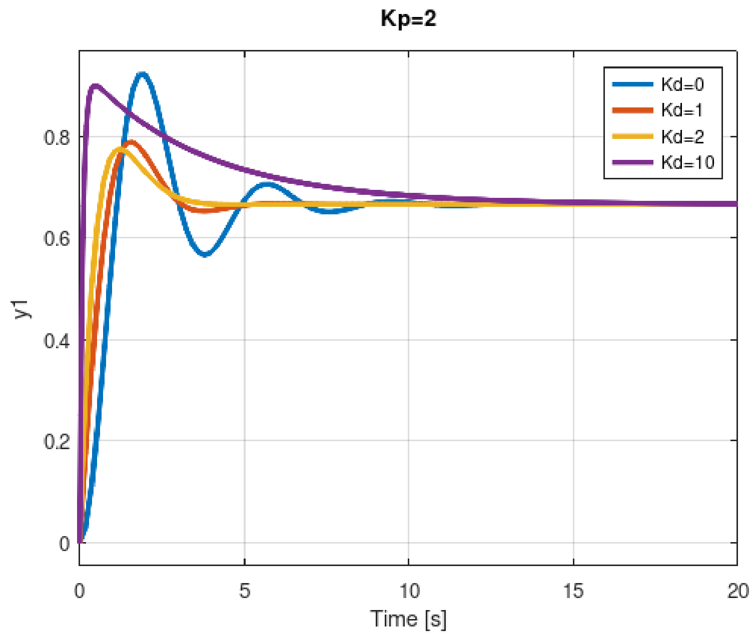
T: sampling time interval

Integral (I)

- The integral term eliminates steady-state error on top of the proportional action.
- Note: when the control output $u(k)$ reaches its maximum or minimum value, the integral action should be stopped, and there should be integral clamping and output clamping

PID Controller

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau)\, d\tau + \boxed{K_D \frac{de(t)}{dt}}$$



Discrete system: $\frac{de(t)}{dt} \approx \frac{e_k - e_{k-1}}{T}$

T: sampling time interval

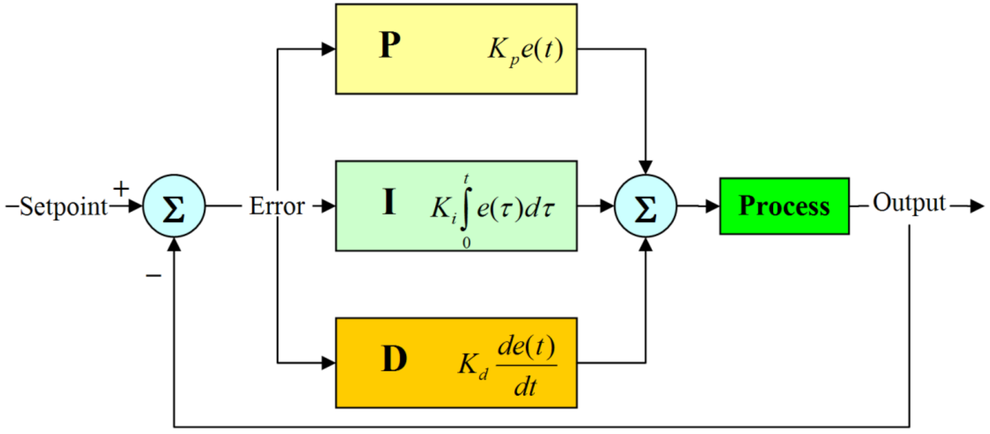
Derivative (D)

- The derivative term has a predictive effect. For control channels with time delays, it can significantly improve the system's dynamic performance.
- It can reduce overshoot, minimize oscillations, improve stability, and decrease dynamic errors.

PID Controller

PID control technology is the most convenient to use:

- when the mathematical model and parameters of the controlled object cannot be obtained



Continues system:

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau)\, d\tau + K_D \frac{de(t)}{dt}$$

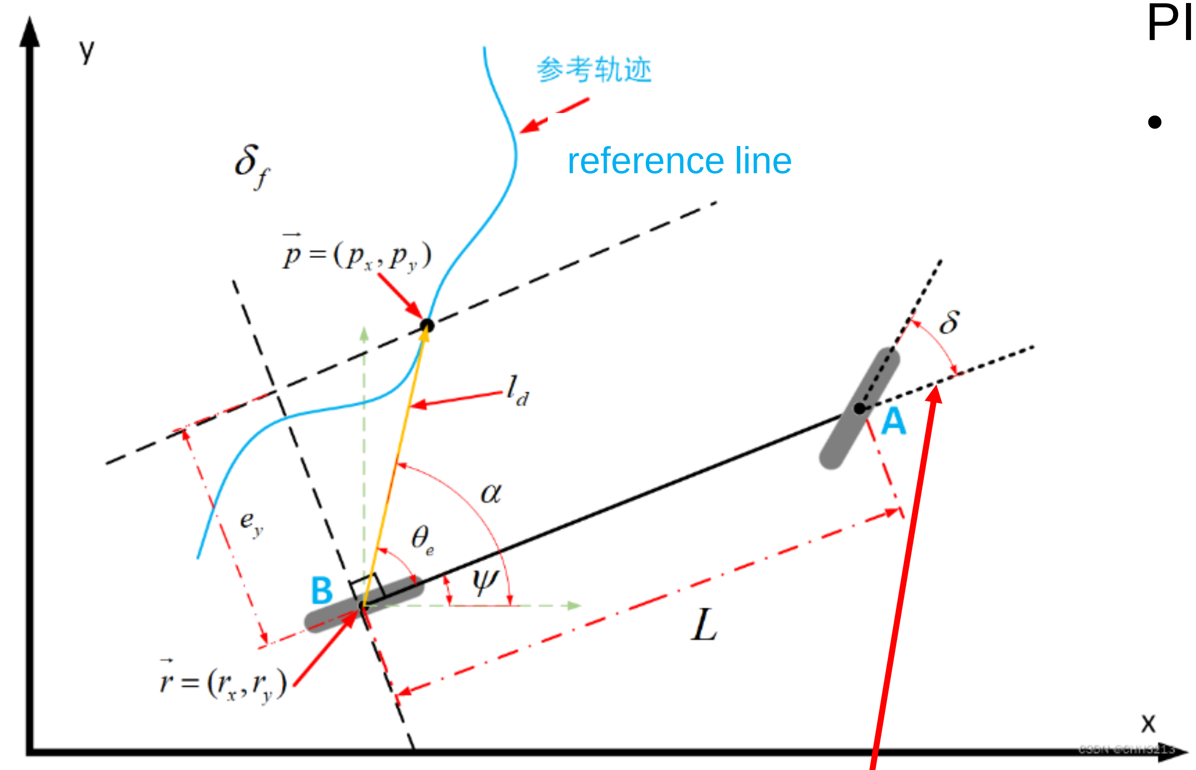
Discrete system:

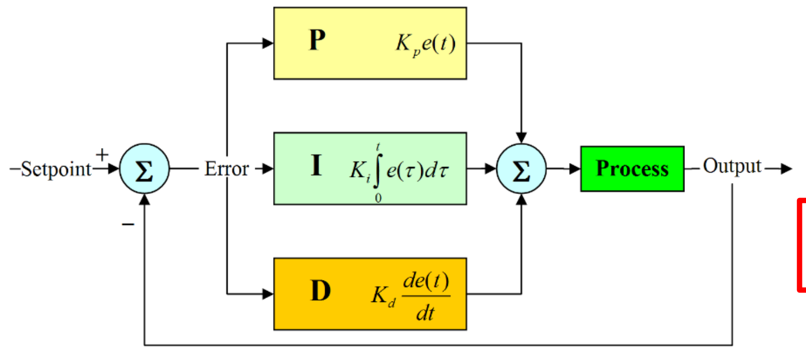
$$u_k = K_P e_k + K_I T \sum_{j=0}^{k} e_j + K_D \frac{e_k - e_{k-1}}{T}$$

$$u_k = K_P e_k + K_I \sum_{j=0}^{k} e_j + K_D (e_k - e_{k-1})$$

The characteristics of the controller can be adjusted by tuning the gains K_P , K_I , and K_D for the proportional, integral, and derivative terms, respectively.

PID Controller





PID for lateral control

$\mathbf{u} = [v, \delta]^T$

Lateral control

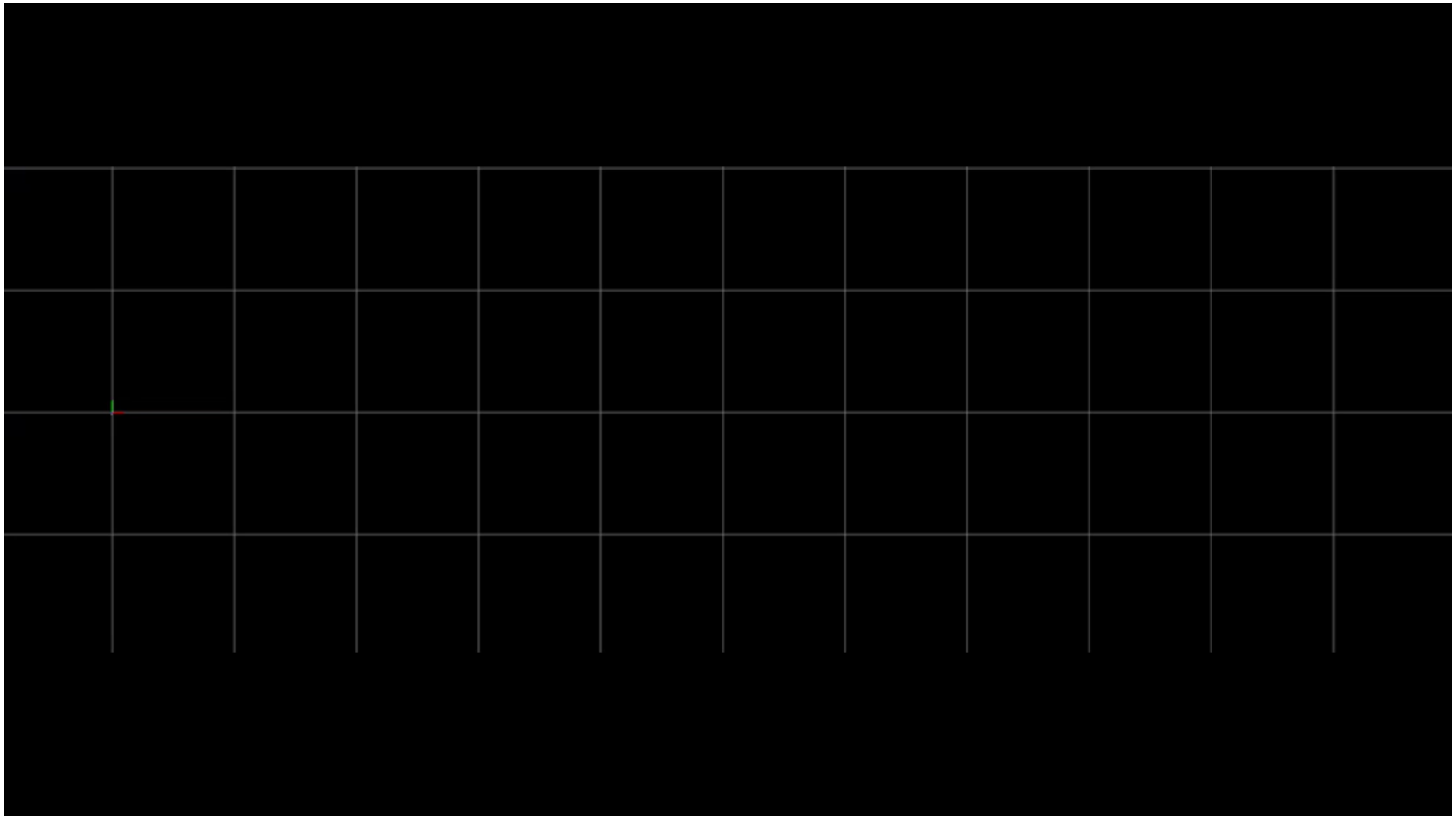
- Lateral error e_y
  - Vehicle center (r_x, r_y) to reference point (p_x, p_y)

$$\boxed{e_y} = l_d \sin \theta_e$$

$$\theta_e = \alpha - \psi = \arctan \frac{p_y - r_y}{p_x - r_x} - \psi$$

$$\boxed{u_k} = K_P e_k + K_I \sum_{j=0}^{k} e_j + K_D (e_k - e_{k-1})$$

PID Controller



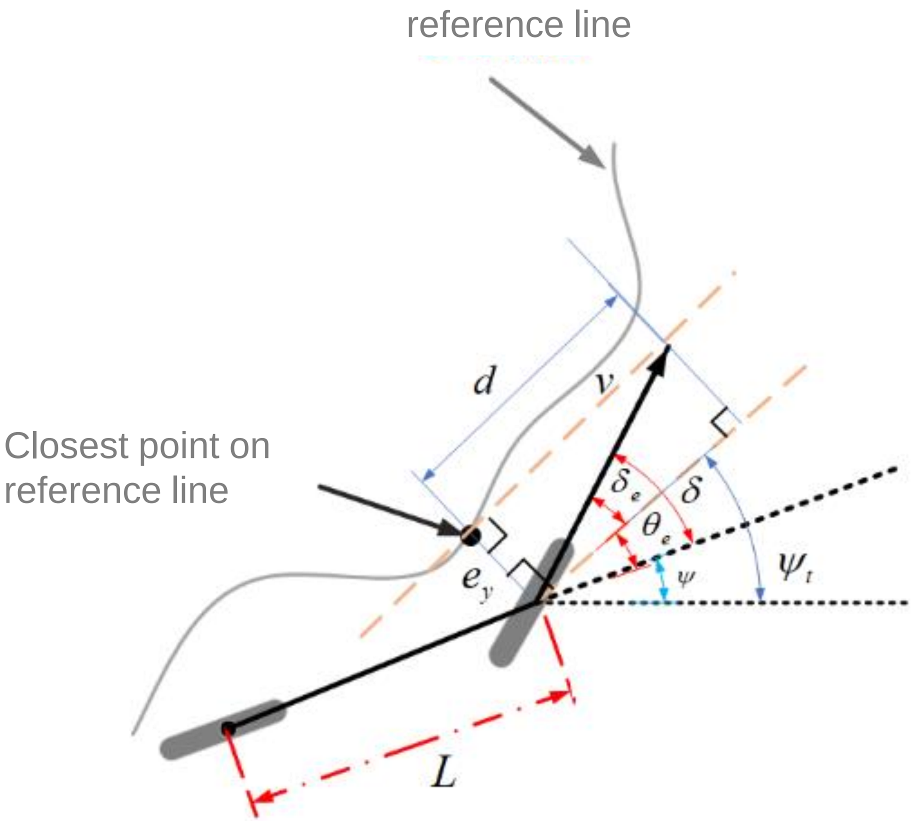
Stanley Controller

Stanley Method

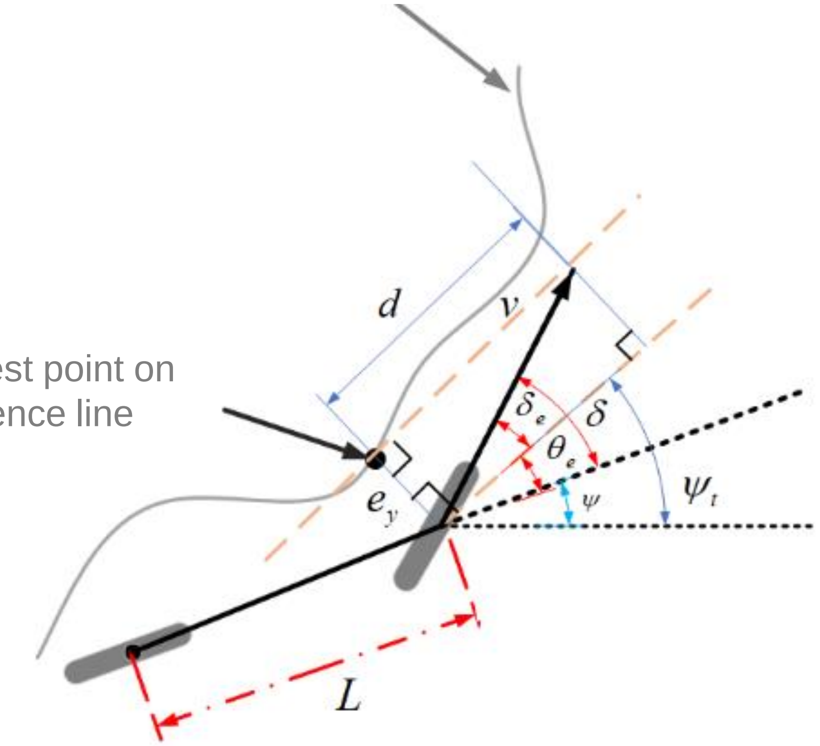
Stanley controller (Front wheel feedback control), is based on calculating the lateral path tracking error at the front wheel center.

The front wheel steering control input consists of two parts:

- lateral error
  - the lateral distance from the front axle center to the nearest point on the reference path.
- heading error
  - current vehicle orientation ψ
  - reference path heading ψ_t



Stanley Method



d is proportional to the vehicle speed *v*

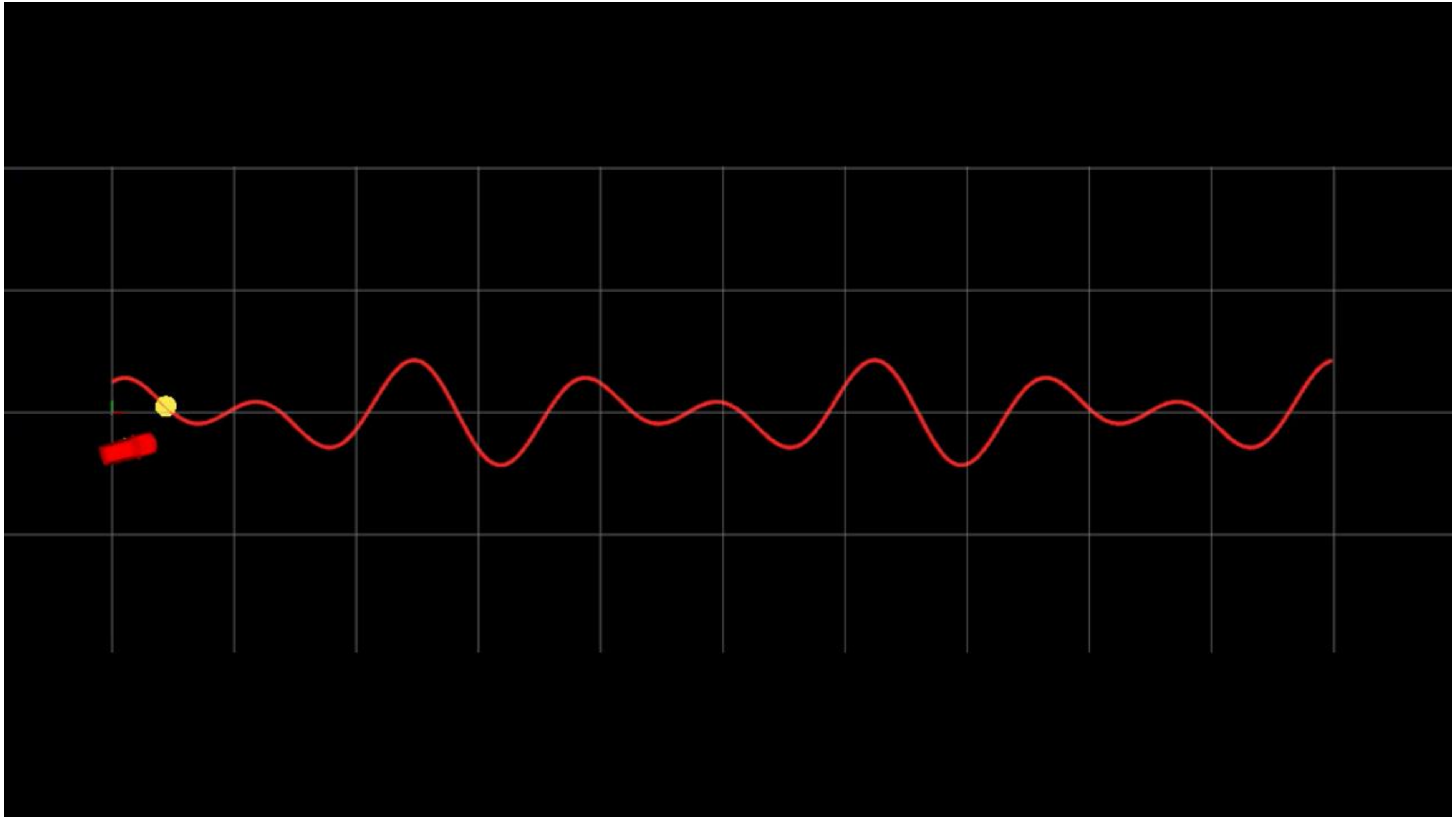
$$d = \frac{v}{k}$$

Lateral error $\delta_e = \arctan \frac{e_y}{d} = \arctan \frac{ke_y}{v}$

Heading error $\theta_e = \psi_t - \psi$

$$\delta = \theta_e + \delta_e$$

Stanley Method

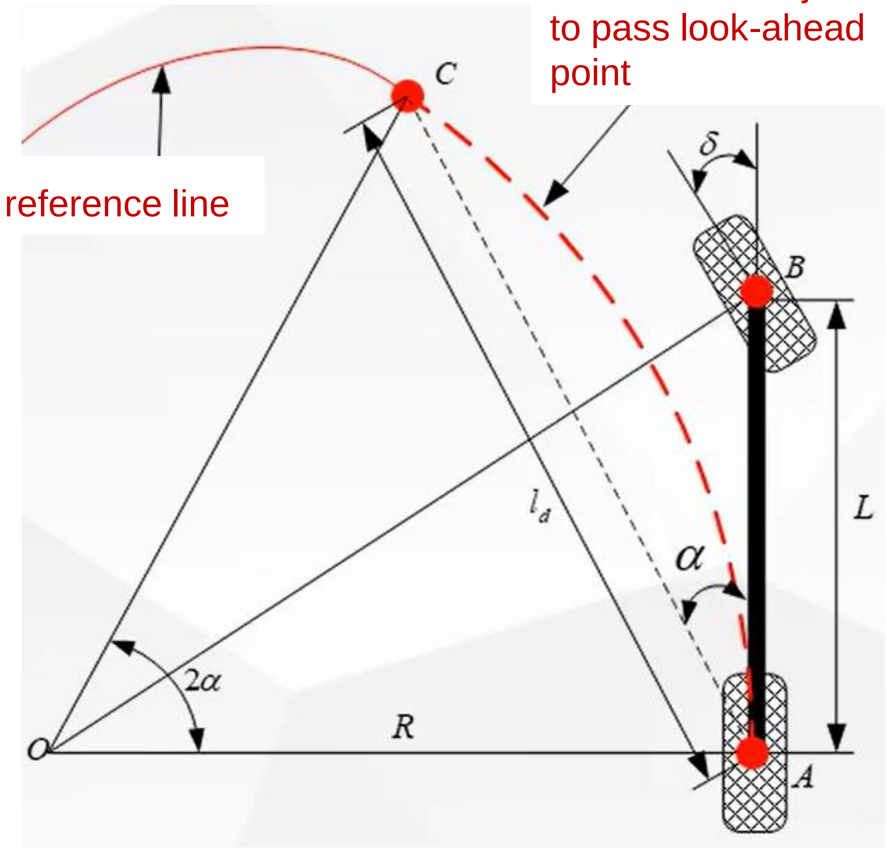


Pure Pursuit

Pure Pursuit

Pure Pursuit

- Adjusts the front wheel steering angle δ to pass look-ahead point



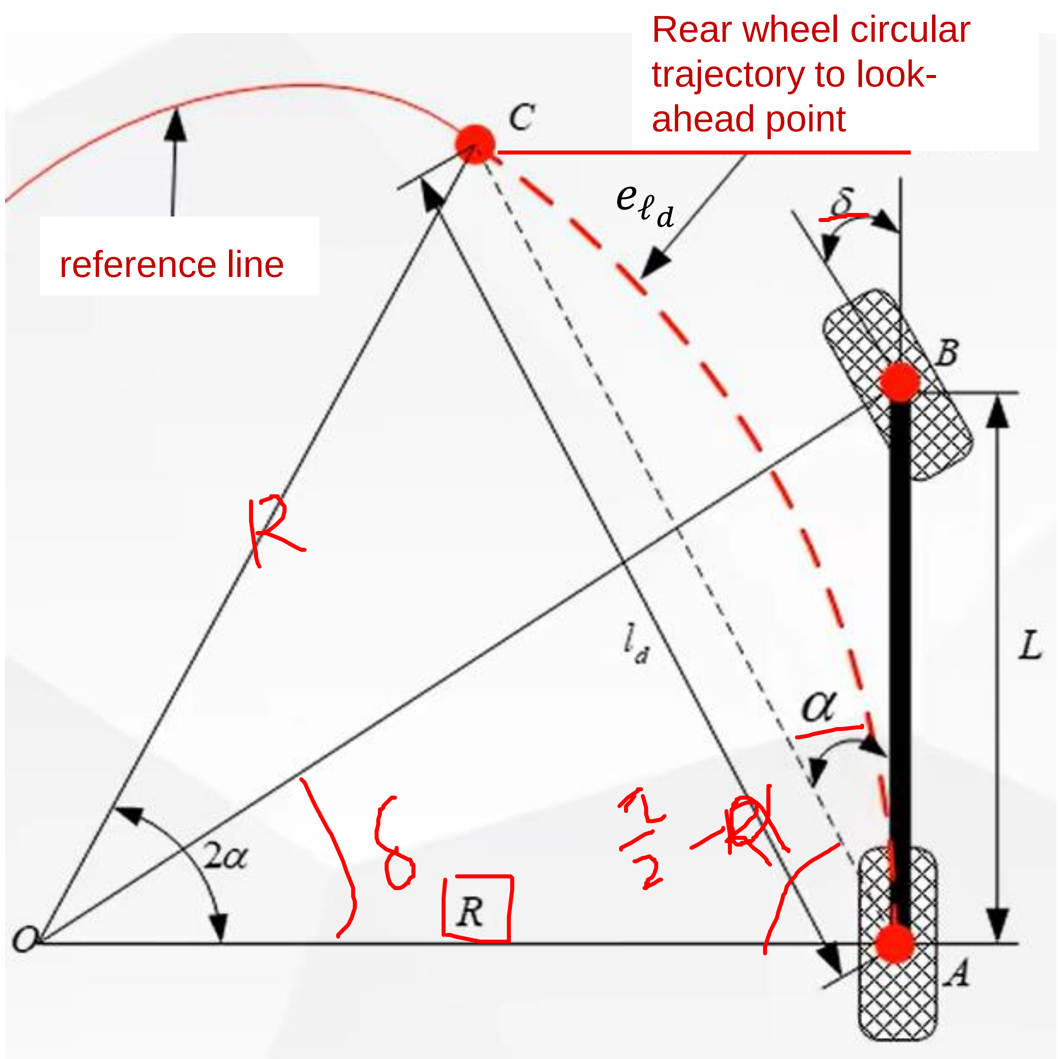
look-ahead distance l_d

Pure Pursuit

law of sines

$$\frac{a}{\sin\alpha} = \frac{b}{\sin\beta} = \frac{c}{\sin\gamma}$$

$$\sin(2A) = 2\sin(A)\cos(A)$$



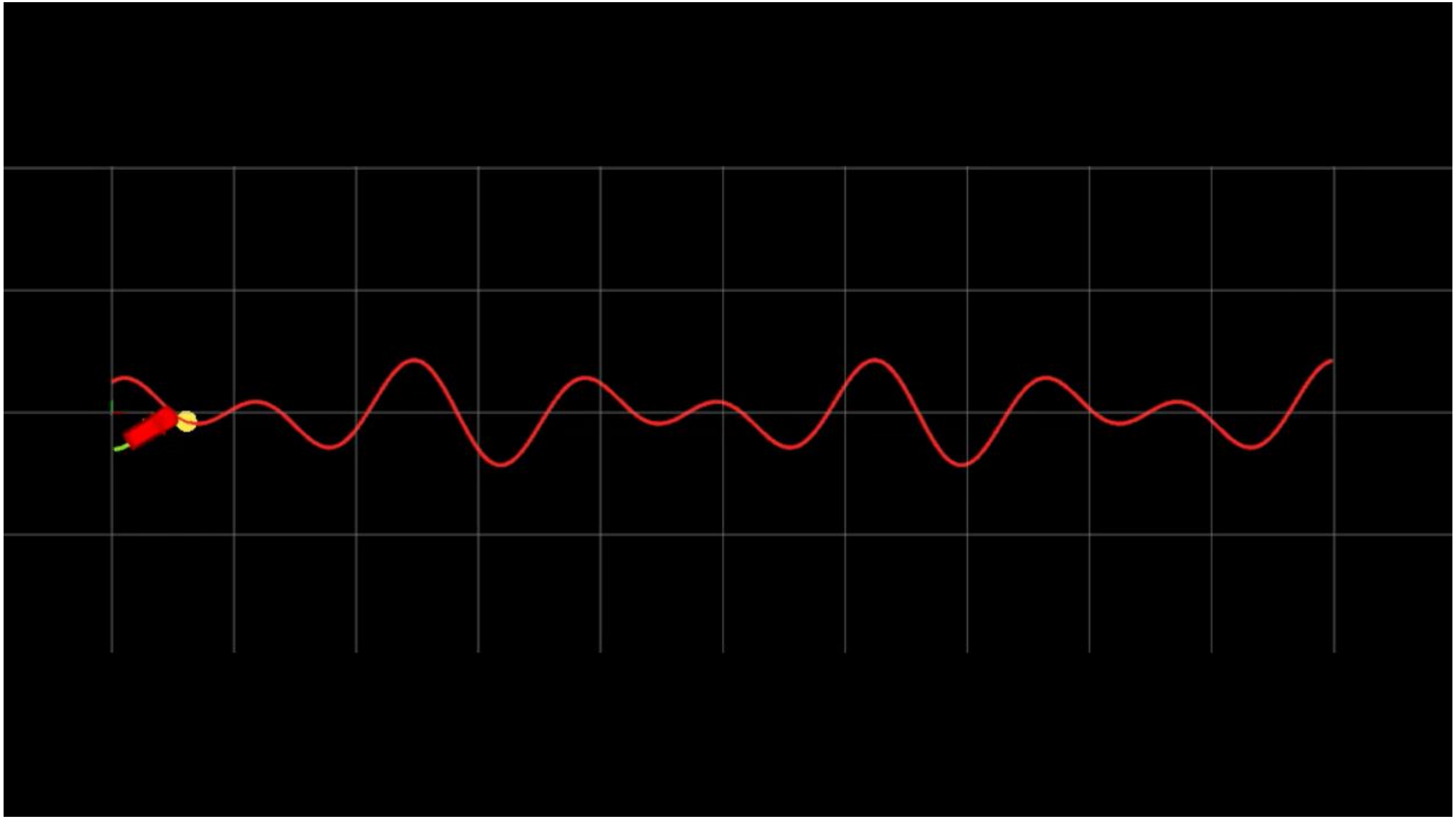
In ΔOCA, $\dfrac{\ell_d}{sin(2\alpha)} = \dfrac{R}{sin\left(\frac{\pi}{2} - \alpha\right)} \Rightarrow \dfrac{\ell_d}{2sin(\alpha)cos(\alpha)} = \dfrac{R}{cos(\alpha)} \Rightarrow$

$$R = \frac{\ell_d}{2sin(\alpha)}$$

In ΔOAB, $\tan\delta = \frac{L}{R} \Rightarrow \delta = tan^{-1}\left(\frac{L}{R}\right)$

$$\delta(t) = tan^{-1}\left(\frac{2Lsin(\alpha(t))}{\ell_d}\right)$$

Pure Pursuit



LQR Controller
(Linear Quadratic Regulator)

LQR controller

- Linear model：

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k$$

- Cost function:

$$J = \sum_{k=1}^{N} \Big(\underbrace{\boldsymbol{x}_k^T \boldsymbol{Q} \boldsymbol{x}_k}_{\text{(error) state cost}} + \underbrace{\boldsymbol{u}_k^T \boldsymbol{R} \boldsymbol{u}_k}_{\text{control cost}} \Big)$$

Positive semi-definite

$$Q = Q^T \succ 0$$
$$R = R^T \succ 0$$

- Goal: find the best control sequence $u_{1:N}^*$ that minimizes J
  - Solve the *Riccati* equation:
    - Set the initial values $P_N = Q$, and perform the iterative loop from the back to the front: $k = N, \ldots, 1$.
      - $\boldsymbol{P}_{k-1} = \boldsymbol{A}^T \boldsymbol{P}_k \boldsymbol{A} + \boldsymbol{Q} - \boldsymbol{A}^T \boldsymbol{P}_k \boldsymbol{B} \left(\boldsymbol{B}^T \boldsymbol{P}_k \boldsymbol{B} + \boldsymbol{R} \right)^{-1} \boldsymbol{B}^T \boldsymbol{P}_k \boldsymbol{A}$
      - $\boldsymbol{K}_k = \left(\boldsymbol{B}^T \boldsymbol{P}_k \boldsymbol{B} + \boldsymbol{R} \right)^{-1} \boldsymbol{B}^T \boldsymbol{P}_k \boldsymbol{A}$
      - $u_k^* = -\boldsymbol{K}_k \boldsymbol{x}_k$

LQR controller

Ideal LQR problem

- Linear discrete model:

$$x_{k+1} = Ax_k + Bu_k$$

- Cost function:
  - State is error

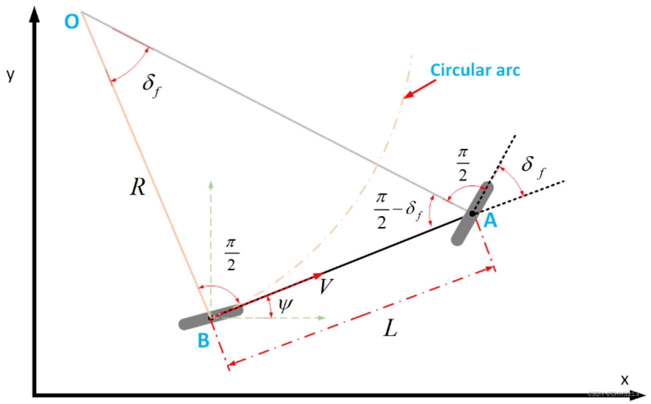
$$J = \sum_{k=1}^{N} \left(x_k^T Q x_k + u_k^T R u_k \right)$$

Vehicle Kinematic Model

$$\mathbf{x} = [x, y, \psi]^T \qquad \begin{cases} \dot{x} = v\cos(\psi) \\ \dot{y} = v\sin(\psi) \\ \dot{\psi} = \frac{v}{L}\tan\delta_f \end{cases}$$

$$\mathbf{u} = [v, \delta]^T$$

- Nonlinear
- Continuous system
- State is not error



LQR controller

Linearization

State: $\boldsymbol{\chi} = [x, y, \psi]^T$

Control input : $\mathbf{u} = [v, \delta]^T$

Model:

$$\begin{cases} \dot{x} = v\cos(\psi) = f_1 \\ \dot{y} = v\sin(\psi) = f_2 \\ \dot{\psi} = \frac{v}{L}\tan\delta_f = f_3 \end{cases}$$

$$\frac{\partial f(\boldsymbol{\chi},\mathbf{u})}{\partial \boldsymbol{\chi}} = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial \psi} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial \psi} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial \psi} \end{bmatrix} = \begin{bmatrix} 0 & 0 & -v_r\sin\varphi_r \\ 0 & 0 & v_r\cos\varphi_r \\ 0 & 0 & 0 \end{bmatrix}$$

$$\frac{\partial f(\boldsymbol{\chi},\mathbf{u})}{\partial \mathbf{u}} = \begin{bmatrix} \frac{\partial f_1}{\partial v} & \frac{\partial f_1}{\partial \delta} \\ \frac{\partial f_2}{\partial v} & \frac{\partial f_2}{\partial \delta} \\ \frac{\partial f_3}{\partial v} & \frac{\partial f_3}{\partial \delta} \end{bmatrix} = \begin{bmatrix} \cos\psi_r & 0 \\ \sin\psi_r & 0 \\ \frac{\tan\delta_r}{L} & \frac{v_r}{L\cos^2\delta_r} \end{bmatrix}$$

Linearize at the points on the reference trajectory:

- Reference point: $\boldsymbol{\chi}_r = [x_r, y_r, \psi_r]^T \quad \mathbf{u}_r = [v_r, \delta_r]^T$ e.g. $\begin{bmatrix} v_r = 2\text{m/s} \\ \delta_r = 0 \end{bmatrix}$

- Use the first-order Taylor series expansion

$$\dot{\boldsymbol{\chi}} = f(\boldsymbol{\chi}_r, \mathbf{u}_r) + \frac{\partial f(\boldsymbol{\chi},\mathbf{u})}{\partial \boldsymbol{\chi}}(\boldsymbol{\chi} - \boldsymbol{\chi}_r) + \frac{\partial f(\boldsymbol{\chi},\mathbf{u})}{\partial \mathbf{u}}(\mathbf{u} - \mathbf{u}_r)$$

$$\dot{\boldsymbol{\chi}} - f(\boldsymbol{\chi}_r, \mathbf{u}_r) = \frac{\partial f(\boldsymbol{\chi},\mathbf{u})}{\partial \boldsymbol{\chi}}(\boldsymbol{\chi} - \boldsymbol{\chi}_r) + \frac{\partial f(\boldsymbol{\chi},\mathbf{u})}{\partial \mathbf{u}}(\mathbf{u} - \mathbf{u}_r)$$

LQR controller

State -> Error State

$$\underbrace{\dot{\boldsymbol{\chi}} - f(\boldsymbol{\chi}_r, \mathbf{u}_r)} = \frac{\partial f(\boldsymbol{\chi}, \mathbf{u})}{\partial \boldsymbol{\chi}} (\boldsymbol{\chi} - \boldsymbol{\chi}_r) + \frac{\partial f(\boldsymbol{\chi}, \mathbf{u})}{\partial \mathbf{u}} (\mathbf{u} - \mathbf{u}_r)$$

$$\begin{bmatrix} \dot{x} - \dot{x}_r \\ \dot{y} - \dot{y}_r \\ \dot{\psi} - \dot{\psi}_r \end{bmatrix} = \begin{bmatrix} 0 & 0 & -v_r \sin\psi_r \\ 0 & 0 & v_r \cos\psi_r \\ 0 & 0 & 0 \end{bmatrix} \boxed{\begin{bmatrix} x - x_r \\ y - y_r \\ \psi - \psi_r \end{bmatrix}} + \begin{bmatrix} \cos\psi_r & 0 \\ \sin\psi_r & 0 \\ \frac{\tan\delta_r}{L} & \frac{v_r}{L\cos^2\delta_r} \end{bmatrix} \boxed{\begin{bmatrix} v - v_r \\ \delta - \delta_r \end{bmatrix}}$$

Trajectory error $\tilde{\chi}$

Control error $\tilde{u}$

$$\dot{\tilde{\chi}} = A\tilde{\chi} + B\tilde{\mathbf{u}}$$

$$\begin{cases} \dot{x} = v\cos(\psi) \\ \dot{y} = v\sin(\psi) \\ \dot{\psi} = \frac{v}{L}\tan\delta_f \end{cases} \Longrightarrow \dot{\tilde{\chi}} = A\tilde{\chi} + B\tilde{\mathbf{u}}$$

LQR controller

Discretization

$$
\begin{aligned}
\dot{\tilde{\chi}} &= \begin{bmatrix} \dot{x} - \dot{x}_r \\ \dot{y} - \dot{y}_r \\ \dot{\psi} - \dot{\psi}_r \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 & -v_r \sin\psi_r \\ 0 & 0 & v_r \cos\psi_r \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x - x_r \\ y - y_r \\ \psi - \psi_r \end{bmatrix} + \begin{bmatrix} \cos\psi_r & 0 \\ \sin\psi_r & 0 \\ \frac{\tan\delta_r}{L} & \frac{v_r}{L\cos^2\delta_r} \end{bmatrix} \begin{bmatrix} v - v_r \\ \delta - \delta_r \end{bmatrix} \\
&= A\tilde{\chi} + B\tilde{\mathbf{u}}
\end{aligned}
$$

- One step Euler method

$$
\begin{aligned}
\tilde{\chi}(k+1) &= (TA + I)\tilde{\chi}(k) + TB\tilde{\mathbf{u}}(k) \\
&= \begin{bmatrix} 1 & 0 & -Tv_r \sin\psi_r \\ 0 & 1 & Tv_r \cos\psi_r \\ 0 & 0 & 1 \end{bmatrix} \tilde{\chi}(k) + \begin{bmatrix} T\cos\psi_r & 0 \\ T\sin\psi_r & 0 \\ \frac{T\tan\delta_r}{L} & \frac{v_r T}{L\cos^2\delta_r} \end{bmatrix} \tilde{\mathbf{u}}(k) \\
&= \tilde{A}\tilde{\chi}(k) + \tilde{B}\tilde{\mathbf{u}}(k)
\end{aligned}
$$

LQR controller

System model:

$$\begin{aligned}\tilde{\chi}(k+1) &= (TA+I)\tilde{\chi}(k) + TB\tilde{\mathbf{u}}(k) \\ &= \begin{bmatrix} 1 & 0 & -Tv_r \sin\psi_r \\ 0 & 1 & Tv_r \cos\psi_r \\ 0 & 0 & 1 \end{bmatrix} \tilde{\chi}(k) + \begin{bmatrix} T\cos\psi_r & 0 \\ T\sin\psi_r & 0 \\ \frac{T\tan\delta_r}{L} & \frac{v_r T}{L\cos^2\delta_r} \end{bmatrix} \tilde{\mathbf{u}}(k) \\ &= \tilde{A}\tilde{\chi}(k) + \tilde{B}\tilde{\mathbf{u}}(k)\end{aligned}$$

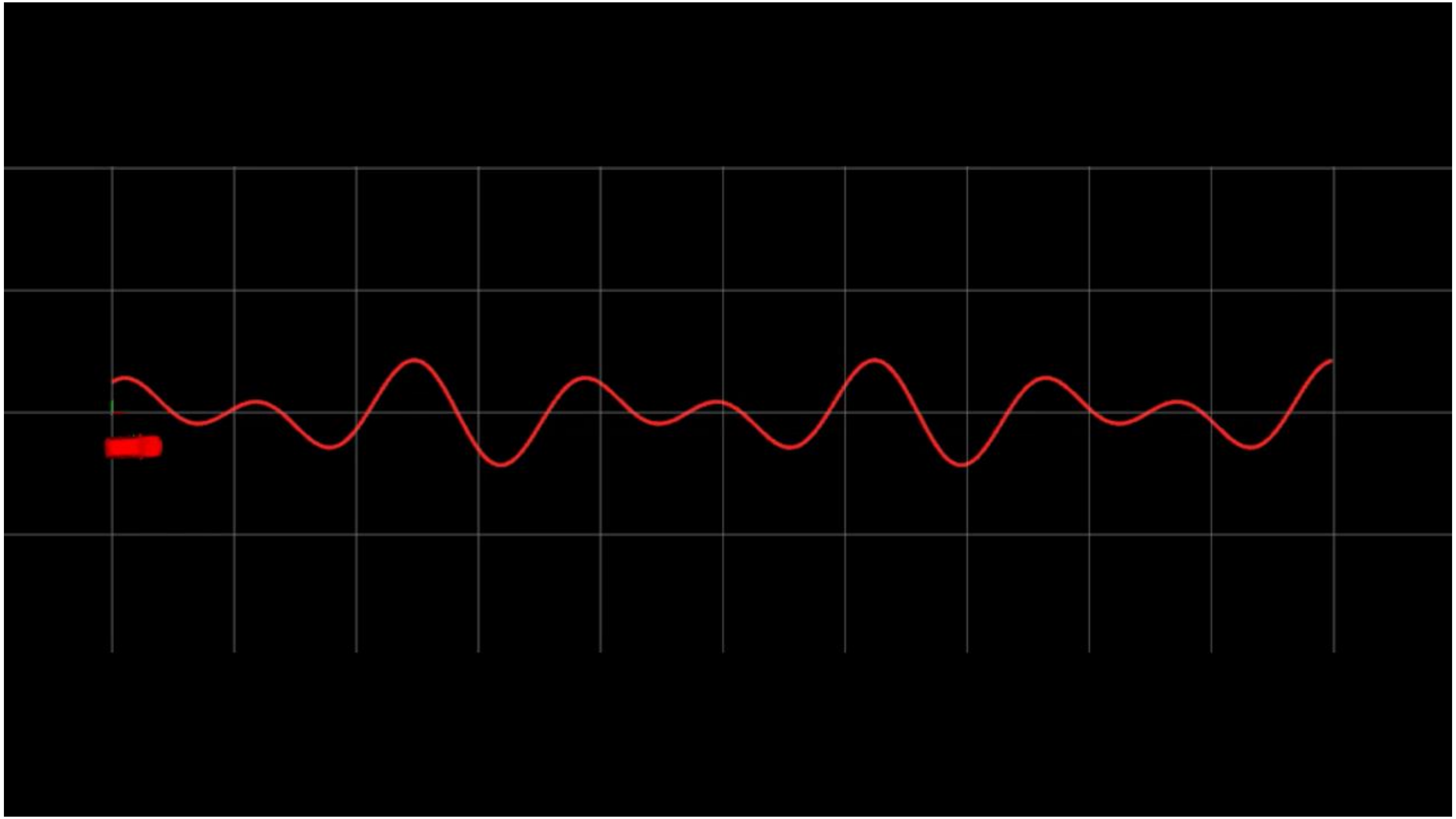
Quadratic cost function:

$$\mathrm{J} = \sum_{\mathrm{k}=1}^{\mathrm{N}} (\tilde{\mathbf{X}}^{\mathrm{T}}\mathrm{Q}\tilde{\mathbf{X}} + \tilde{\mathbf{u}}^{\mathrm{T}}\mathrm{R}\tilde{\mathbf{u}})$$

Solve the *Riccati* equation:

- Set the initial values $P_N = Q$, and perform the iterative loop from the back to the front: $k = N, \dots, 1$.
  - $\boldsymbol{P_{k-1}} = \boldsymbol{A^T P_k A} + \boldsymbol{Q} - \boldsymbol{A^T P_k B}\left(\boldsymbol{B^T P_k B} + \boldsymbol{R}\right)^{-1}\boldsymbol{B^T P_k A}$
  - $\boldsymbol{K}_k = \left(\boldsymbol{B^T P_k B} + \boldsymbol{R}\right)^{-1}\boldsymbol{B^T P_k A}$
  - $\tilde{\boldsymbol{u}}_k^* = -\boldsymbol{K}_k \tilde{x}_k$
  - $\boldsymbol{u}^* = \tilde{\boldsymbol{u}}_k^* + \boldsymbol{u}_r$

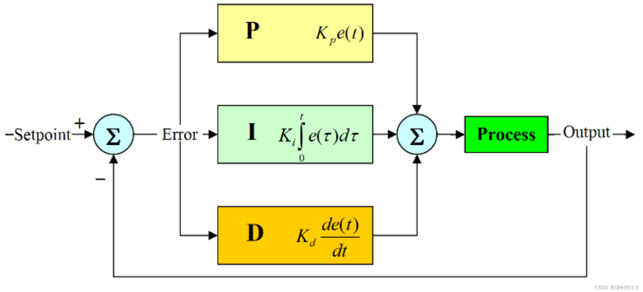
LQR controller



Control Method Comparison (Lateral Tracking)

PID

- Lateral error



$$u_k = K_P e_k + K_I \sum_{j=0}^{k} e_j + K_D (e_k - e_{k-1})$$

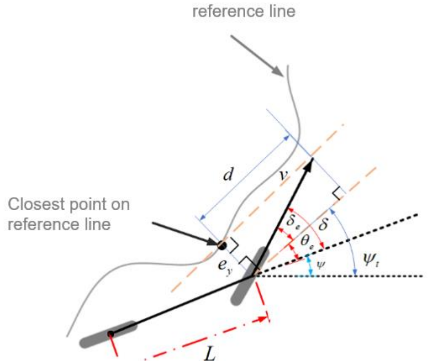
Stanley

- Lateral error
- Heading error

$$\delta_e = \arctan \frac{e_y}{d} = \arcsin \frac{k e_y}{v}$$

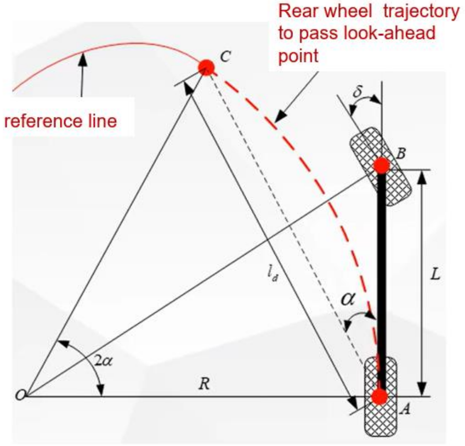
$$\theta_e = \psi_t - \psi$$

$$\delta = \theta_e + \delta_e$$



Pure Pursuit

- Look-ahead point



$$\delta(t) = tan^{-1}\left(\frac{2L sin(\alpha(t))}{\ell_d}\right)$$

LQR

- Model-based Optimal control

$$\begin{aligned}
\tilde{\chi}(k+1) &= (TA + I)\tilde{\chi}(k) + TB\tilde{\mathbf{u}}(k) \\
&= \begin{bmatrix} 1 & 0 & -Tv_r \sin\psi_r \\ 0 & 1 & Tv_r \cos\psi_r \\ 0 & 0 & 1 \end{bmatrix} \tilde{\chi}(k) + \begin{bmatrix} T\cos\psi_r & 0 \\ T\sin\psi_r & 0 \\ \frac{T\tan\delta_r}{L} & \frac{v_r T}{L\cos^2\delta_r} \end{bmatrix} \tilde{\mathbf{u}}(k) \\
&= \tilde{A}\tilde{\chi}(k) + \tilde{B}\tilde{\mathbf{u}}(k)
\end{aligned}$$

$$\mathrm{J} = \sum_{\mathrm{k}=1}^{\mathrm{N}} (\tilde{\mathbf{X}}^{\mathrm{T}} \mathrm{Q} \tilde{\mathbf{X}} + \tilde{\mathbf{u}}^{\mathrm{T}} \mathrm{R} \tilde{\mathbf{u}})$$

Content

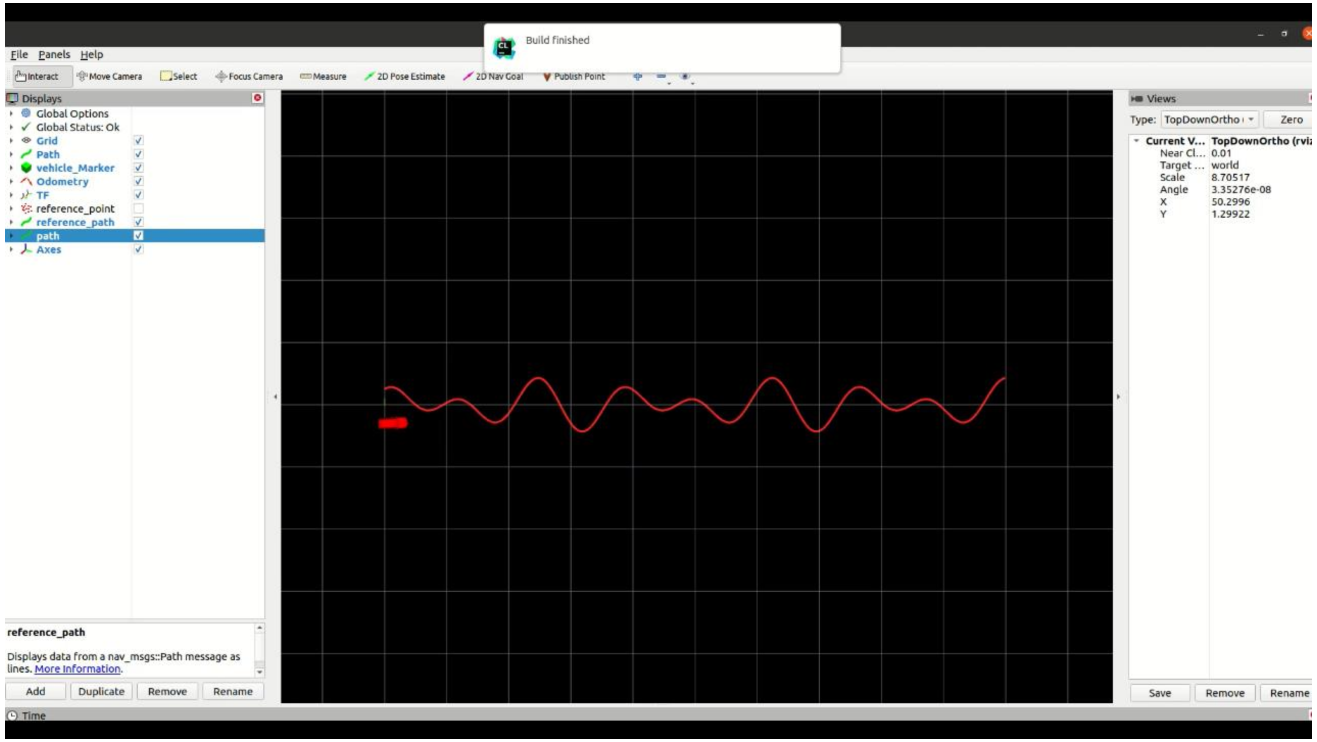
Assignment

Lateral Control in ROS

- Complete different functions in *controller.cpp*
  - PIDController() (example)
    - Ture the gain to make it better
  - StanlyController()
  - PurePursuitController()
  - LQRController()

- catkin_make your workspace
  - rosrun avp_control avp_control

Visualization in RVIZ



Compare tracking performance from different method.

avp_control_node.cpp - main()

```
double lateral_control;
// lateral_control = controller.PIDController(robot_state, refer_path, vehicle_model);
// lateral_control = controller.StanlyController(robot_state, refer_path, vehicle_model);
// lateral_control = controller.PurePursuitController(robot_state, refer_path, vehicle_model);
lateral_control = controller.LQRController(robot_state, refer_path, ugv: vehicle_model);
```

感谢聆听！

Thanks for Listening!