Lecture 6: Vehicle Modeling

Professor Katie Driggs-Campbell

February 25, 2021

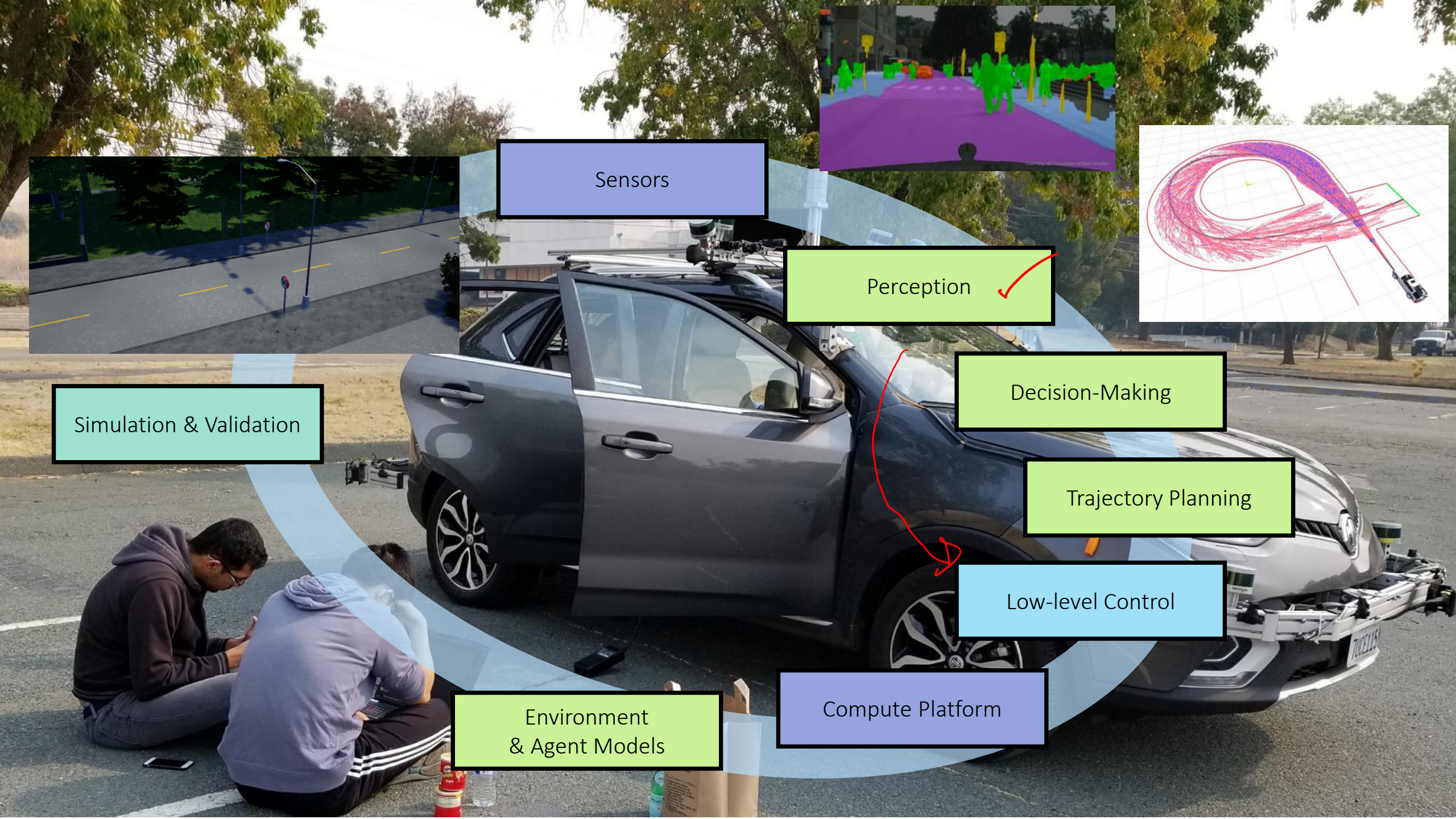
ECE484: Principles of Safe Autonomy

Administrivia

- Late policy has been posted → four “grace” days
- Oral exam update
  - Monday 4/12 to Tuesday 5/4
  - 15-minute time slots on Mondays 5-6pm, Tuesdays 5-6pm, Wednesdays 11am-12pm, or by appointment (if needed)
  - Questions and rubric released by 4/5
  - Decreasing bonus points, e.g.: (9-session_num)/10*bonus_points ← tentative

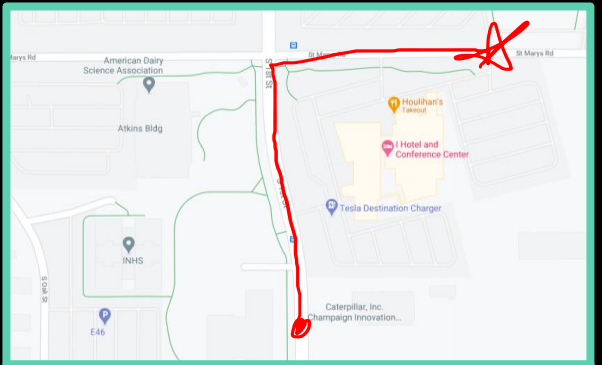
Today’s Plan

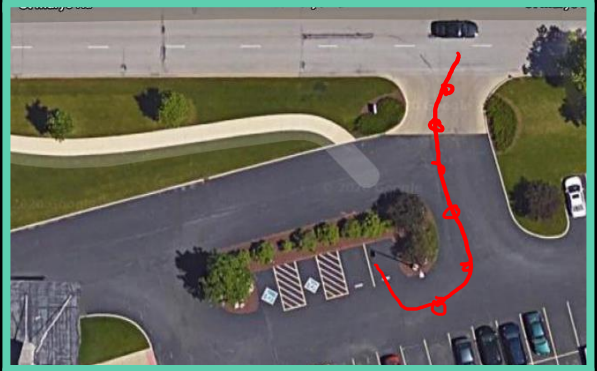
- What’s a model?
- Planning and Control Motivation
  - Open-loop control
- Vehicle Models
  - How to design your model
  - Dubin’s Car
  - Advanced Models: bicycle, tire dynamics

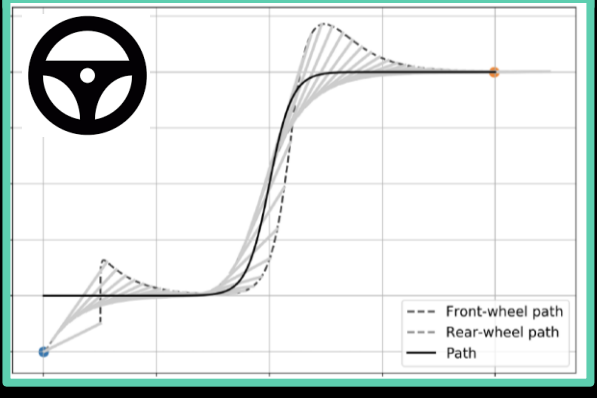

Sensors
Perception
Decision-Making
Trajectory Planning
Low-level Control
Compute Platform
Environment
& Agent Models
Simulation & Validation

Typical planning and control modules

- Global navigation and planner
  - Find paths from source to destination with static obstacles
  - Algorithms: Graph search, Dijkstra, Sampling-based planning
  - Time scale: Minutes
  - Look ahead: Destination
  - Output: reference center line, semantic commands
- Local planner
  - Dynamically feasible trajectory generation
  - Dynamic planning w.r.t. obstacles
  - Time scales: 10 Hz
  - Look ahead: Seconds
  - Output: Waypoints, high-level actions, directions / velocities
- Controller
  - Waypoint follower using steering, throttle
  - Algorithms: PID control, MPC, Lyapunov-based controller
  - Lateral/longitudinal control
  - Time scale: 100 Hz
  - Look ahead: current state
  - Output: low-level control actions







What is control?

- A means of regulating or limiting something
- Algorithms (or process) for manipulating a system to achieve to desired value

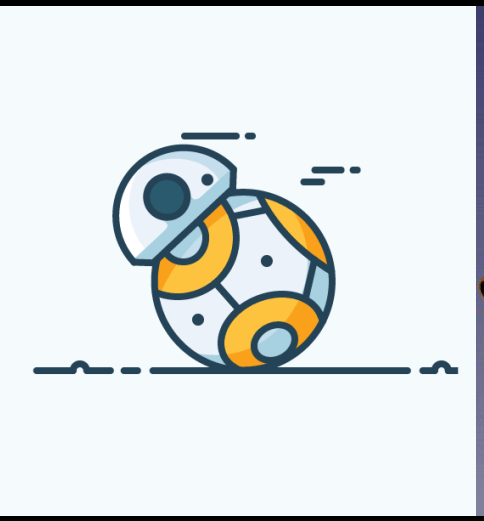


Image Credit: Justas Galaburda and Vincent Mokuenko

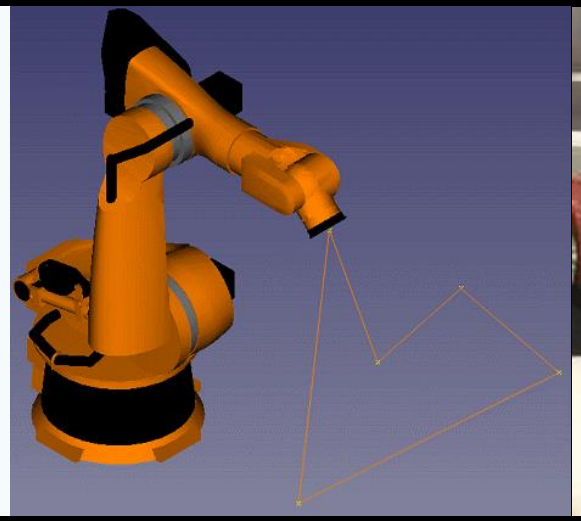


Image Credit: FreeCAD

Open-Loop Example

temp sensor → controller → heater

check temp every 30 min

on/off

if temp ≤ 70 then heat on for T min

if temp ≥ 75 then heat off for T min

Closed-Loop Example

$v \rightarrow$ want to stay v_d

Goals

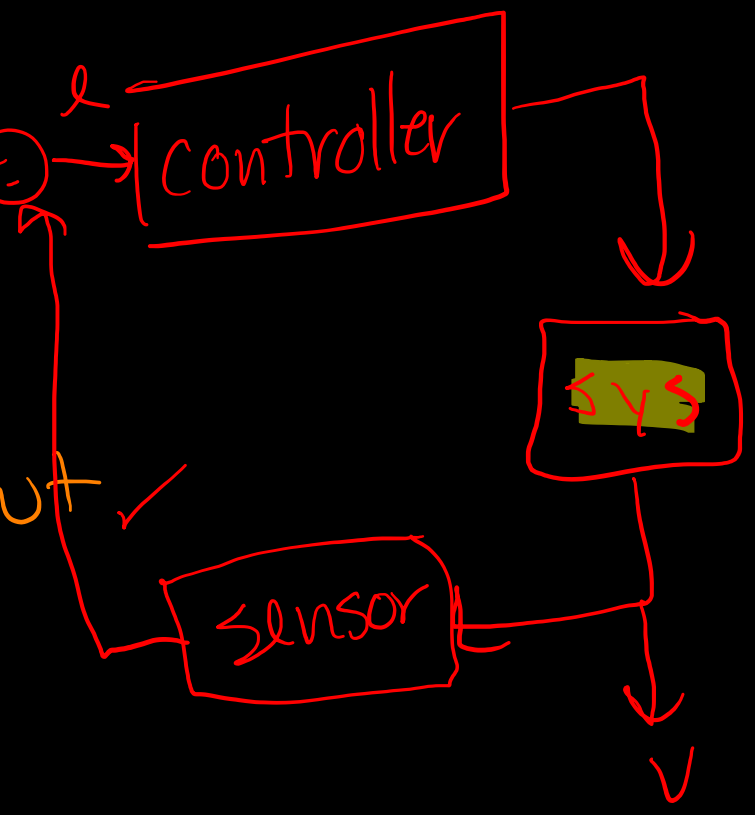
if $v > v_d$, want ↓ input ✓

if $v < v_d$, want ↑ input ✓

if $v \approx v_d$, want a small or no input ✓

look at errors instead!

$e = v_d - v$



Lecture 2: Modeling the scenario

- What is a model of a system?
- A *mathematical model* describes how a system behaves.
  - What are the key parameters and states?
  - How are the parameters selected by nature?
  - What are the initial conditions of the state?
  - How do the state change over time?
  - What parts of the model are available for observation/analysis?
- Models include the implicit and explicit assumptions (biases) we are making about the system

Breakout Room Discussion

1. Pick an AV scenario and try to think of the desired behaviors, the requirements of the controller, and the actions / inputs to the system
2. How would you define your model for your scenario?

   Hint: think of the model and simple controller in MP0
3. Other than control, what are some other use cases of models?

Breakout Room Discussion Notes

1. Pick an AV scenario and try to think of the desired behaviors, the requirements of the controller, and the actions / inputs to the system
2. How would you define your model for your scenario?
3. What are some other use cases of models?

Drones

→ input: ~~fans~~ propellers

Cars

ACC → maintain distance

regulate error

MPC

→ improved control design

Drive-thru Bot

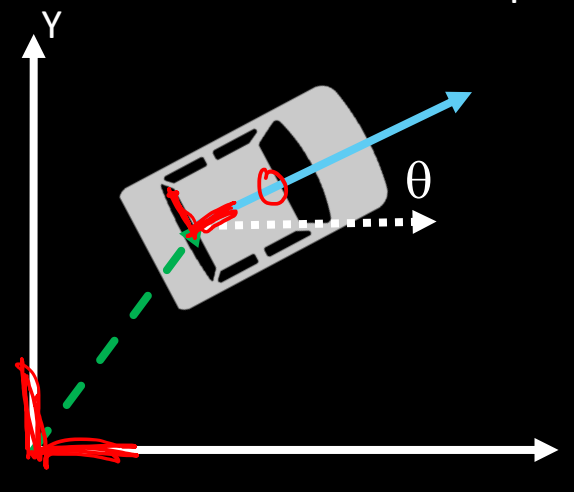
→ many tasks!

Following a curve

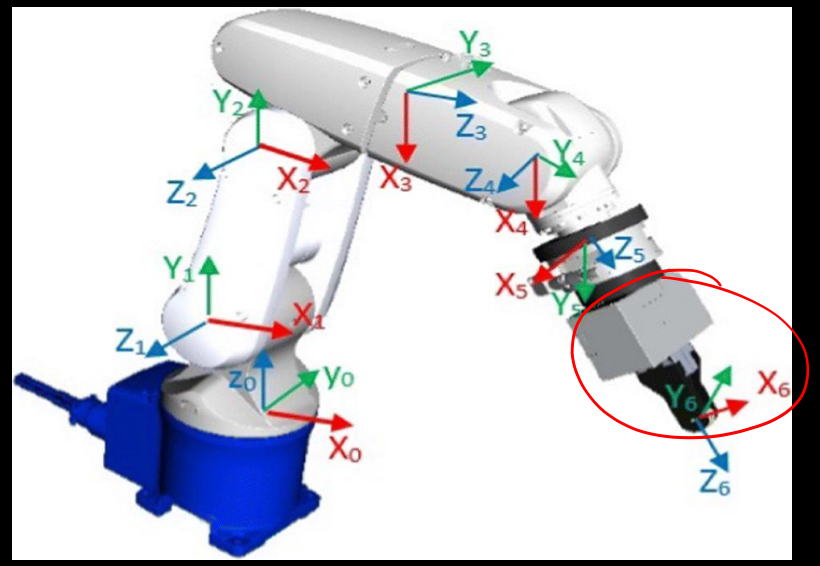
→ steering + acc

Coordinate Systems and Configurations

Car pose = $\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$



End effector pose = $\begin{bmatrix} x \\ y \\ z \\ \theta_x \\ \theta_y \\ \theta_z \end{bmatrix}$



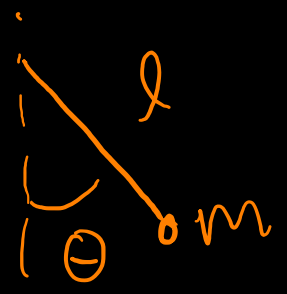
Dynamical Systems Model

Describe behavior in terms of instantaneous laws:

$$\frac{dx(t)}{dt} = \dot{x}(t) = f\big(x(t), u(t)\big)$$

$$x[t+1] = f(x[t], u[t])$$

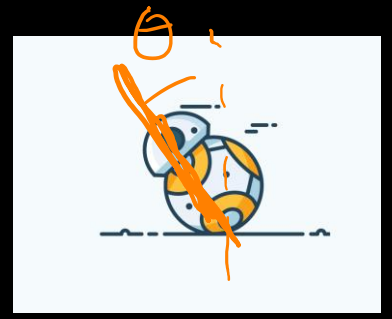
where $t \in \mathbb{R}, x(t) \in \mathbb{R}^n, u(t) \in \mathbb{R}^m$, and $f: \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ gives the dynamics / transition function



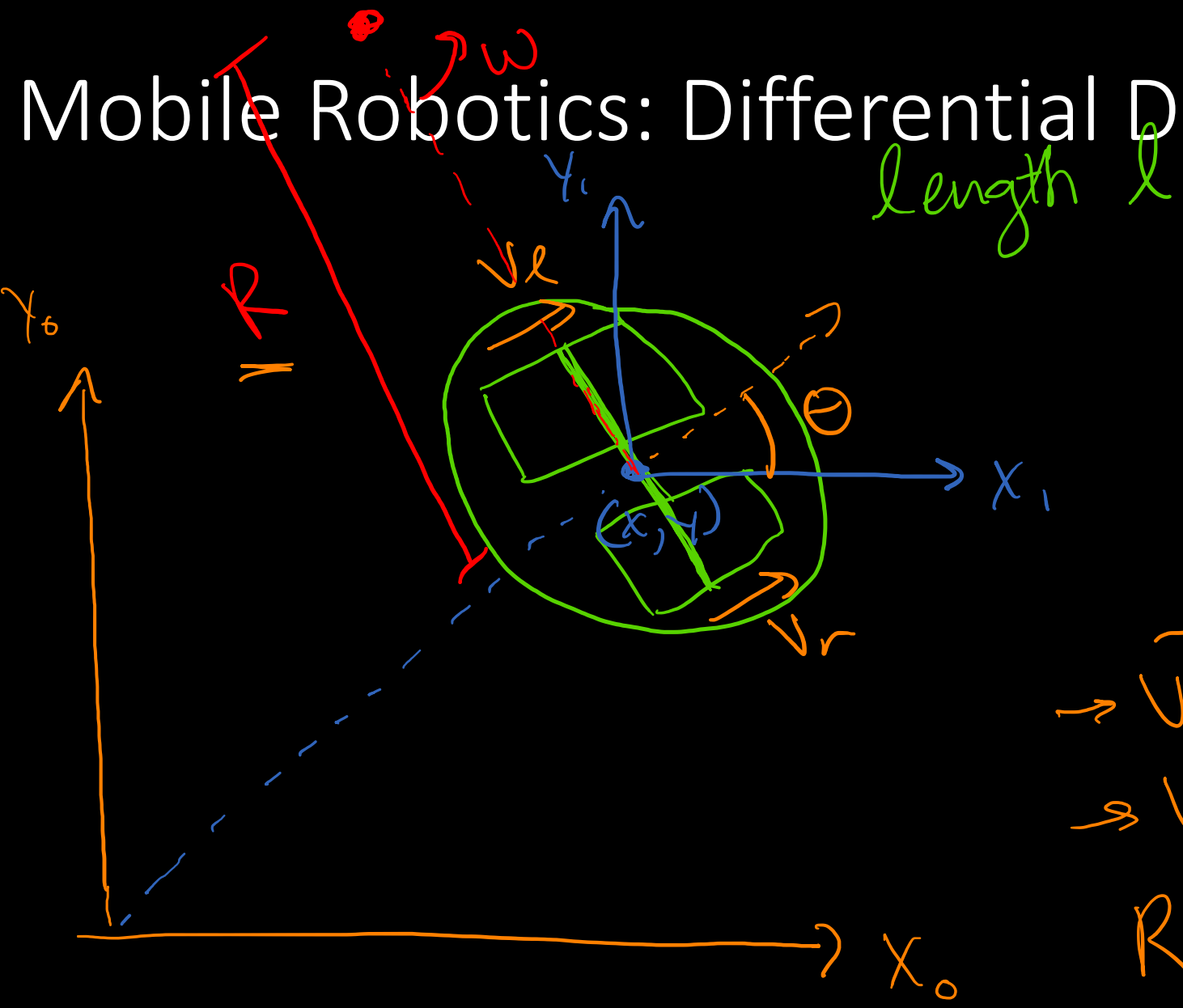
$$x_1 = \theta$$

$$x_2 = \dot{\theta} = \dot{x}_1$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ \cdots \end{bmatrix}$$



Mobile Robotics: Differential Drive



Instantaneous Center of Curvature:

$$\begin{bmatrix} ICC_x \\ ICC_y \end{bmatrix} = \begin{bmatrix} x - R\sin\theta \\ y - R\cos\theta \end{bmatrix}$$

$$\rightarrow V_r = \omega(R + l/2)$$

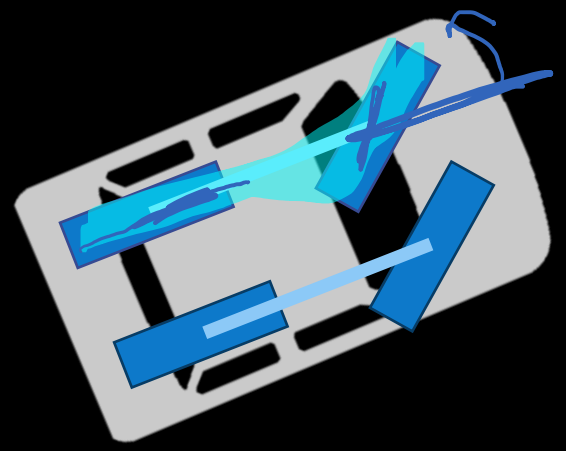
$$\rightarrow V_l = \omega(R - l/2)$$

$$R = l/2 \frac{(V_r + V_l)}{(V_r - V_l)}$$

$$\omega = \frac{V_r - V_l}{l}$$

Simple vehicle model: Dubin's car

- Key assumptions
  - Front and rear wheel in the plane in a stationary coordinate system
  - Steering input, front wheel steering angle δ
  - No slip: wheels move only in the direction of the plane they reside in
- Zeroing out the accelerations perpendicular to the plane in which the wheels reside, we can derive simple equations



Reference: Paden, Brian, Michal Cap, Sze Zheng Yong, Dmitry S. Yershov, and Emilio Frazzoli. 2016. A survey of motion planning and control techniques for self-driving urban vehicles. IEEE Transactions on Intelligent Vehicles 1 (1): 33–55.

Dubin's Model

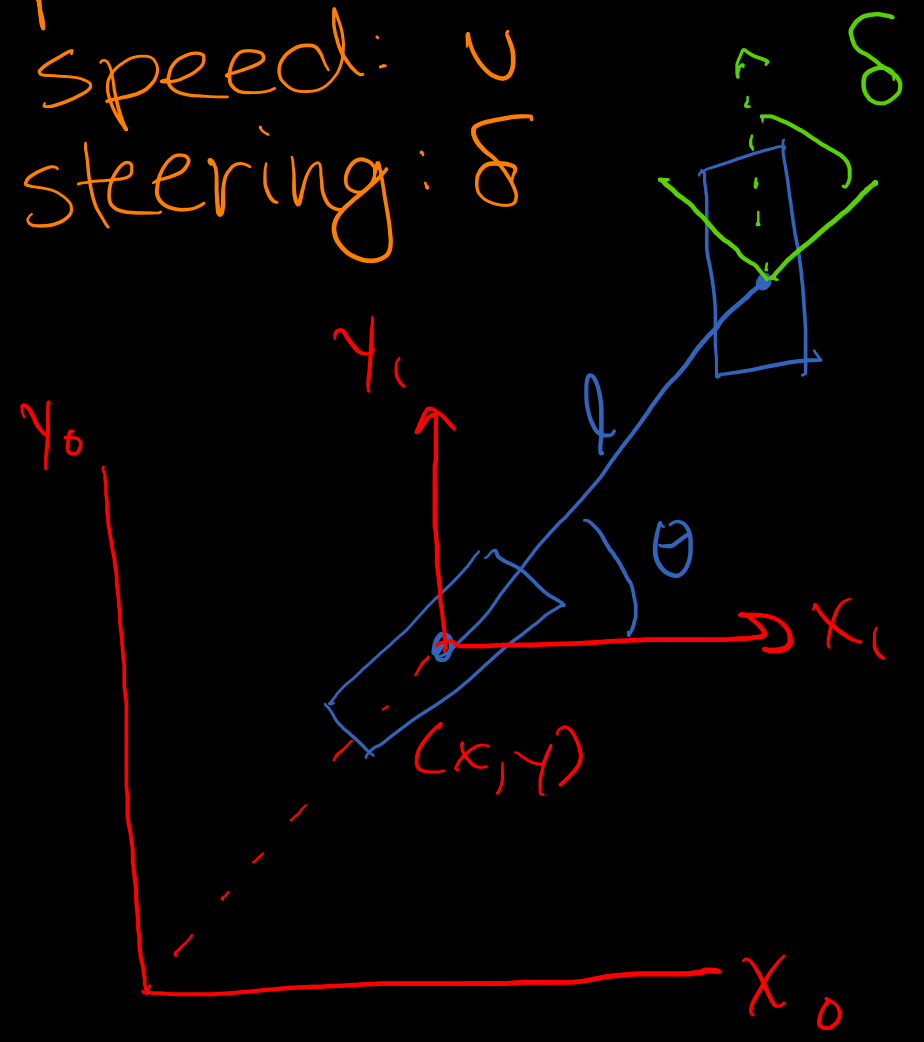
$$\dot{x} = v\cos\theta$$
$$\dot{y} = v\sin\theta$$
$$\dot{\theta} = \frac{v}{l}\tan\delta$$

length l

pose: $[x, y, \theta]^T$

speed: v

steering: δ



Many more advanced models…

[Kinematic] Bicycle Model

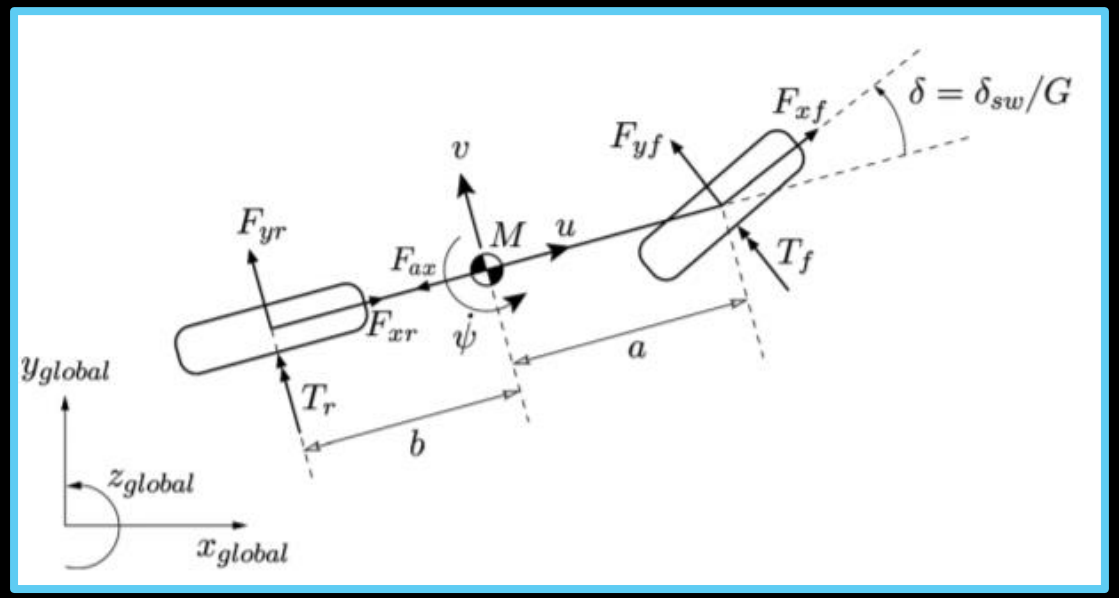
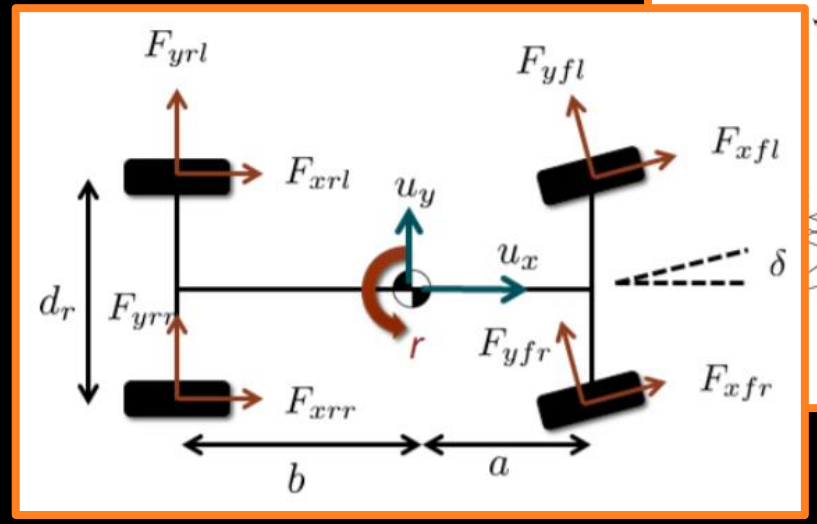


Image Credit and Reference: J.P. Timings and D.J. Cole. "Minimum maneuver time calculation using convex optimization." Journal of Dynamic Systems, Measurement, and Control 135.3 (2013).

[Dynamic] Tire Models



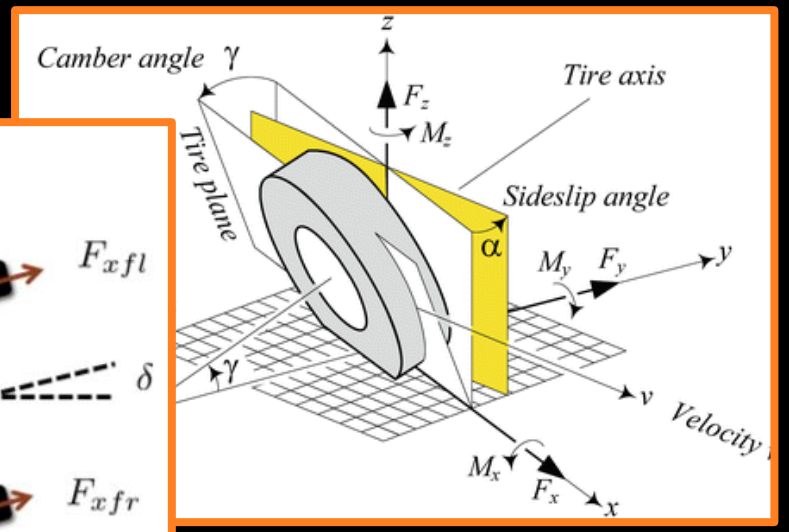
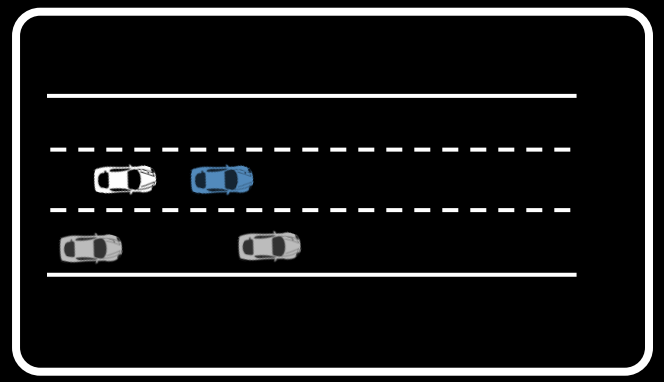


Image Credit: Springer

Image Credit and Reference: J.K. Subosits and J.C. Gerdes. "Impacts of Model Fidelity on Trajectory Optimization for Autonomous Vehicles in Extreme Maneuvers." IEEE Transactions on Intelligent Vehicles, 2021.

Dynamical system models



- Nonlinear dynamics

  Generally, nonlinear ODEs do not have closed form solutions!

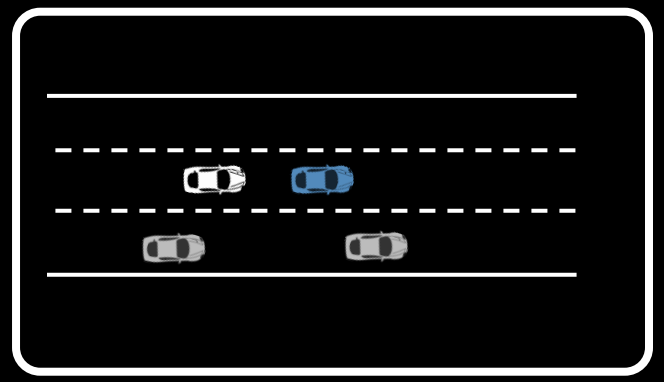
Dubin's car model

$\dot{v} = a$	Speed
$\frac{ds_x}{dt} = v\cos(\psi)$	Horizontal position
$\frac{ds_y}{dt} = v\sin(\psi)$	Vertical position
$\frac{d\delta}{dt} = v_\delta$	Steering angle
$\frac{d\psi}{dt} = \frac{v}{l}\tan(\delta)$	Heading angle

Physical plant

$\frac{dx}{dt} = f(x,u)$	System dynamics
$x[t+1] = f(x[t],u[t])$	
$x = [v, s_x, s_y, \delta, \psi]$	State variables
$u = [a, v_\delta]$	Control inputs

Nonlinear *hybrid* dynamics



Physical plant

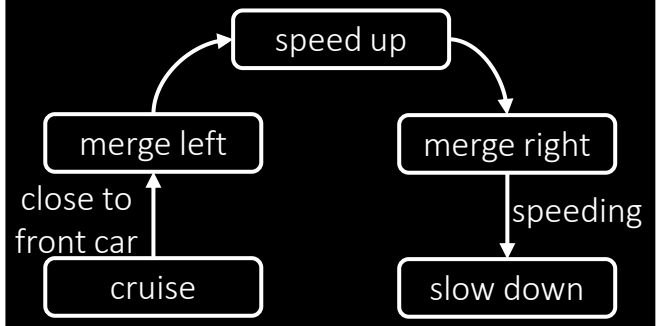
$\frac{dx}{dt} = f(x, u)$ System dynamics

$x[t+1] = f(x[t], u[t])$

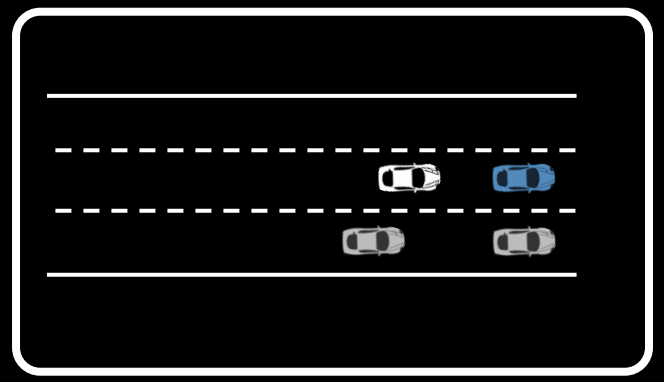
$x = [v, s_x, s_y, \delta, \psi]$ State variables

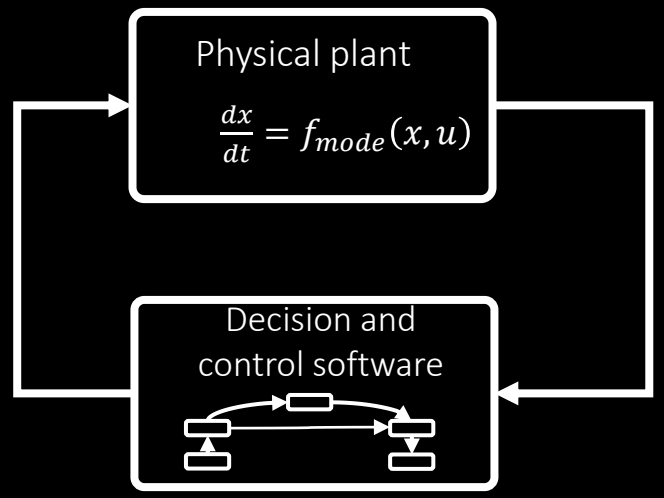
$u = [a, v_\delta]$ Control inputs

Decision and control software



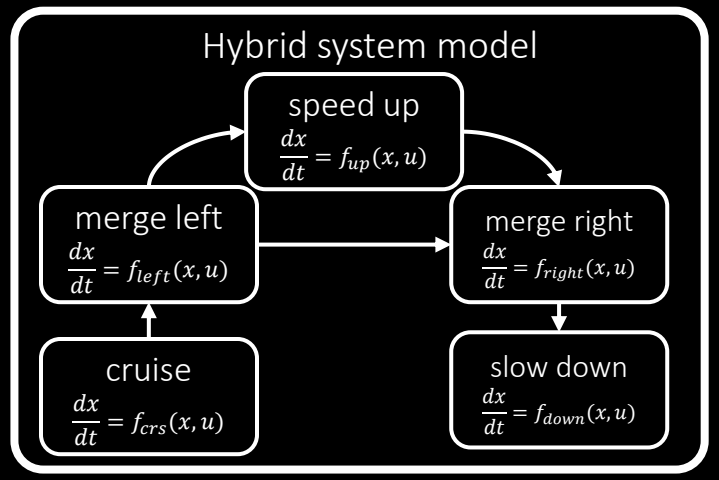
Typical system models





- Nonlinear *hybrid* dynamics

  Interaction between computation and physics can lead to unexpected behaviors



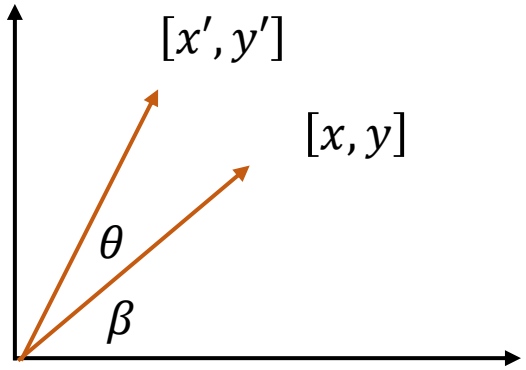
Summary

- Dynamical systems models allow us to reason about low-level behaviors of systems and determine what is and is not feasible
  - Typically required to design controllers!
- Discussed a few types of models from simple to complex
- *Next time:* Look at simple PID control design for trajectory following

Extra Slides

An aside: Coordinate transformations Rotation matrix

The following matrix rotates a vector $[x, y]$ counter-clockwise by an angle of θ



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

That is:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Derivation

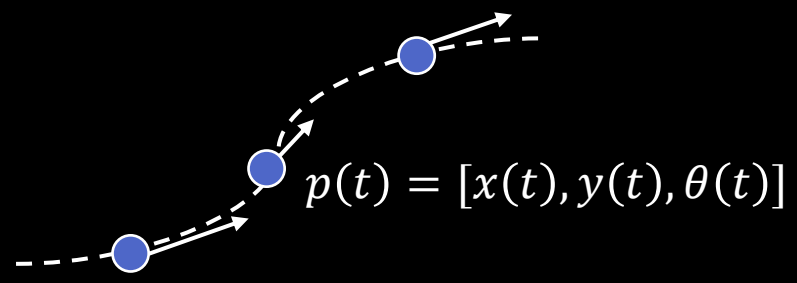
$x' = rcos(\beta + \theta) = r(cos\beta cos\theta - sin\theta \sin\beta)$

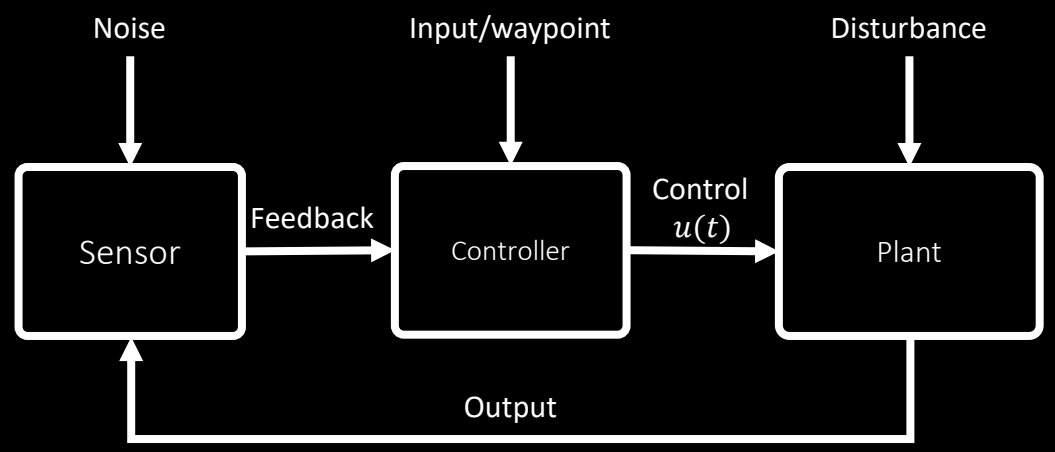
$= r\cos\beta cos\theta - r\sin\theta\sin\beta$

$= xcos\theta - y\sin\theta$

Path following control

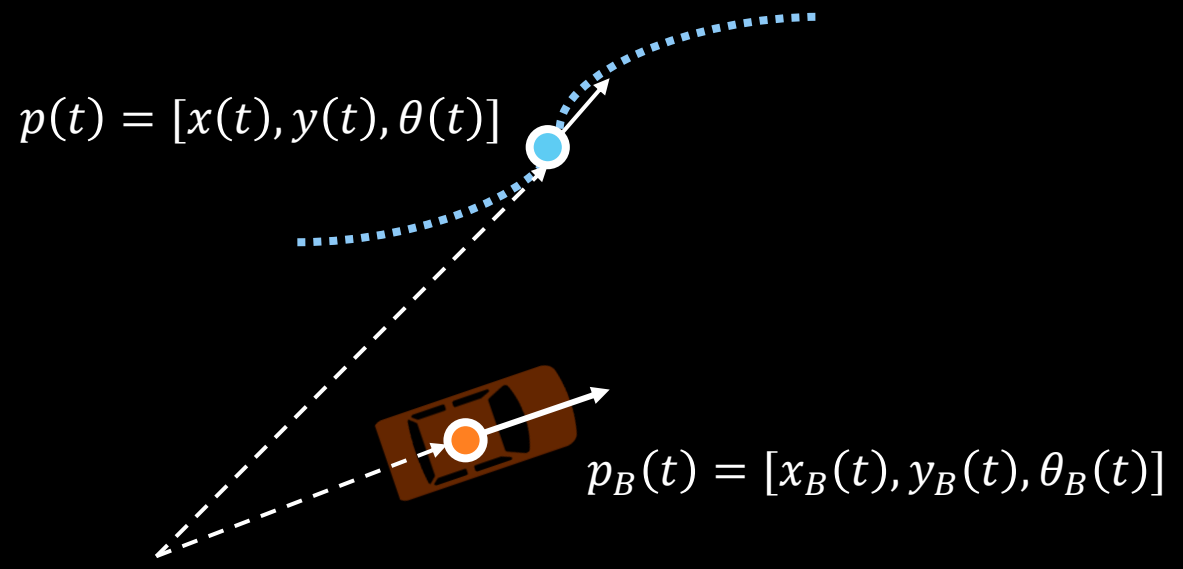
- The path followed by a robot can be represented by a *trajectory or path* parameterized by time
  - → from a higher-level planner
- Defines the desired instantaneous pose $p(t)$





Path following control

- Desired instantaneous pose $p(t)$
- How to define error between actual pose $p_B(t)$ and desired pose $p(t)$ in the form of $y_d(t) - y(t)$



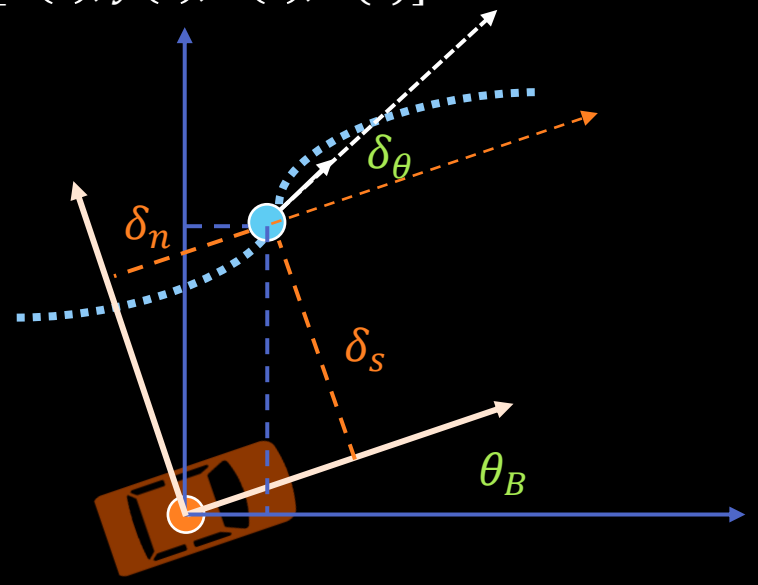
Path following control

The error vector measured vehicle coordinates

$$e(t) = [\delta_s(t), \delta_n(t), \delta_\theta(t), \delta_v(t)]$$

$[\delta_s, \delta_n]$ define the coordinate errors in the vehicle's reference frame: along track error and cross track error

$p(t) = [x(t), y(t), \theta(t), v(t)]$



$p_B(t) = [x_B(t), y_B(t), \theta_B(t), v_B(t)]$

- Along track error: distance ahead or behind the target in the instantaneous direction of motion.

  $$\delta_s = \cos(\theta_B(t))\,(x(t) - x_B(t)) + sin(\theta_B(t))\,(y(t) - y_B(t))$$

- Cross track error: portion of the position error orthogonal to the intended direction of motion

  $$\delta_n = -\sin(\theta_B(t))\,(x(t) - x_B(t)) + cos(\theta_B(t))\,(y(t) - y_B(t))$$

- Heading error: difference between desired and actual orientation and direction

  $$\delta_\theta = \theta(t) - \theta_B(t)$$
  $$\delta_v = v(t) - v_B(t)$$

→ Each of these errors match the form $y_d(t) - y(t)$

Open-loop waypoint following

- We can write an open-loop controller for a robot that is naturally controlled via angular velocity, such as a differential-drive robot:

$$u_{\omega,OL}(t) = \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} = \begin{bmatrix} \sqrt{\dot{x}(t)^2 + \dot{y}(t)^2} \\ \dot{\theta}(t) \end{bmatrix}$$

- We can write an open-loop controller for a robot with car-like steering:

$$u_{\kappa,OL}(t) = \begin{bmatrix} v(t) \\ \kappa(t) \end{bmatrix} = \begin{bmatrix} \sqrt{\dot{x}(t)^2 + \dot{y}(t)^2} \\ \dfrac{\dot{\theta}(t)}{\sqrt{\dot{x}(t)^2 + \dot{y}(t)^2}} \end{bmatrix}$$