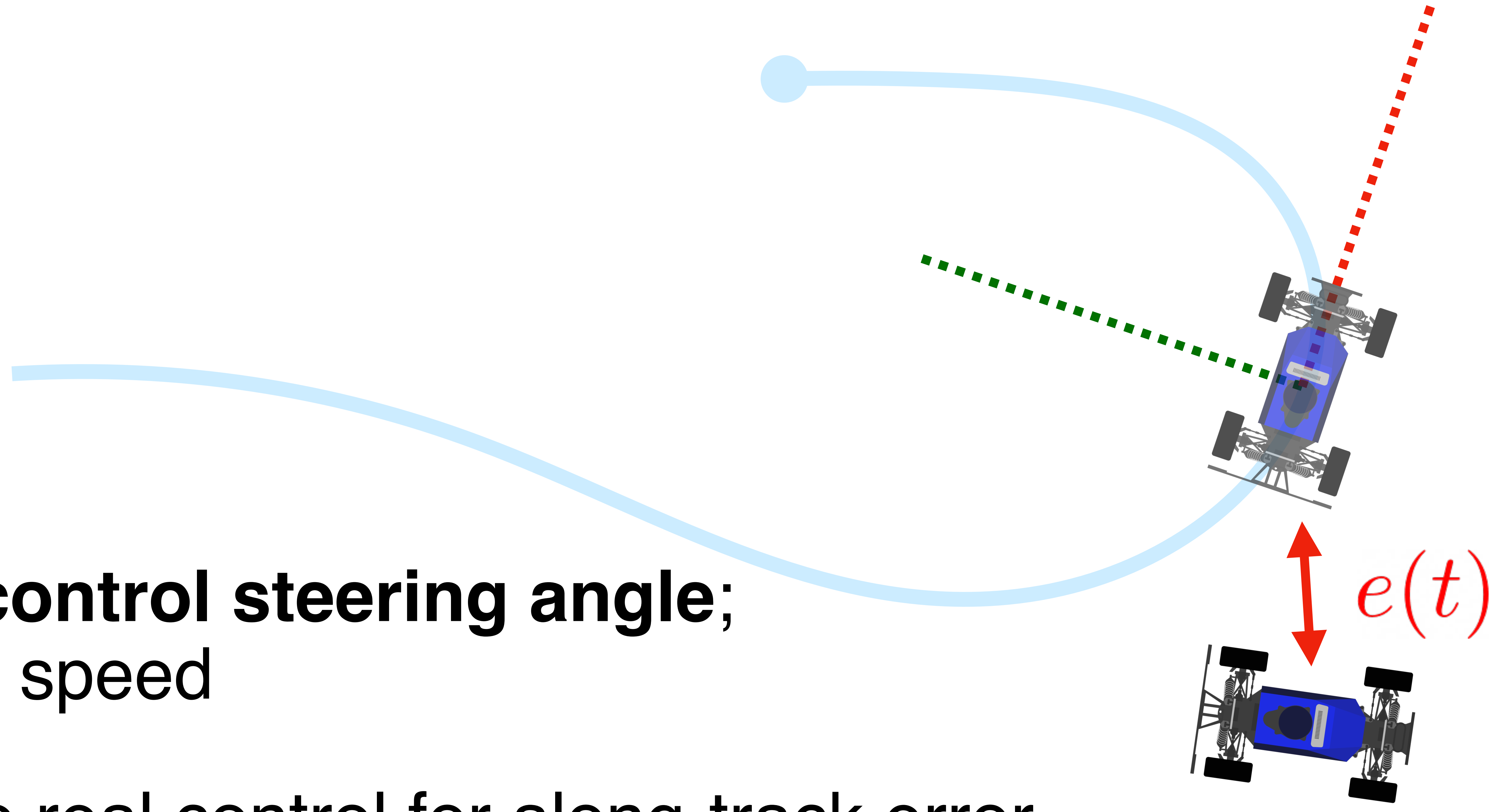
CSE 478 Robot Autonomy
Model Predictive Control
Pure Pursuit

Abhishek Gupta (abhgupta@cs)
Siddhartha Srinivasa (siddh@cs)

TAs:
Carolina Higuera (chiguera@cs)
Rishabh Jain (jrishabh@cs)
Entong Su (ensu@cs)

Step 4: Compute control law



- We will **only control steering angle**; fixed constant speed
- As a result, no real control for along-track error
- Some control laws will only minimize cross-track error, others will also minimize heading

$$u = K(e)$$

Step 4: Compute control law

Compute control action based on instantaneous error

$$u = K(\mathbf{x}, e)$$

control state error

(steering angle, speed)

Apply control action, robot moves a bit, compute new error, repeat

Different laws have different trade-offs

Kinematic Car Model

RECALL

$$\dot{x} = f(x, u)$$

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

X-COORDINATE

Y-COORDINATE

HEADING

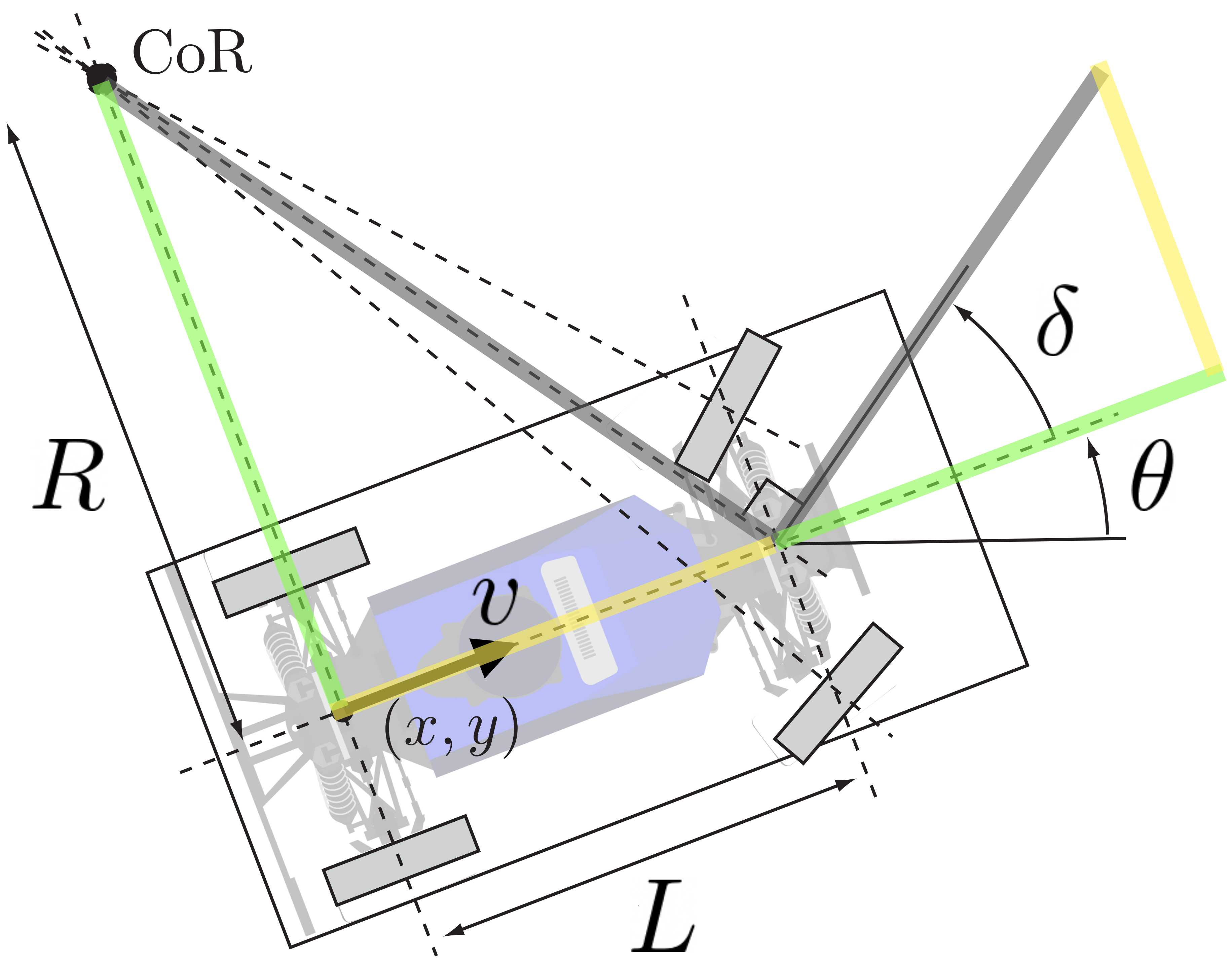
$$u = \begin{bmatrix} v \\ \delta \end{bmatrix}$$

SPEED

STEERING ANGLE

RECALL

Equations of Motion



$$\dot{x} = v\cos\theta$$

$$\dot{y} = v\sin\theta$$

$$\dot{\theta} = \omega = \frac{v}{R} = \frac{v\tan\delta}{L}$$

$$\tan\delta = \frac{L}{R} \rightarrow R = \frac{L}{\tan\delta}$$

Kinematic Car Model

RECALL

$$\dot{x} = f(x, u)$$

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

X-COORDINATE

Y-COORDINATE

HEADING

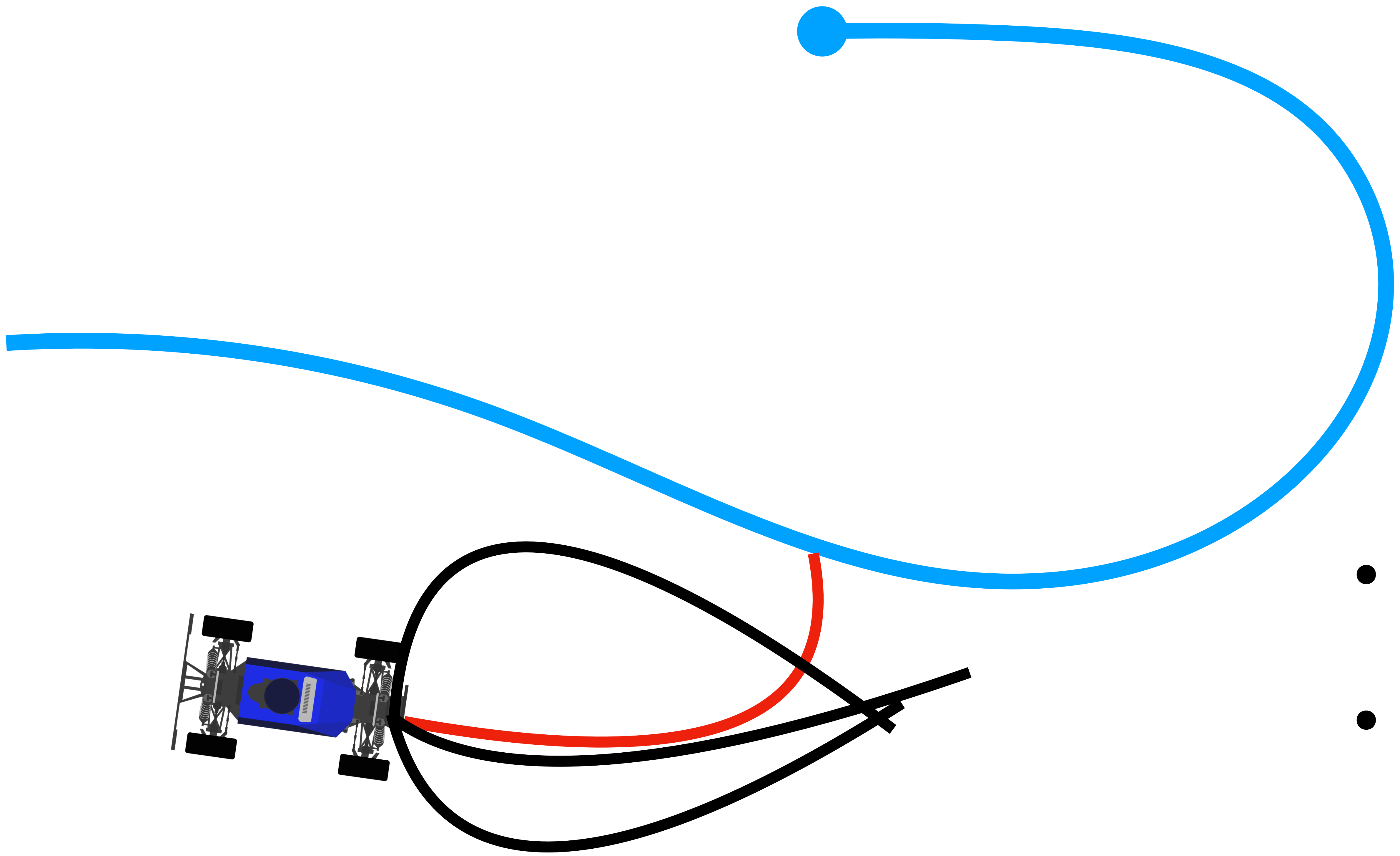
$$\begin{bmatrix} v \\ \delta \end{bmatrix}$$

SPEED

STEERING ANGLE

$$\begin{bmatrix} 1 \\ u \end{bmatrix}$$

Model Predictive Control / Receding Horizon Control

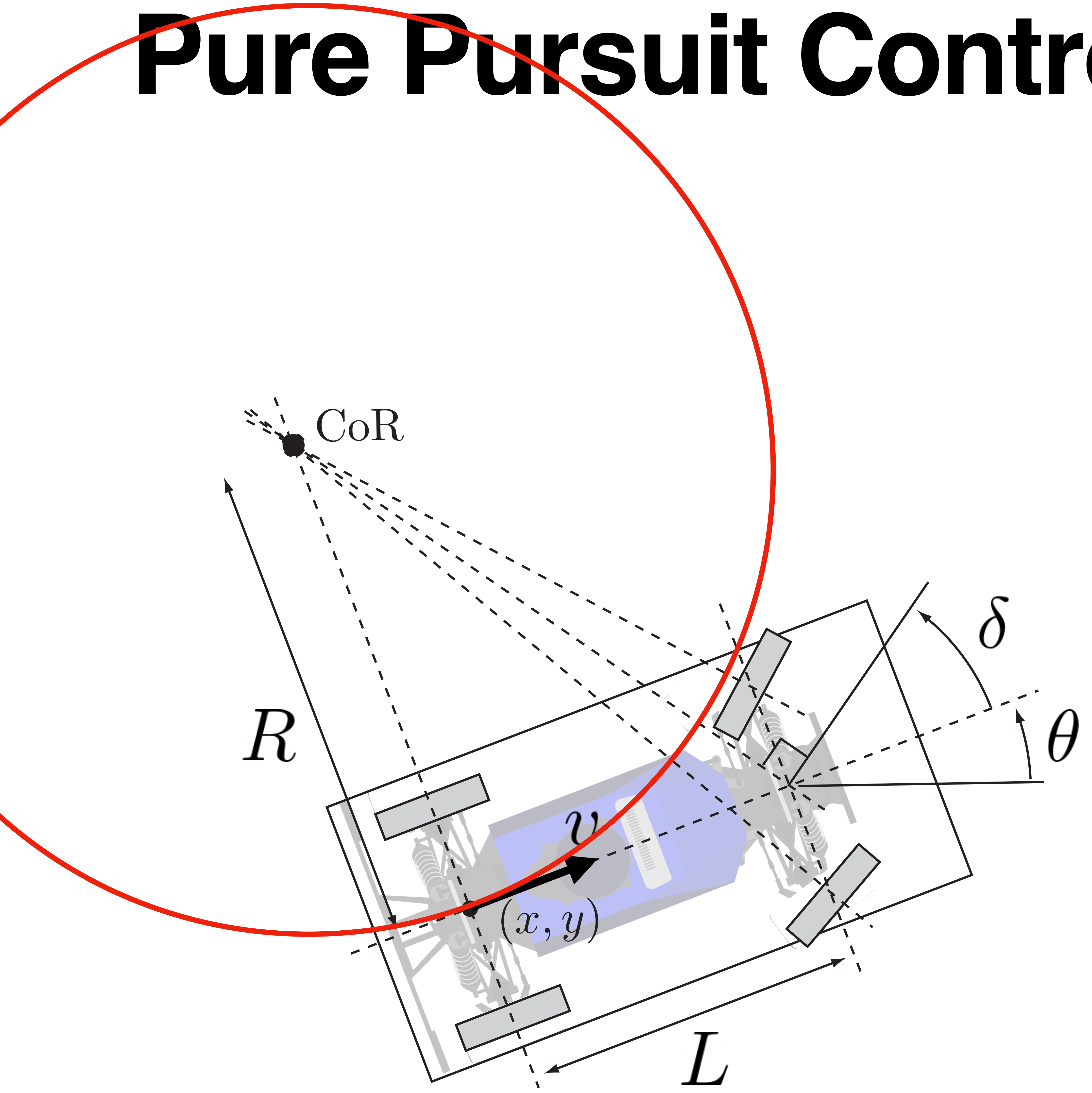


- Select a trajectory class $\xi \in \Xi$
- Select an optimization objective J

$$\xi^* = \arg\min_{\xi \in \Xi} J(\xi)$$

- Execute
- Repeat until end

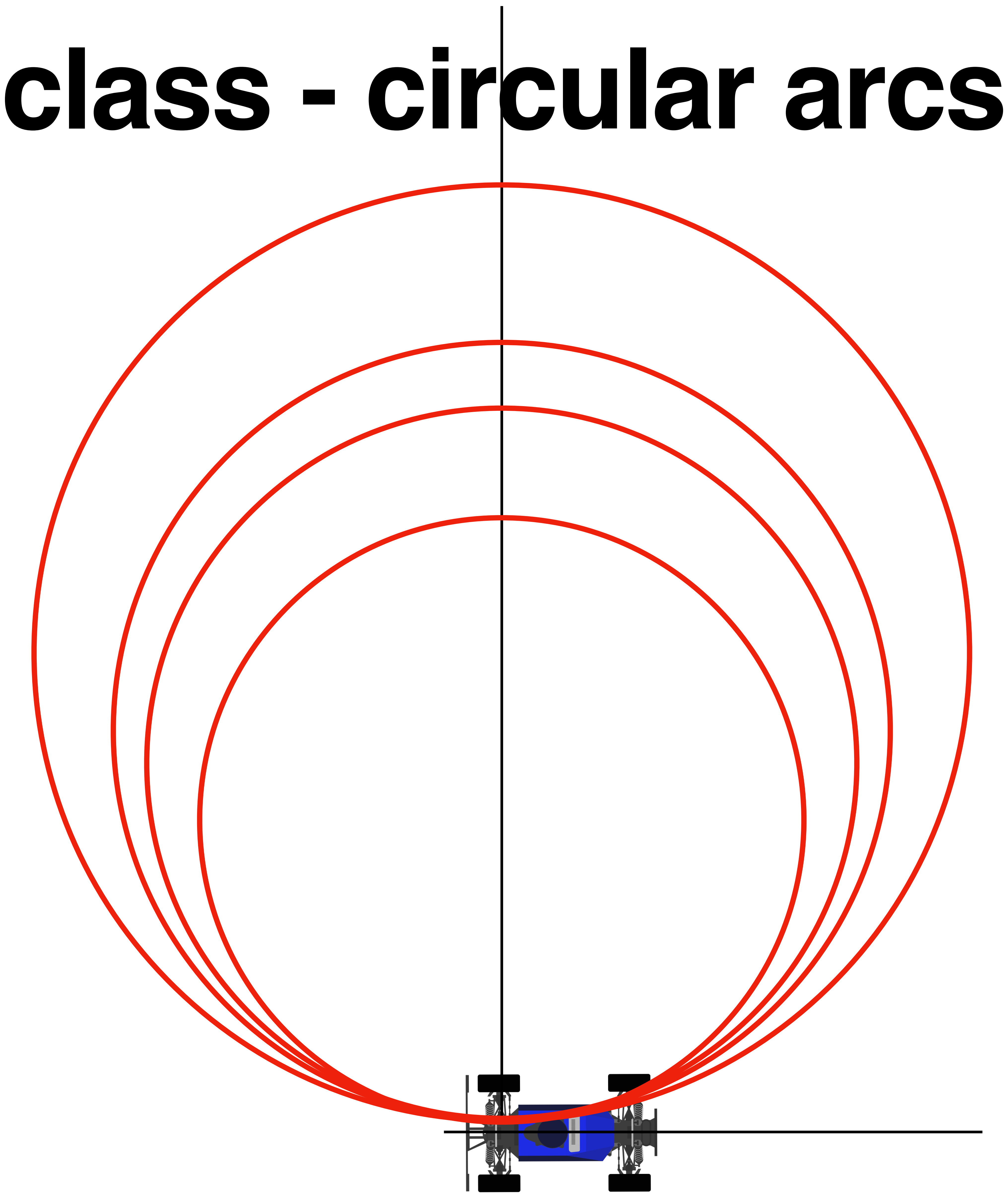
Pure Pursuit Controller



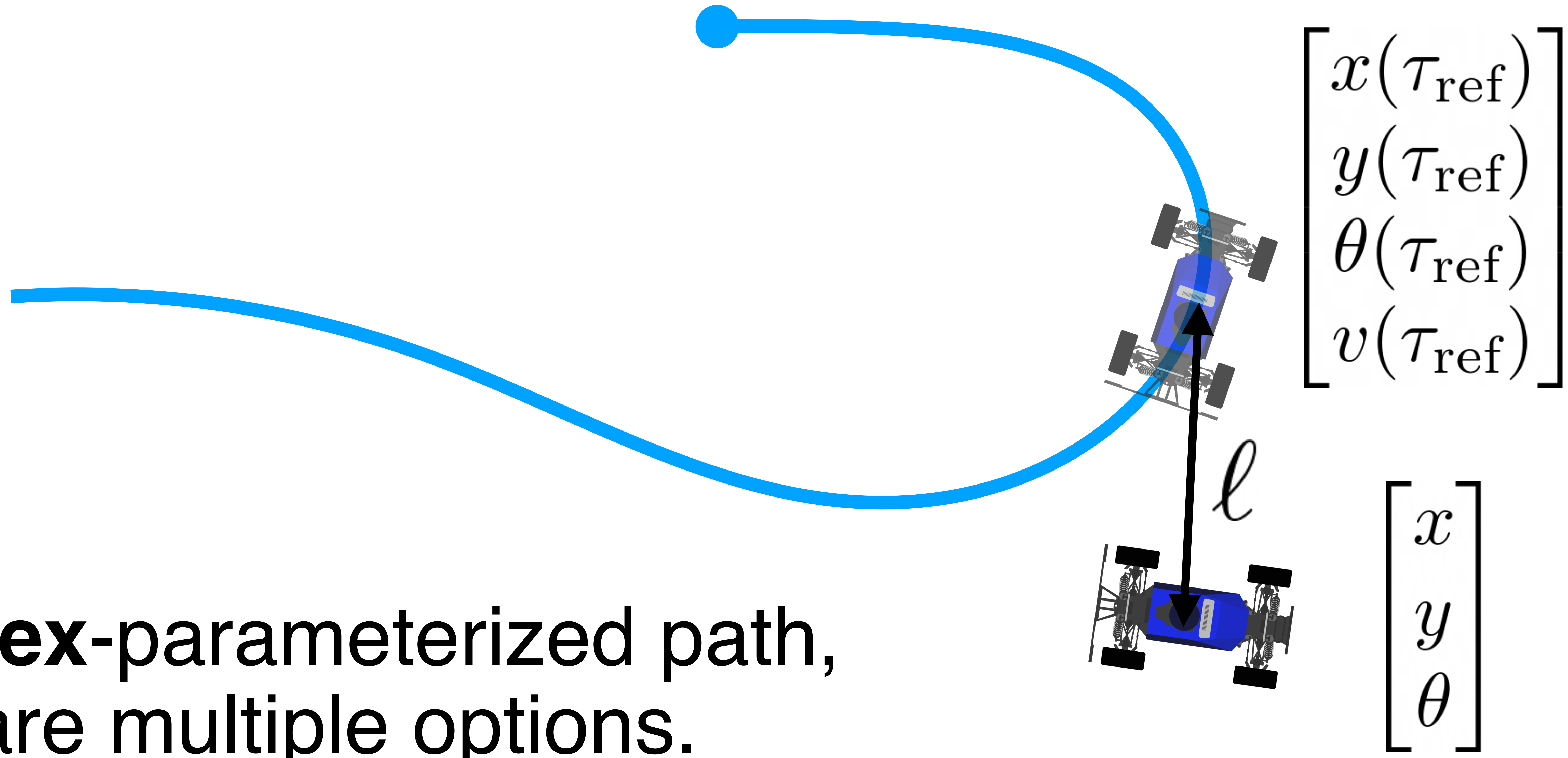
- Assume the car is moving with fixed steering angle

COULTER, 1992

Trajectory class - circular arcs



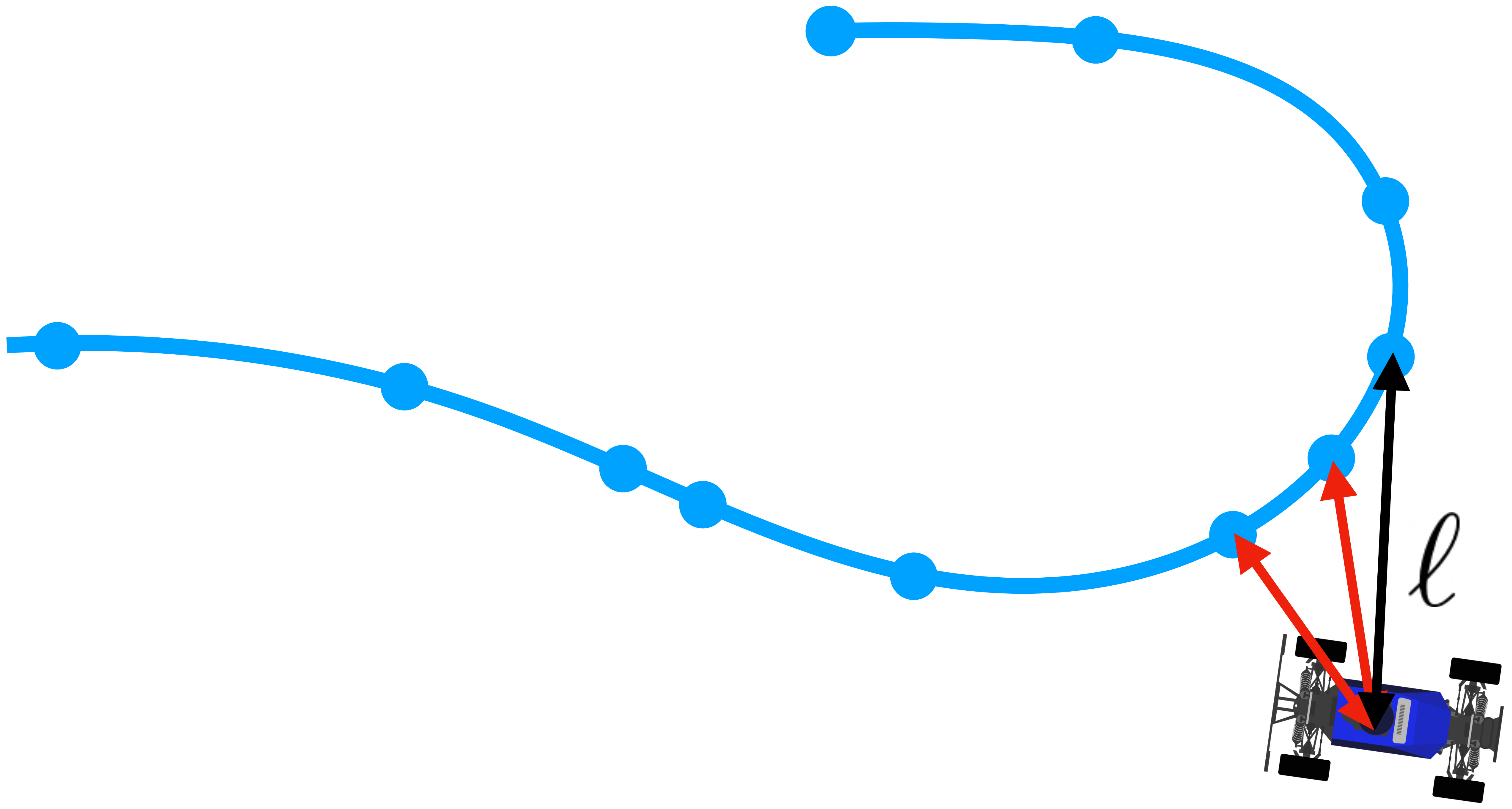
How do we choose a reference position?



For an **index**-parameterized path, there are multiple options.

Lookahead $$\tau_{\text{ref}} = \arg\min_{\tau} \left(\| \begin{bmatrix} x & y \end{bmatrix}^\top - \begin{bmatrix} x(\tau) & y(\tau) \end{bmatrix}^\top \| - \ell \right)^2$$

How do we choose a reference position?

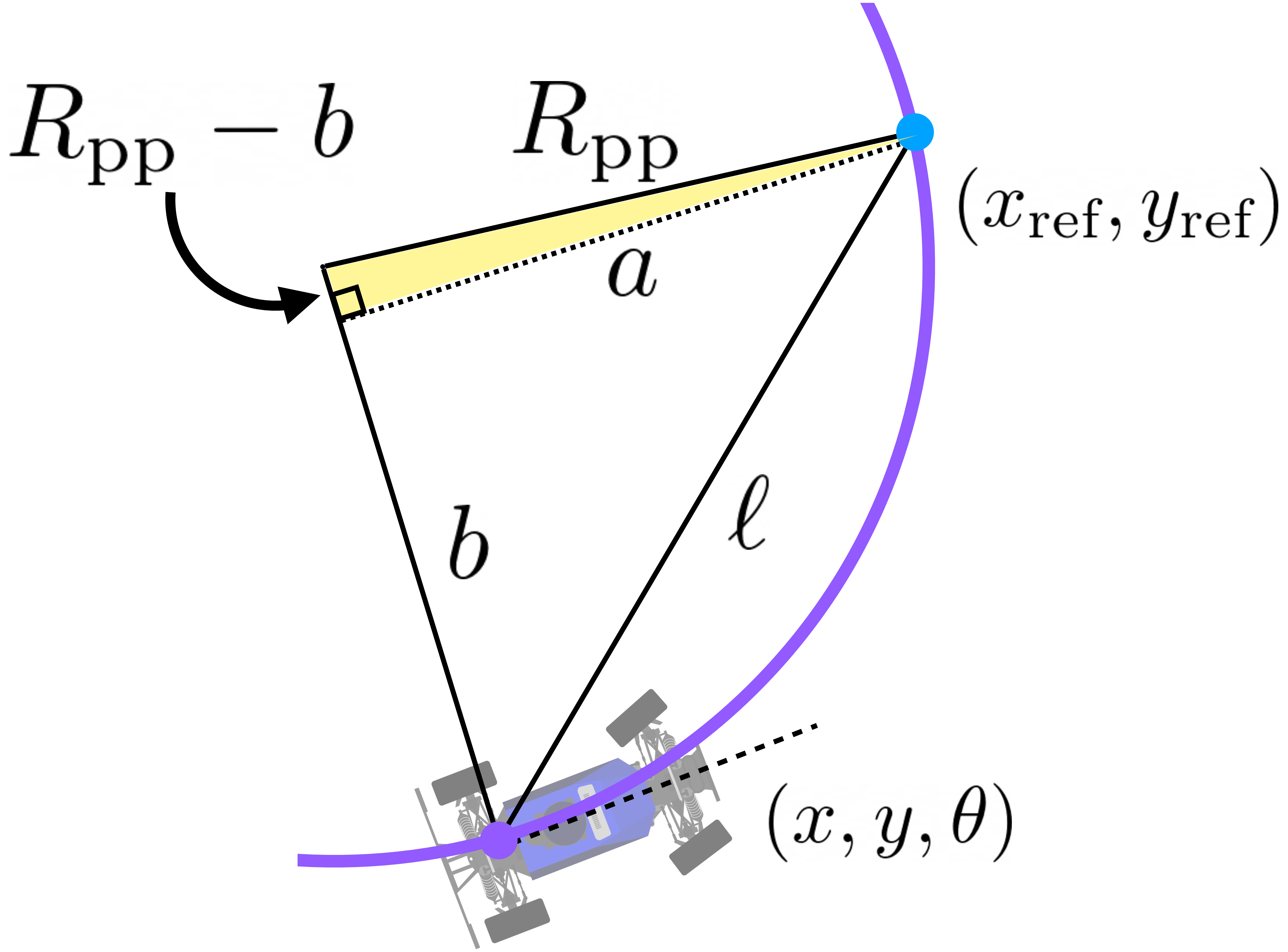


Lookahead $$\tau_{\text{ref}} = \arg\min_{\tau} \left(\| \begin{bmatrix} x & y \end{bmatrix}^\top - \begin{bmatrix} x(\tau) & y(\tau) \end{bmatrix}^\top \| - \ell \right)^2$$

Computing the Arc Radius

$$(R_{\text{pp}} - b)^2 + a^2 = R_{\text{pp}}^2$$

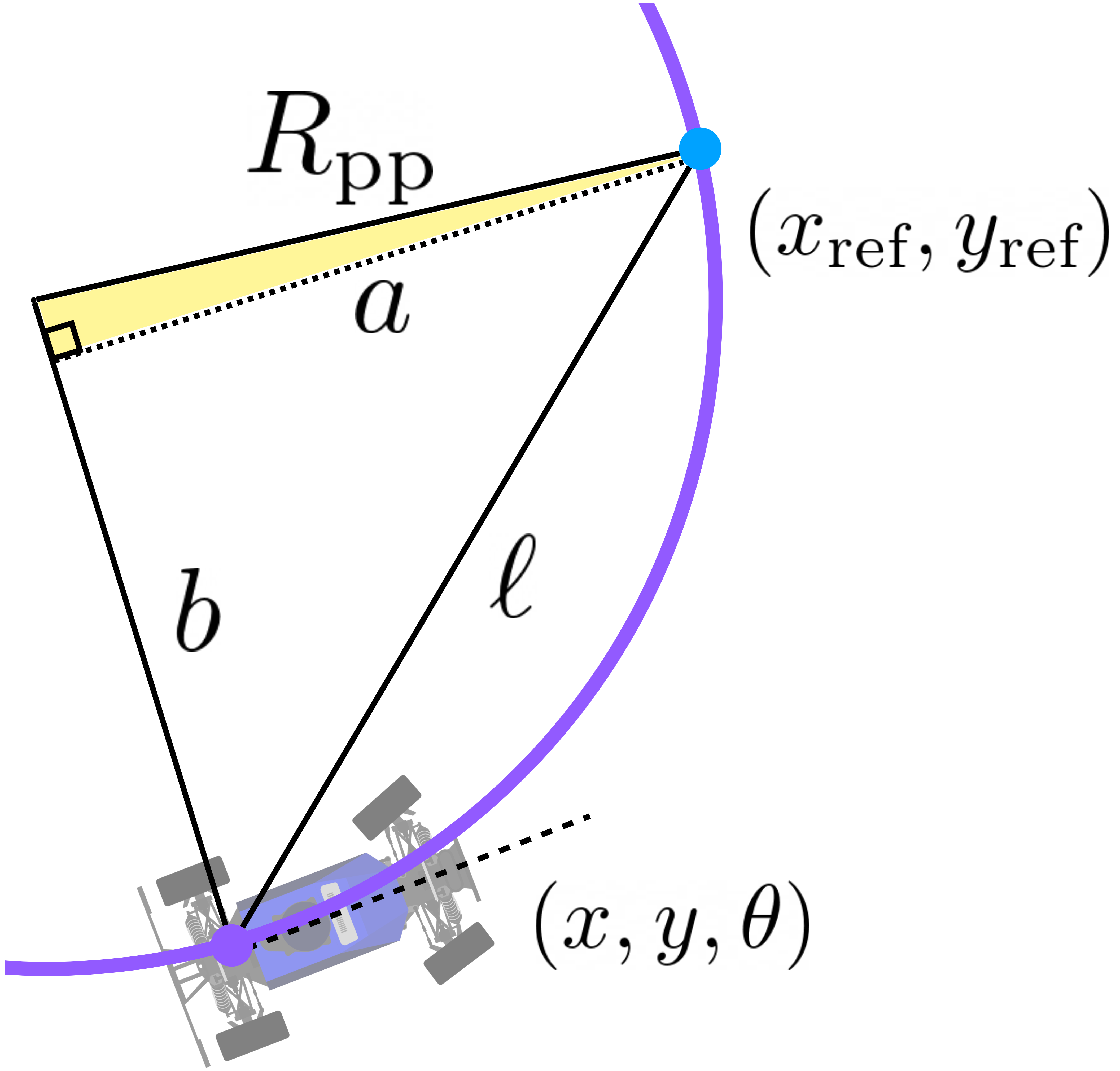
$$R_{\text{pp}} = \frac{a^2 + b^2}{2b}$$



Computing the Arc Radius

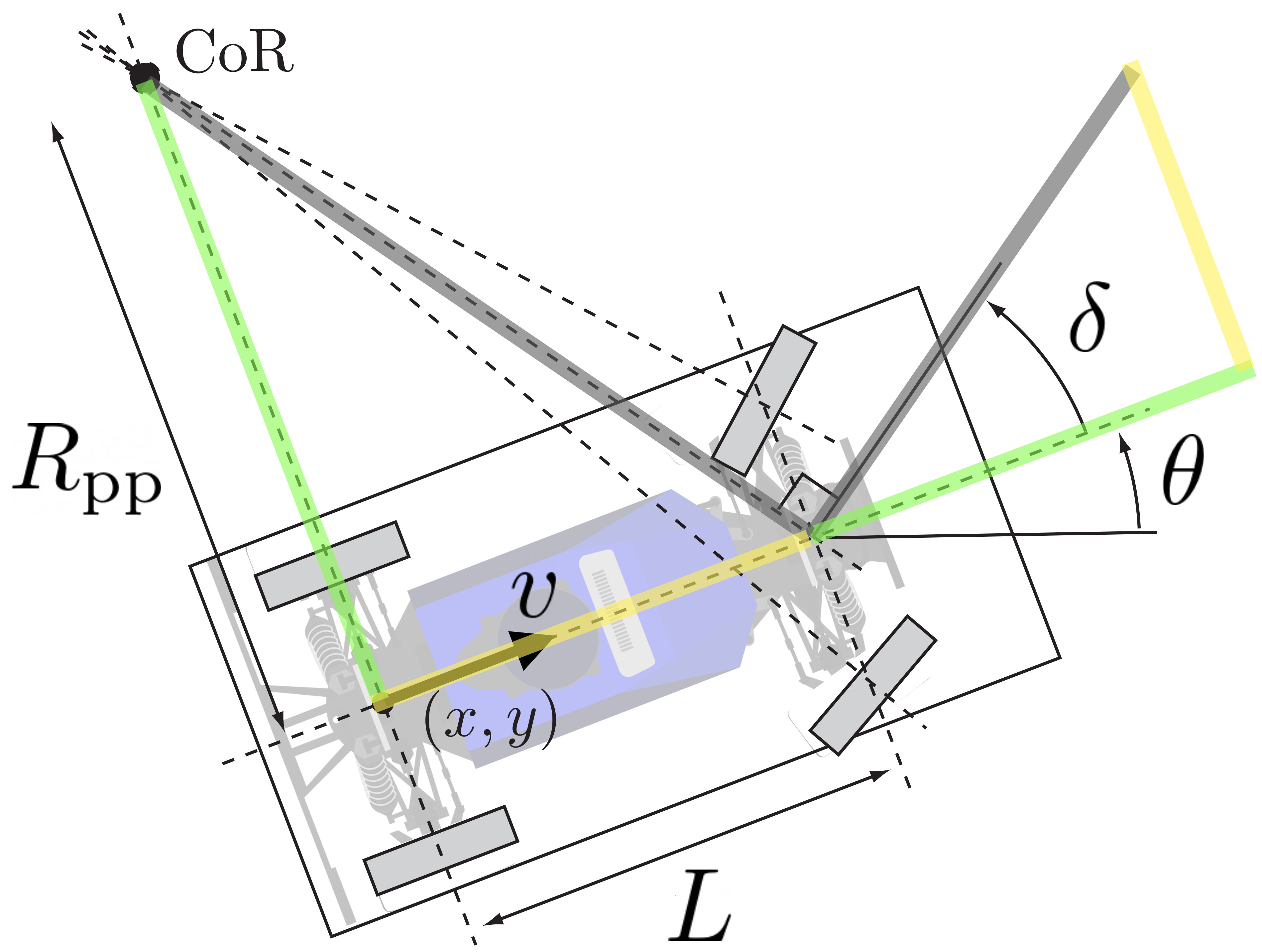
$$R_{\text{pp}} = \frac{a^2 + b^2}{2b}$$

$$\begin{bmatrix} a \\ b \end{bmatrix} = R(-\theta)\left(\begin{bmatrix} x_{\text{ref}} \\ y_{\text{ref}} \end{bmatrix} - \begin{bmatrix} x \\ y \end{bmatrix}\right)$$



Different than cross-track error (this is ref. position in robot frame; vice versa for cross-track error)

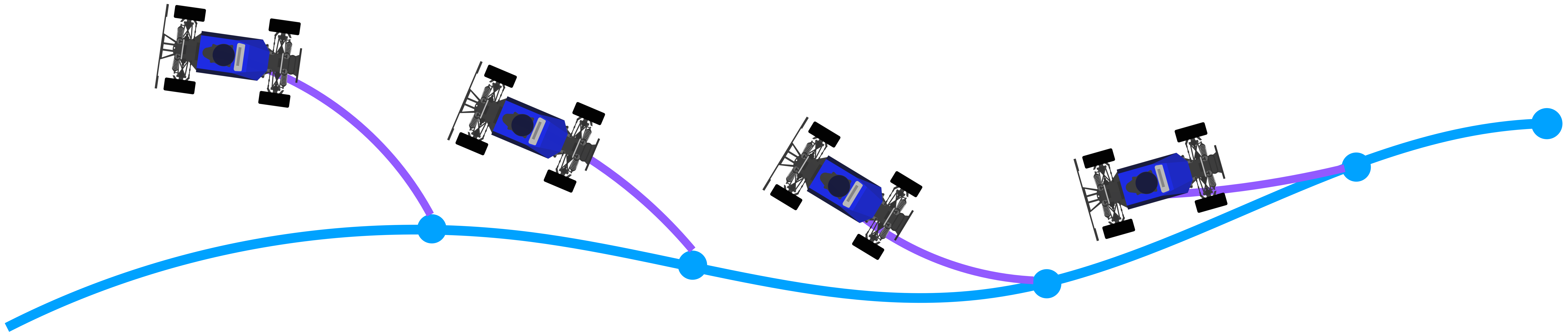
Computing the Steering Angle



$$R_{\mathrm{pp}} = \frac{a^2 + b^2}{2b}$$

$$\tan\delta = \frac{L}{R_{\mathrm{pp}}}$$

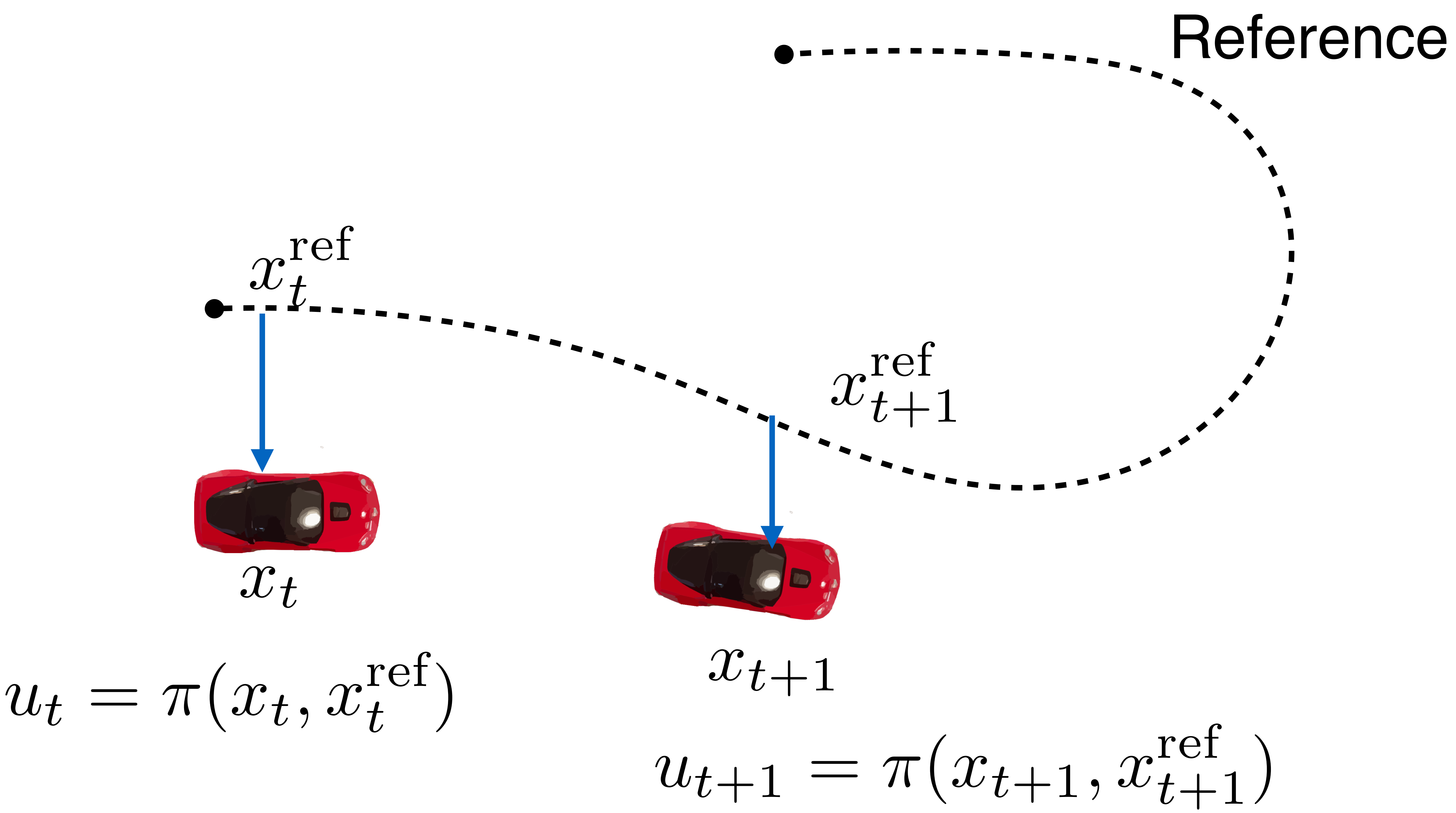
Pure Pursuit: Chasing the Lookahead



Controller Design Decisions

1. Get a reference path/trajectory to track
2. Pick a reference state from the reference path/trajectory
3. Compute error to reference state
4. Compute control law to minimize error

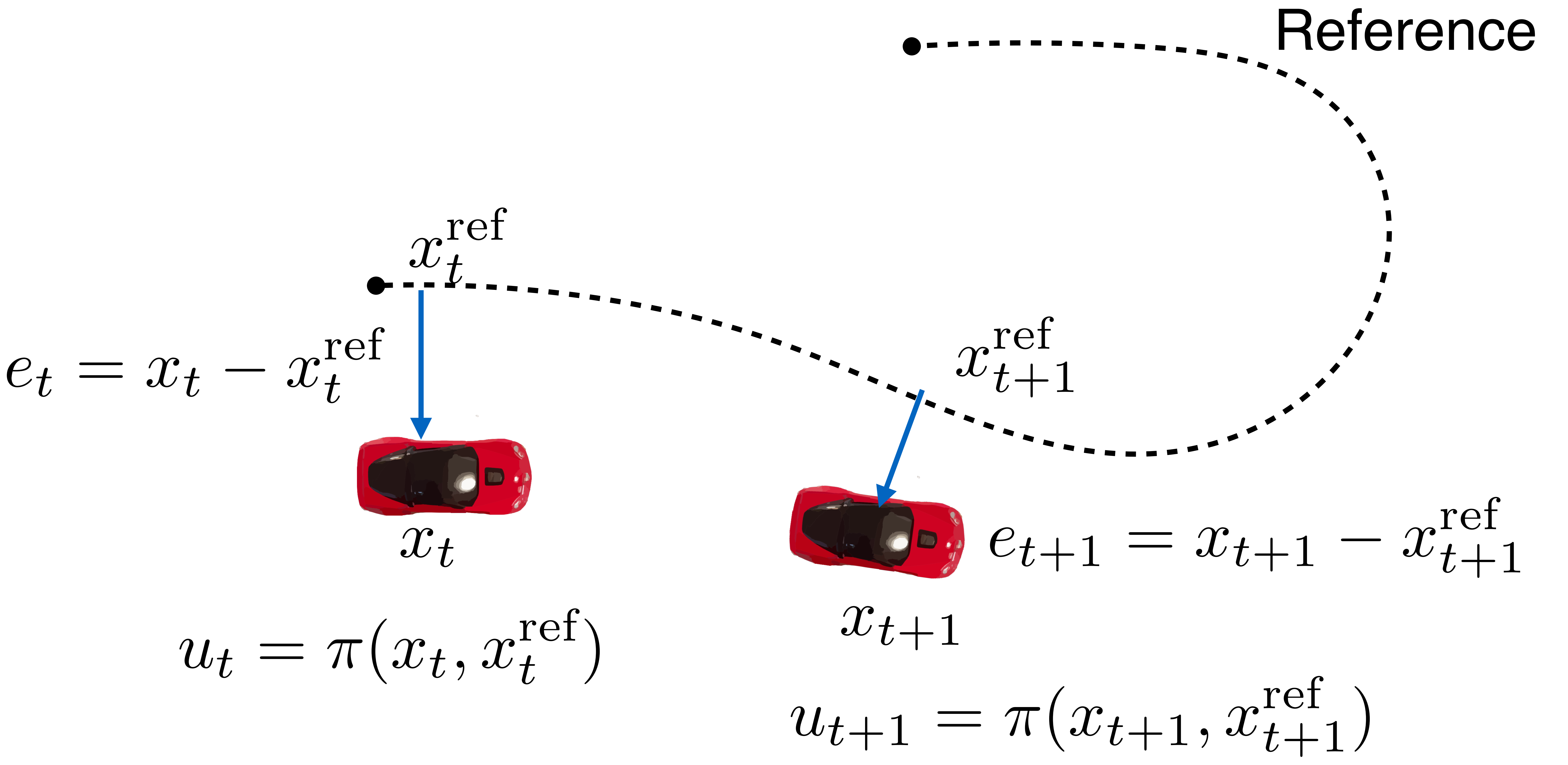
Recap: Feedback control framework



Look at current state error and compute control actions

Goal: To drive error to 0 … to optimally drive it to 0

Recap: Feedback control framework



Look at current state error and compute control actions

Goal: To drive error to 0 … to optimally drive it to 0

Limitations of this framework

A fixed control law that looks at instantaneous feedback

$$u_t = \pi(x_t, x_t^{\text{ref}})$$

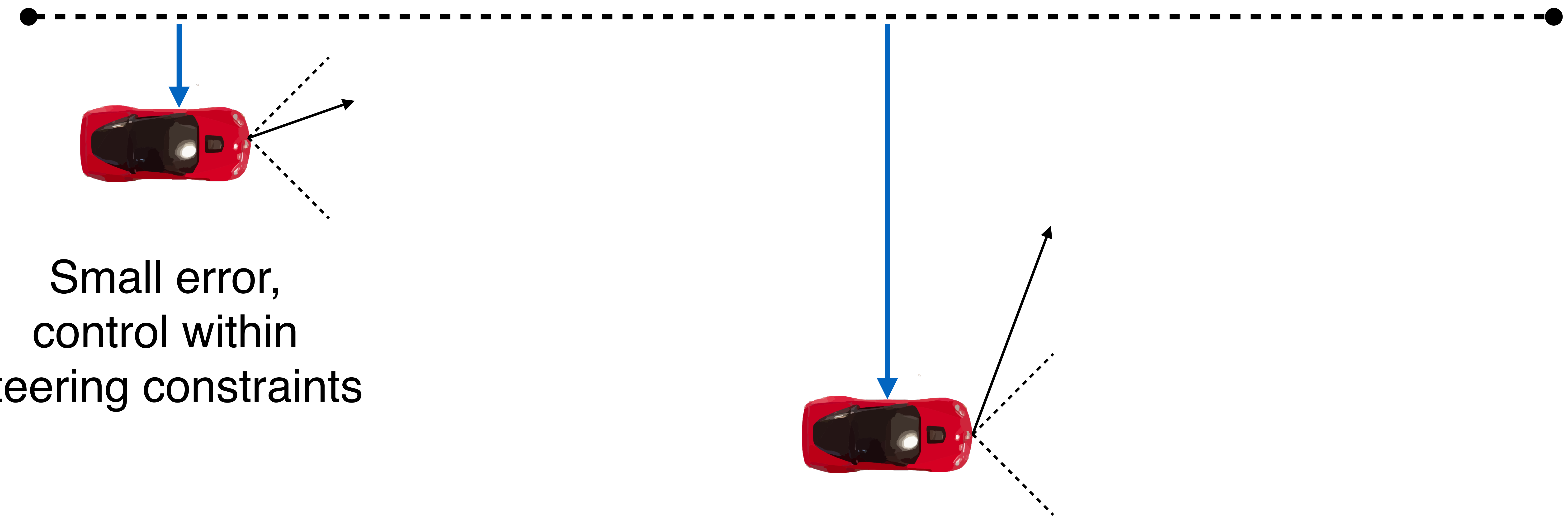
Fixed

Reference

Why is it so difficult to create a magic control law?

Problem 1: What if we have constraints?

Simple scenario: Car tracking a straight line



Small error, control within steering constraints

Large error, control violates steering constraints

We could "clamp control command" …
but what are the implications?

General problem: Complex models

Dynamics $$x_{t+1} = f(x_t, u_t)$$

Constraints $$g(x_t, u_t) \leq 0$$

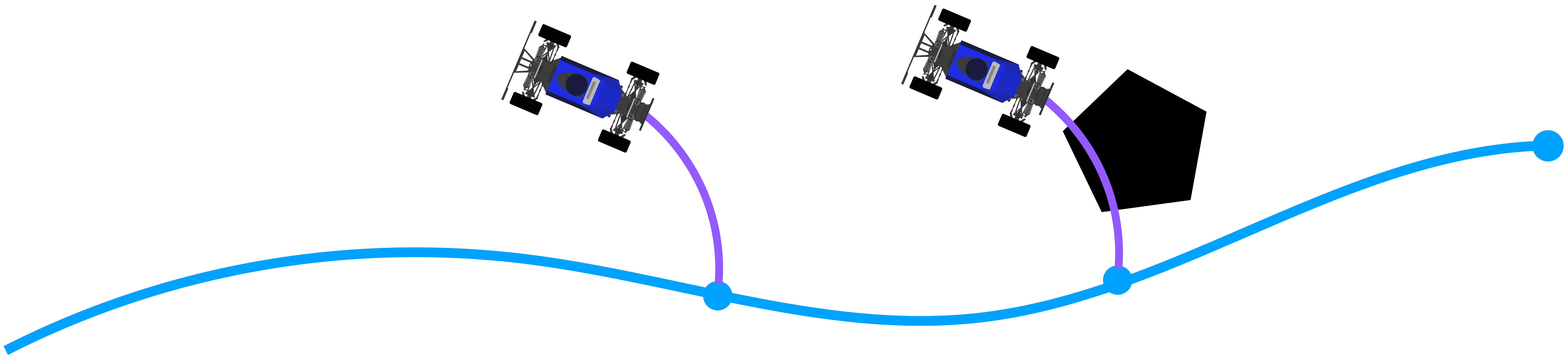
Such complex models imply we need to:

1. Predict the implications of control actions

2. Do corrections NOW that would affect the future

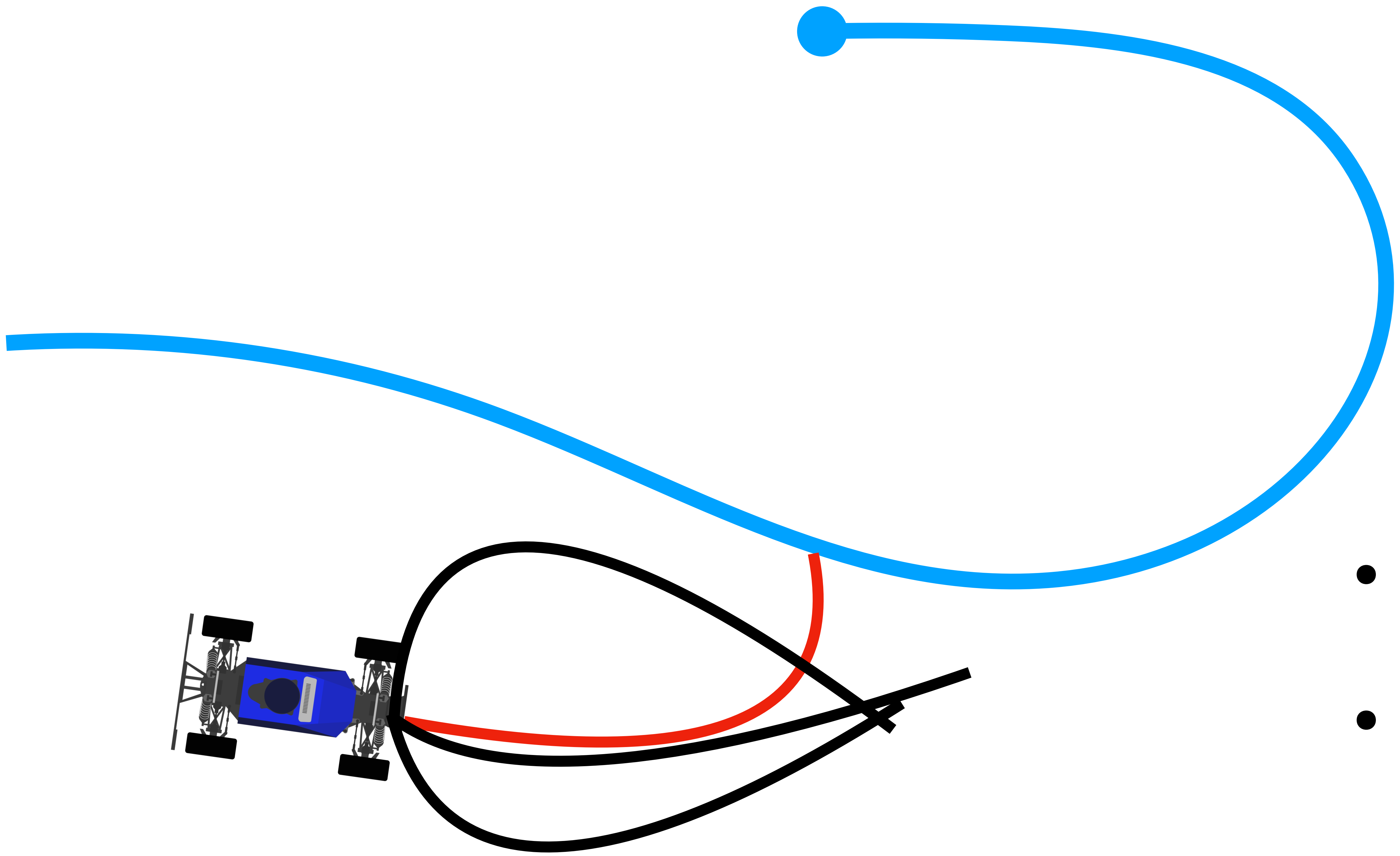
3. It may not be possible to find one law - might need to predict

Problem 2: What if some errors are worse than others?



We need a cost function that penalizes states non-uniformly

Model Predictive Control / Receding Horizon Control



- Select a trajectory class $\xi \in \Xi$
- Select an optimization objective J

$$\xi^* = \arg\min_{\xi \in \Xi} J(\xi)$$

- Execute
- Repeat until end

Control as an Optimization Problem

- For a sequence of H control actions
  1. Use model to predict consequence of actions (i.e., H future states)
  2. Evaluate the cost function and check constraints
- Compute optimal sequence of H control actions, minimizing cost while satisfying constraints

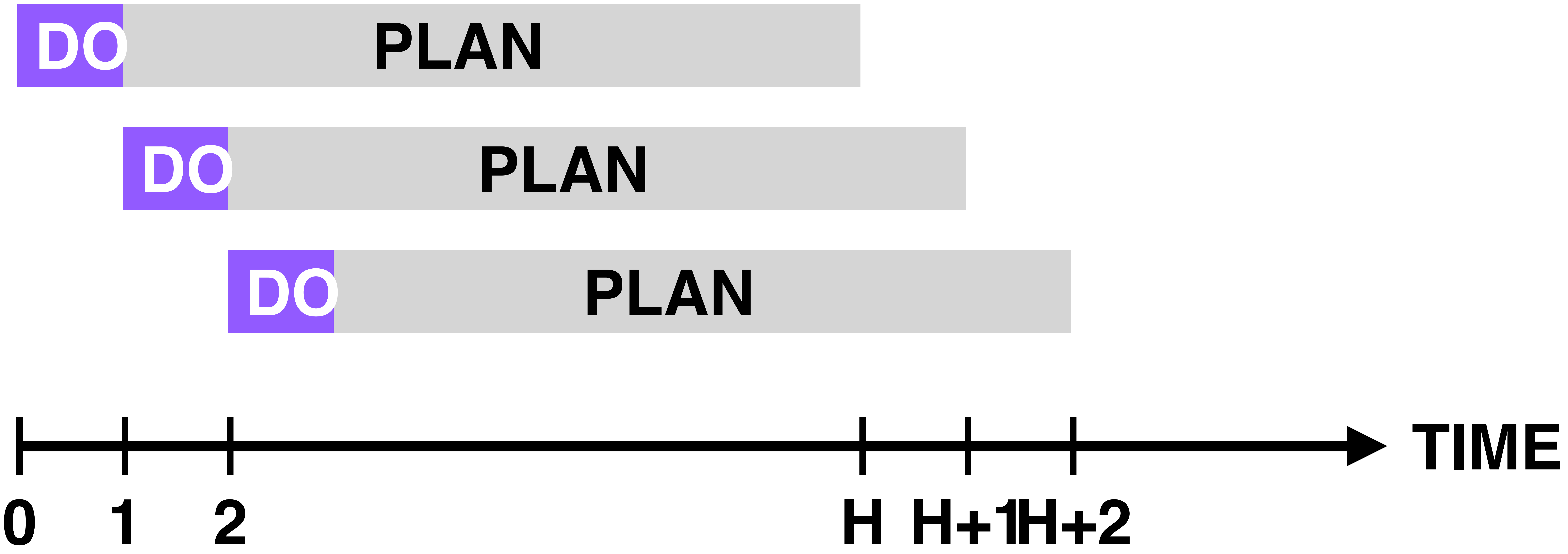
$$\min_{u_{t+1},\cdots,u_{t+H}} \sum_{k=t+1}^{t+H} J(u_k, x_k)$$

SUCH THAT

$$x_k = f(x_{k-1}, u_k)$$

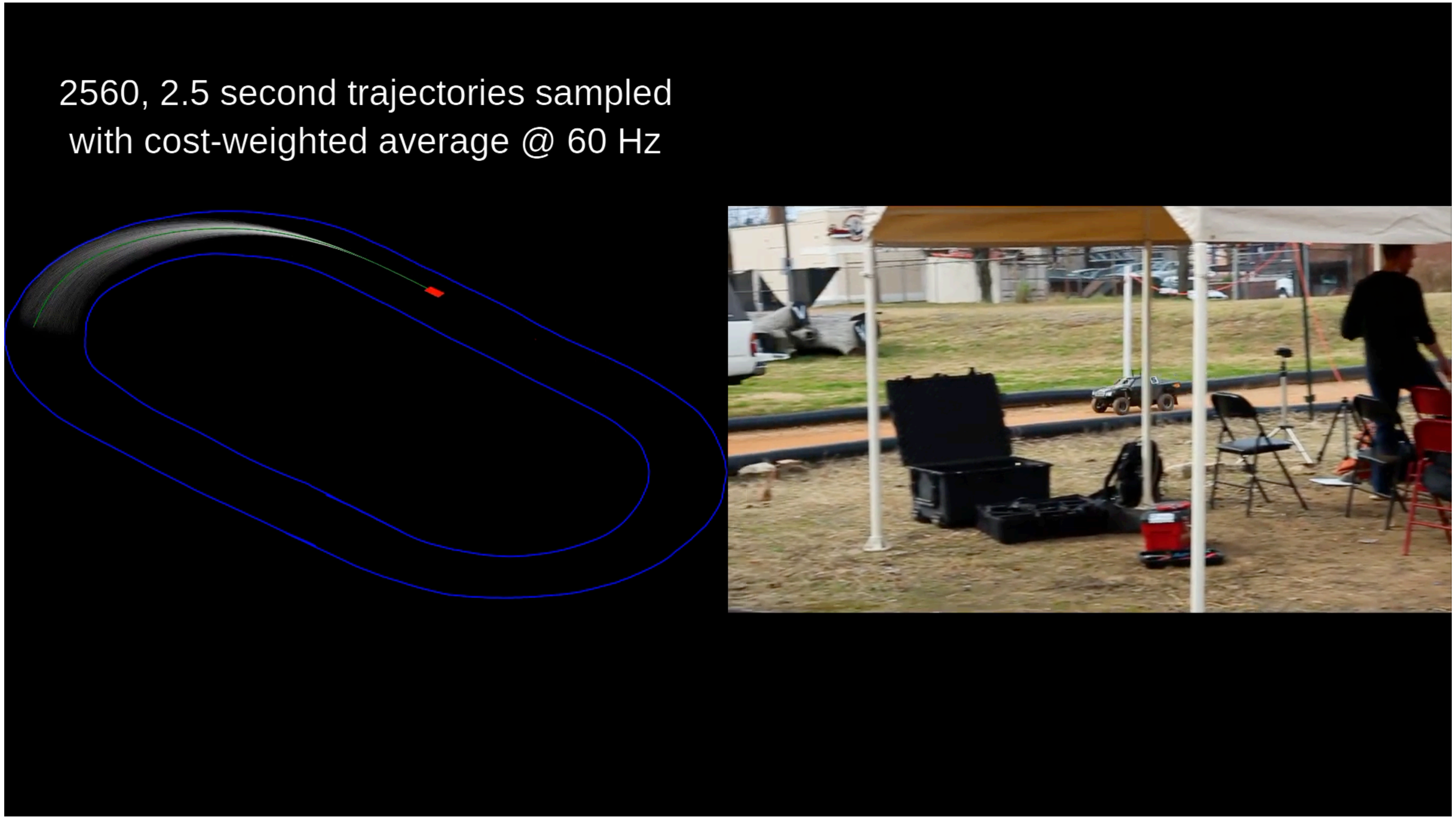
$$g(x_{k-1}, u_k) \leq 0$$

Model-Predictive Control (MPC) Framework



- Solve the H step optimization problem
- Execute first command of the optimal sequence of H control actions

MPC in Action: Georgia Tech AutoRally



WILLIAMS ET AL., 2015 **HTTPS://YOUTU.BE/1AR2-OHCXSQ**

CSE 478 Robot Autonomy
Model Predictive Control
Pure Pursuit

Abhishek Gupta (abhgupta@cs)
Siddhartha Srinivasa (siddh@cs)

TAs:
Carolina Higuera (chiguera@cs)
Rishabh Jain (jrishabh@cs)
Entong Su (ensu@cs)