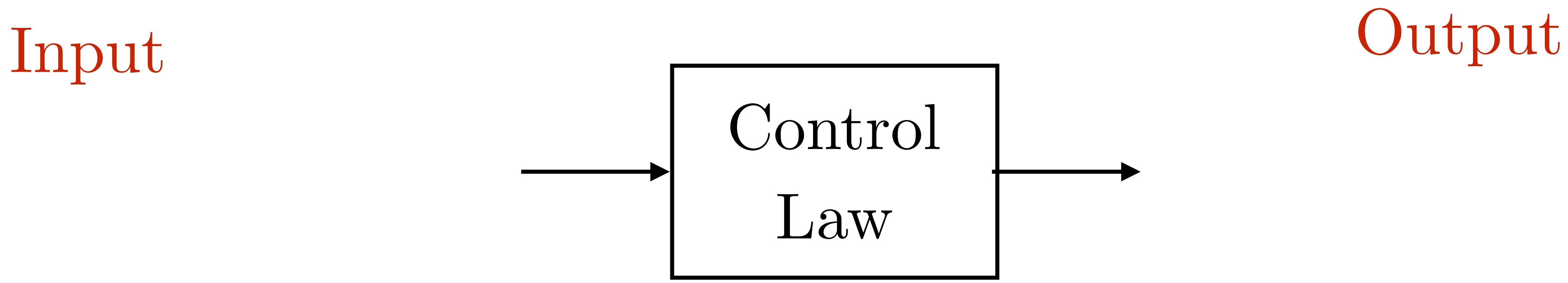
PID and Pure Pursuit Control

Sanjiban Choudhury

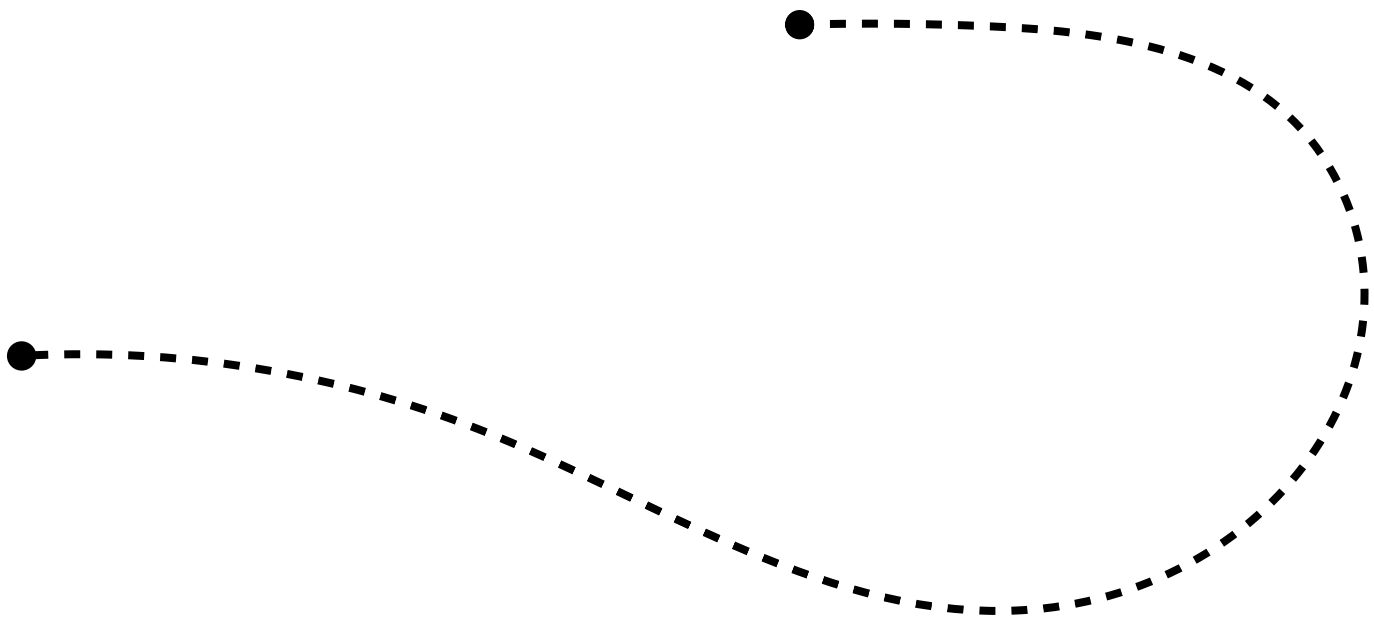
TAs: Matthew Rockett, Gilwoo Lee, Matt Schmittle

The Control API



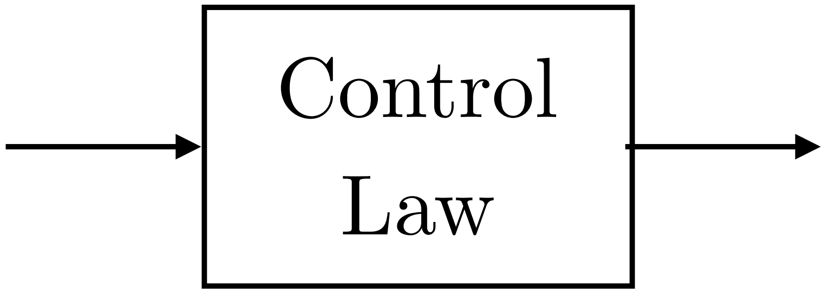
The Control API

$$x(\tau), y(\tau), \theta(\tau), v(\tau)$$



Input

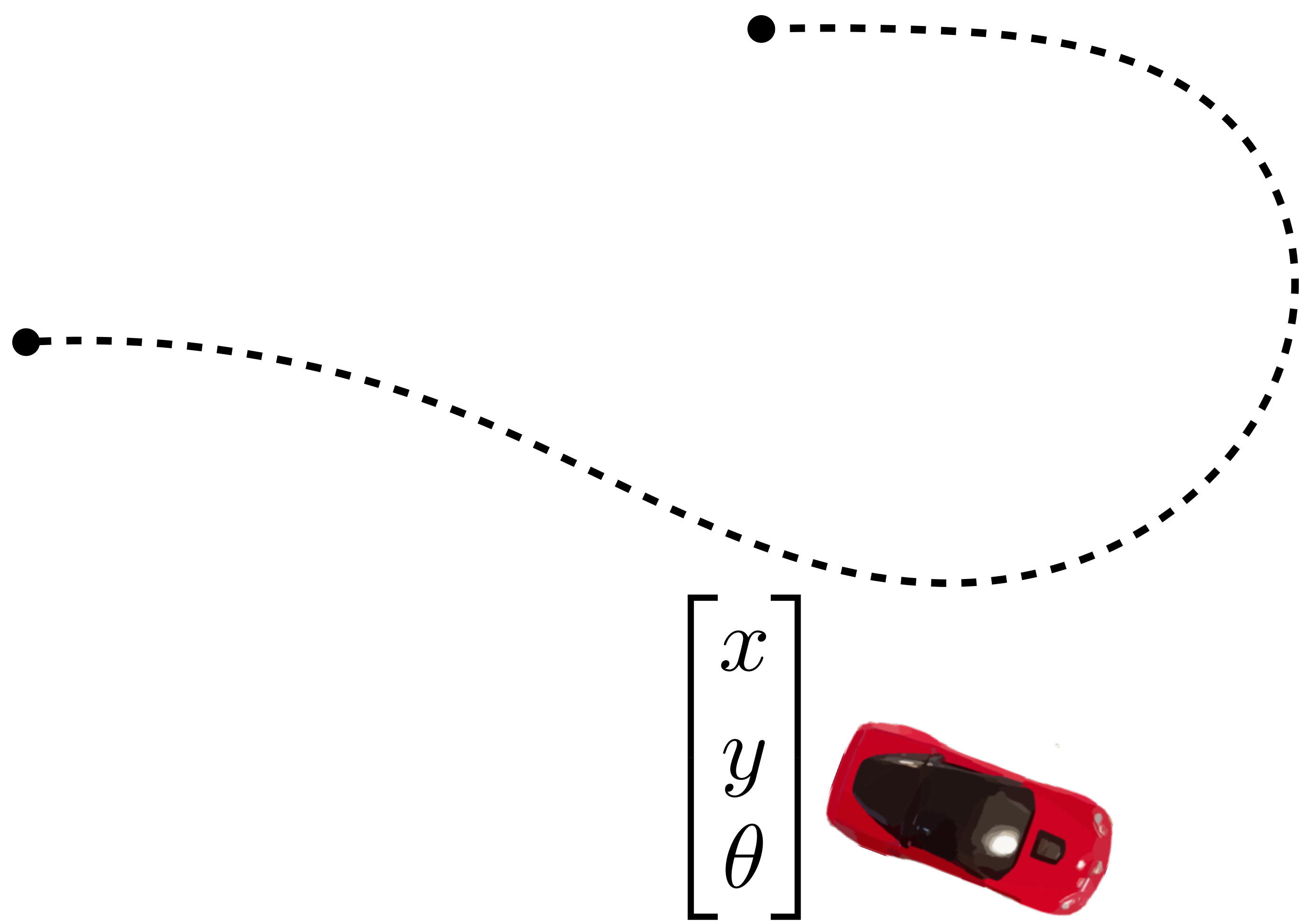
1. Reference path



Output

The Control API

$$x({\color{red}\tau}), y({\color{red}\tau}), \theta({\color{red}\tau}), v({\color{red}\tau})$$



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

Input

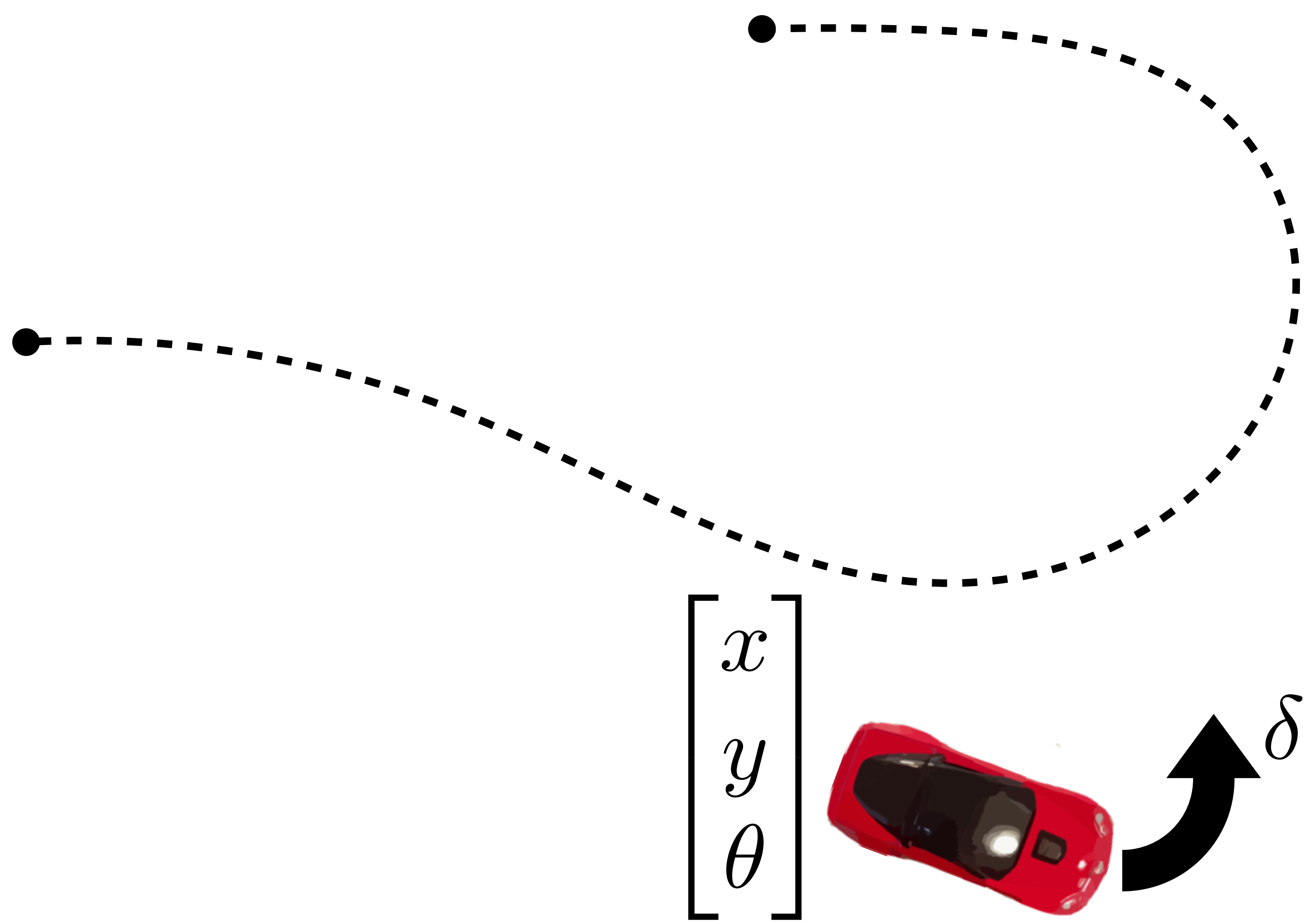
1. Reference path
2. Current state

$\longrightarrow$ Control Law $\longrightarrow$

Output

The Control API

$$x(\tau), y(\tau), \theta(\tau), v(\tau)$$



Input

1. Reference path
2. Current state

$\longrightarrow$ Control Law $\longrightarrow$

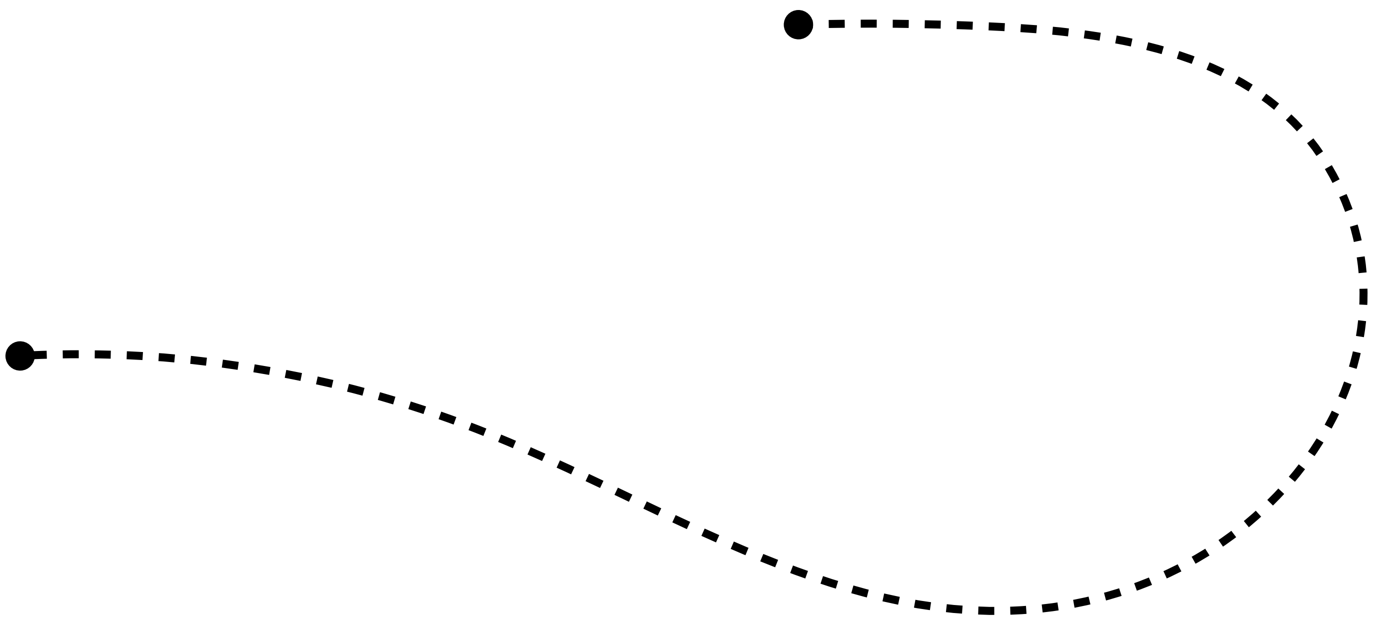
Output

Control action

Steps to designing a controller

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

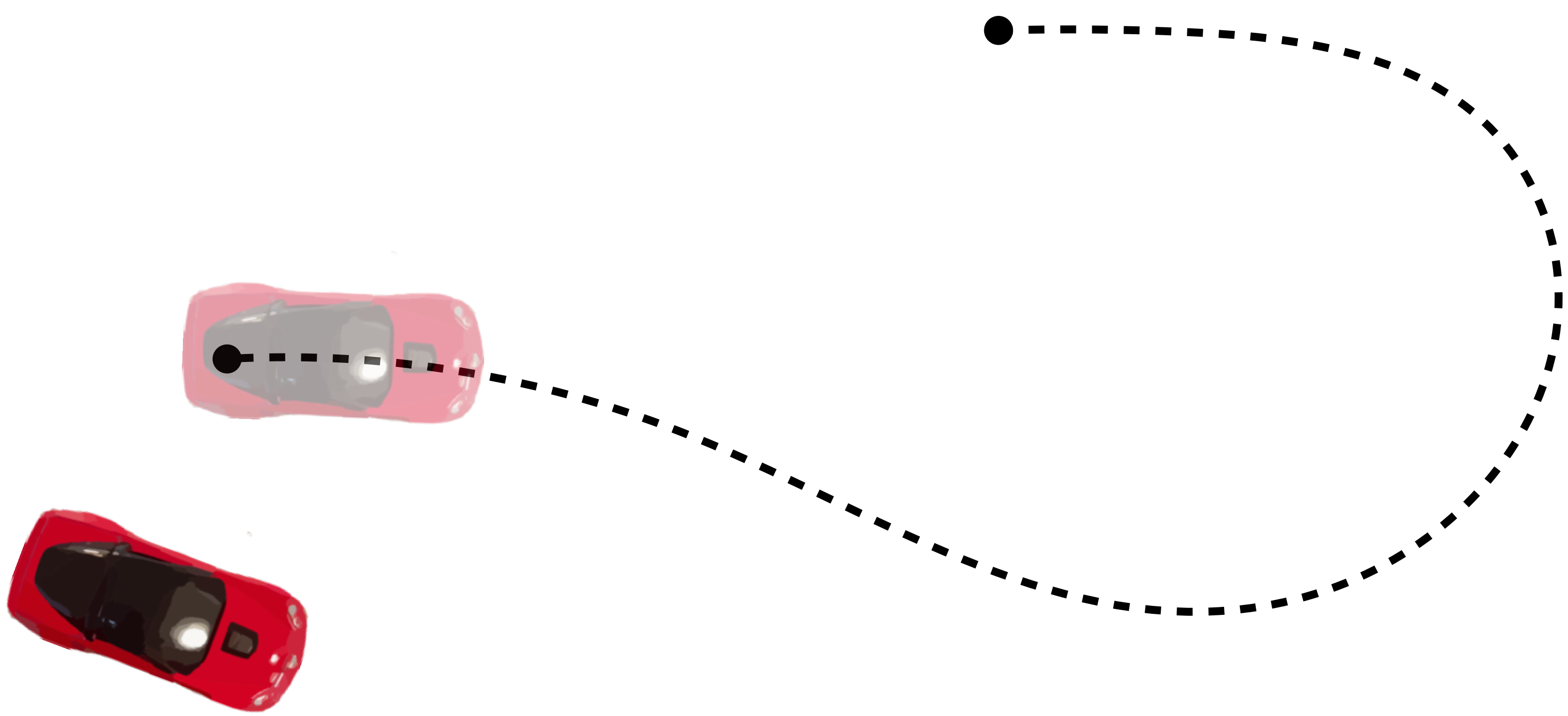
Rough idea of what happens across timesteps



Robot is trying to track a desired state on the reference path

(Take an action to drive down error between desired and current state)

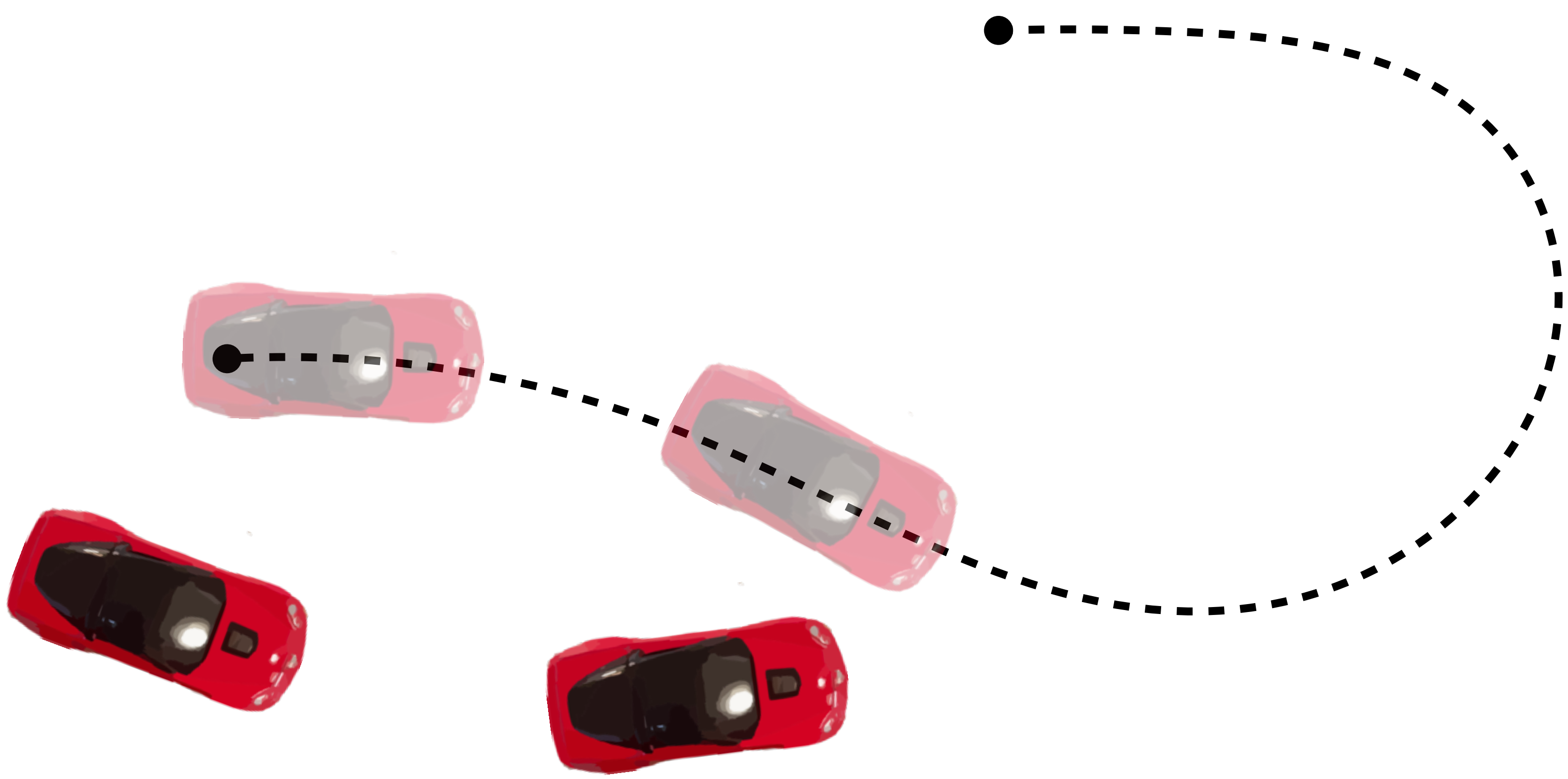
Rough idea of what happens across timesteps



Robot is trying to track a desired state on the reference path

(Take an action to drive down error between desired and current state)

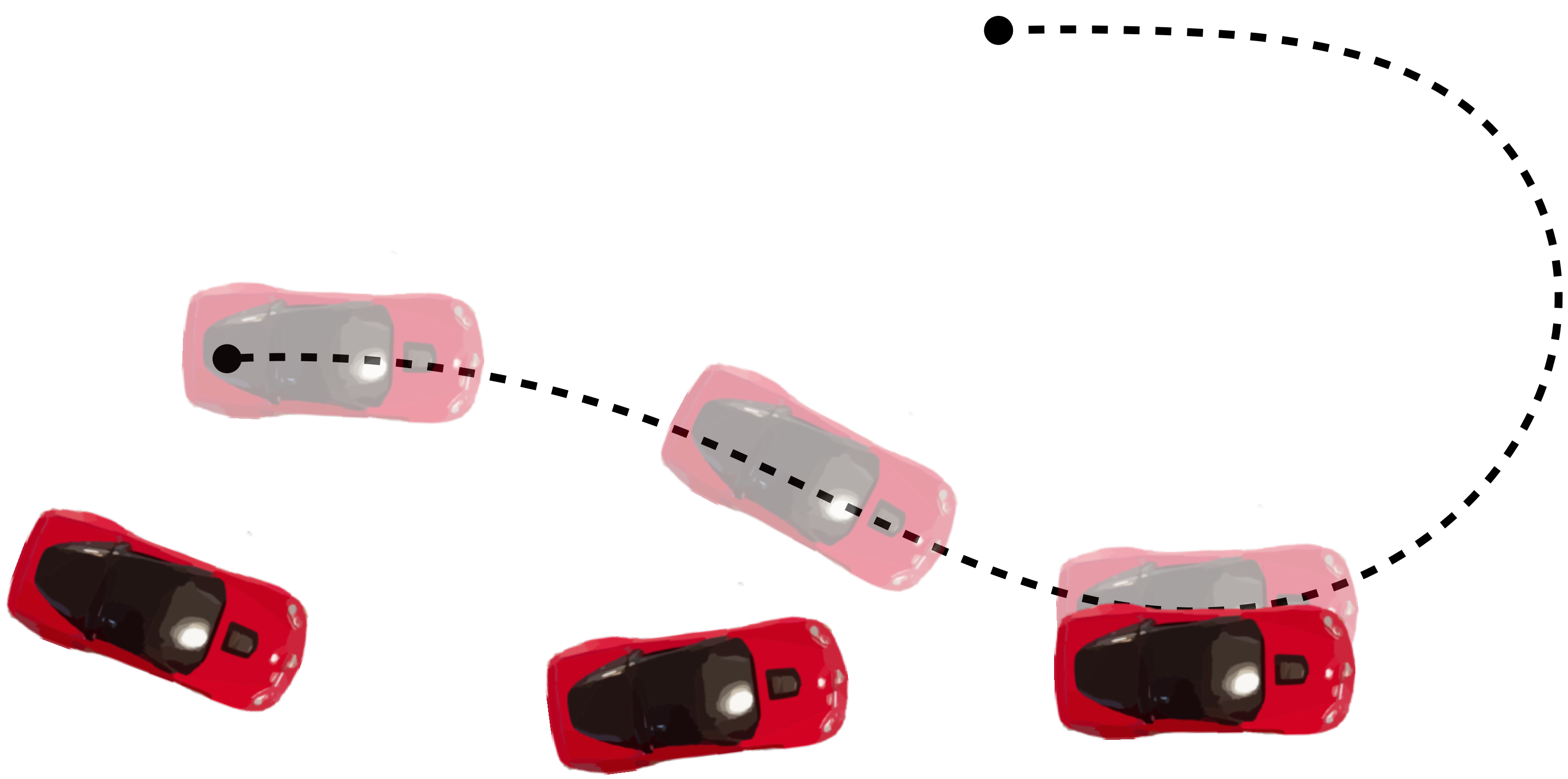
Rough idea of what happens across timesteps



Robot is trying to track a desired state on the reference path

(Take an action to drive down error between desired and current state)

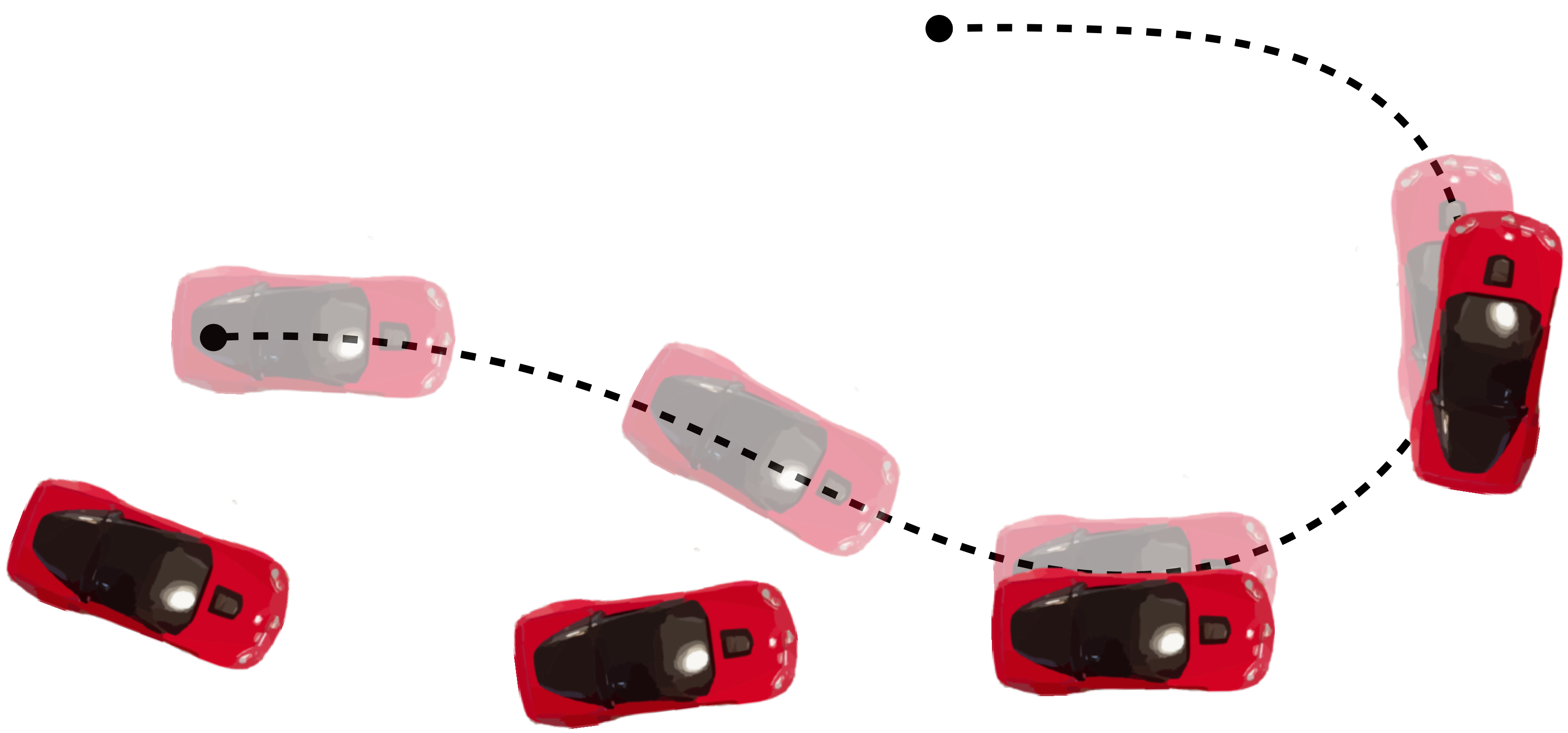
Rough idea of what happens across timesteps



Robot is trying to track a desired state on the reference path

(Take an action to drive down error between desired and current state)

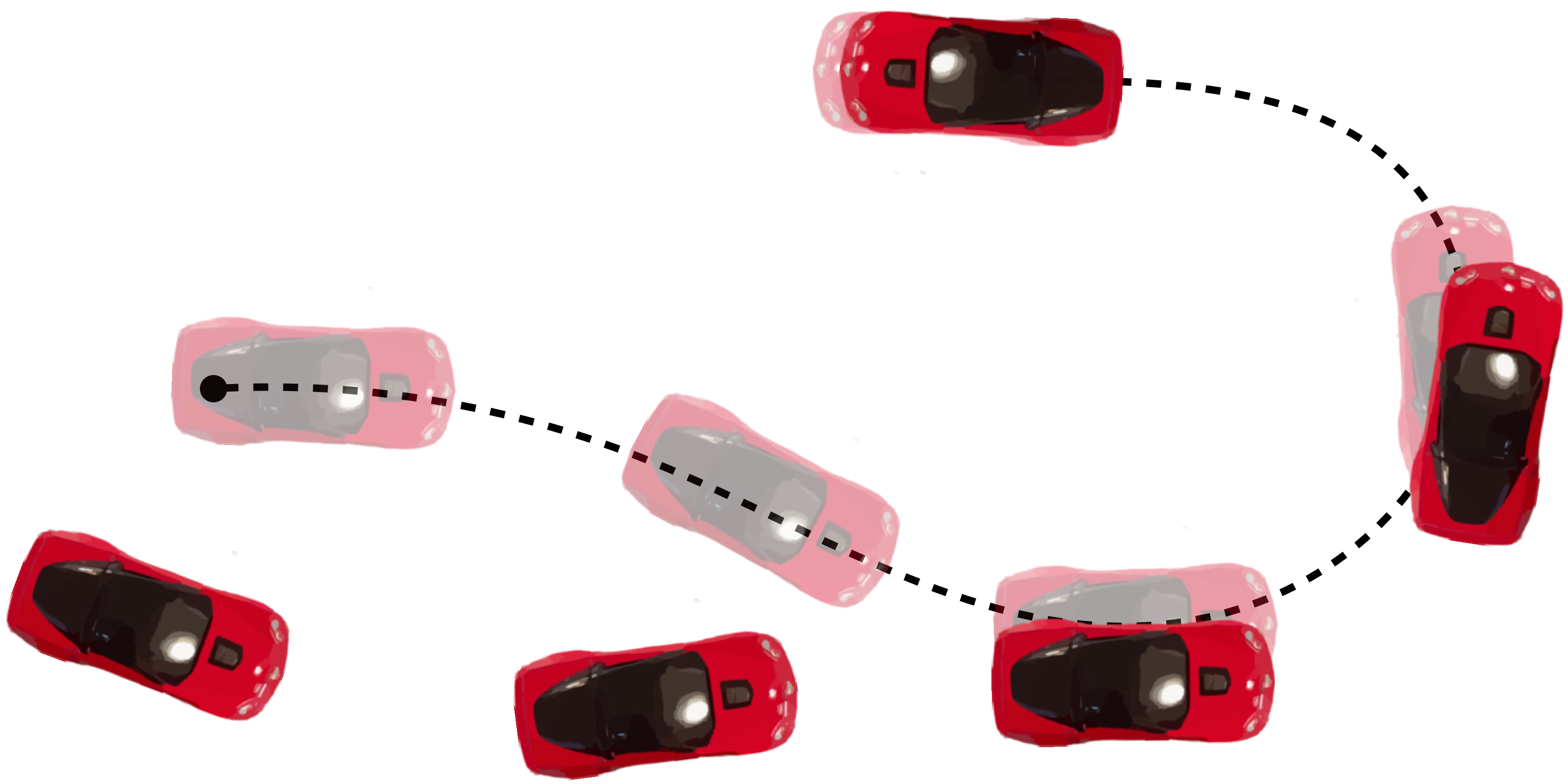
Rough idea of what happens across timesteps



Robot is trying to track a desired state on the reference path

(Take an action to drive down error between desired and current state)

Rough idea of what happens across timesteps



Robot is trying to track a desired state on the reference path

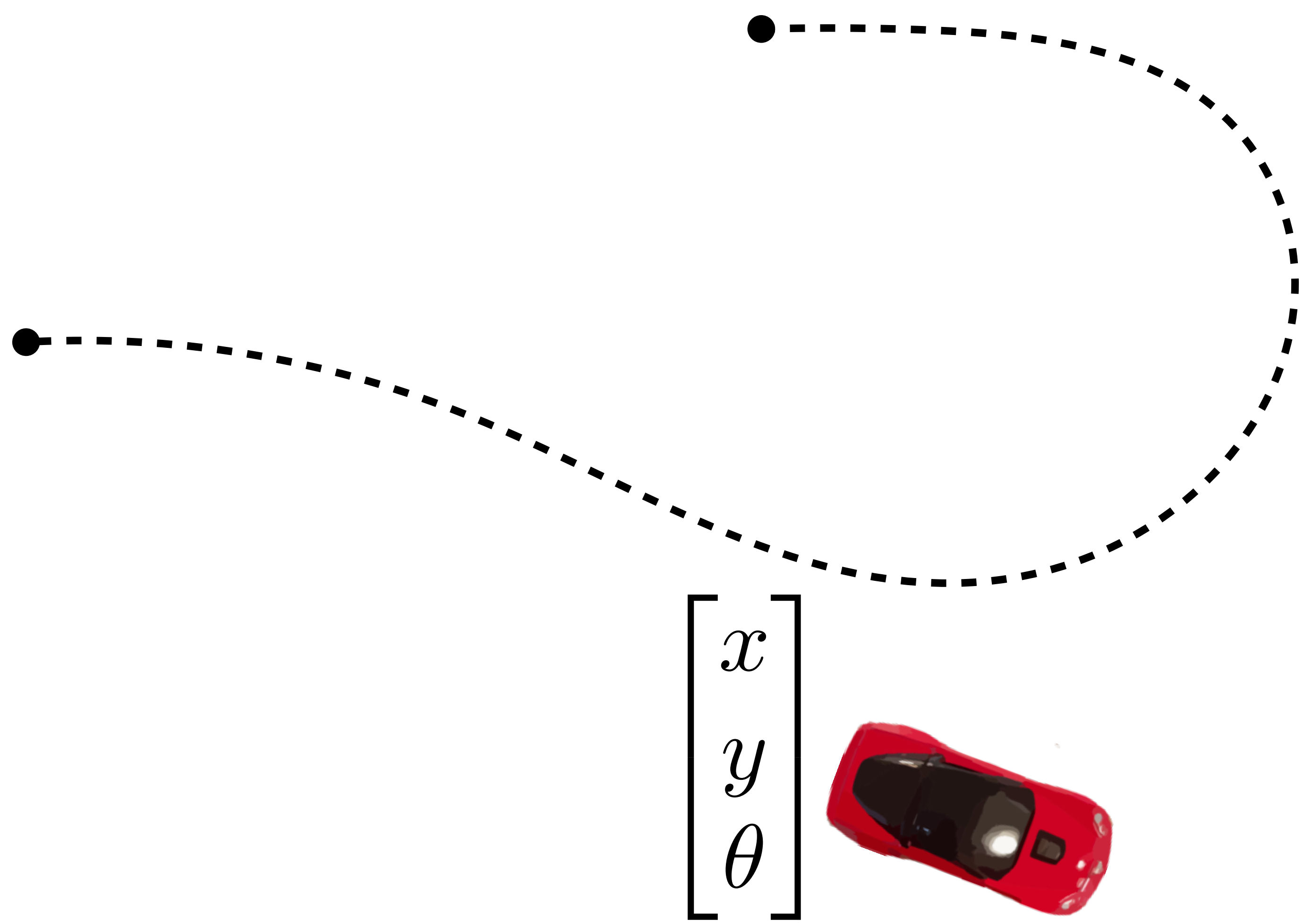
(Take an action to drive down error between desired and current state)

Steps to designing a controller

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

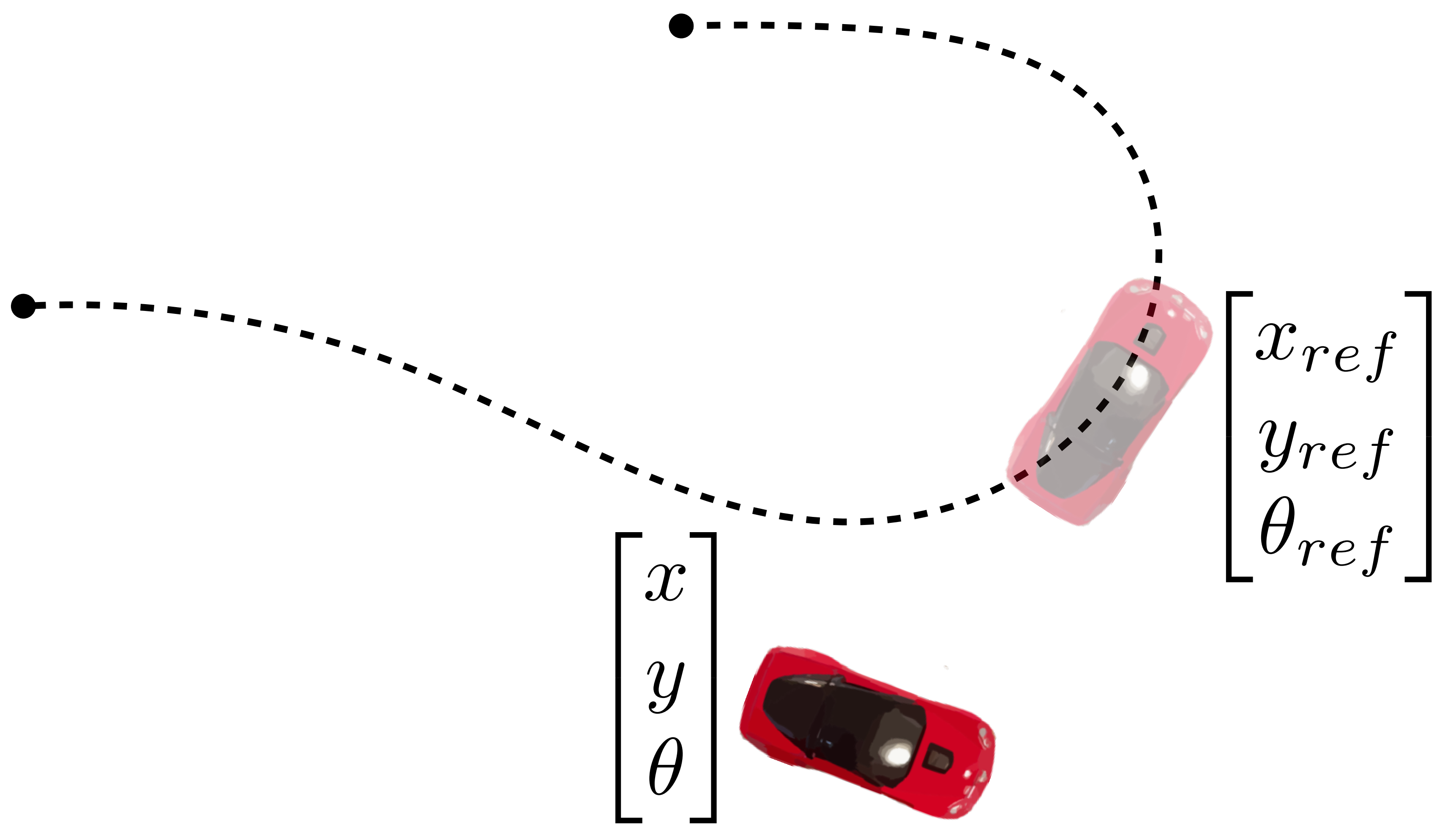
Step 1: Get a reference path

$$x(\tau), y(\tau), \theta(\tau), v(\tau)$$



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

Step 2: Pick a reference (desired) state



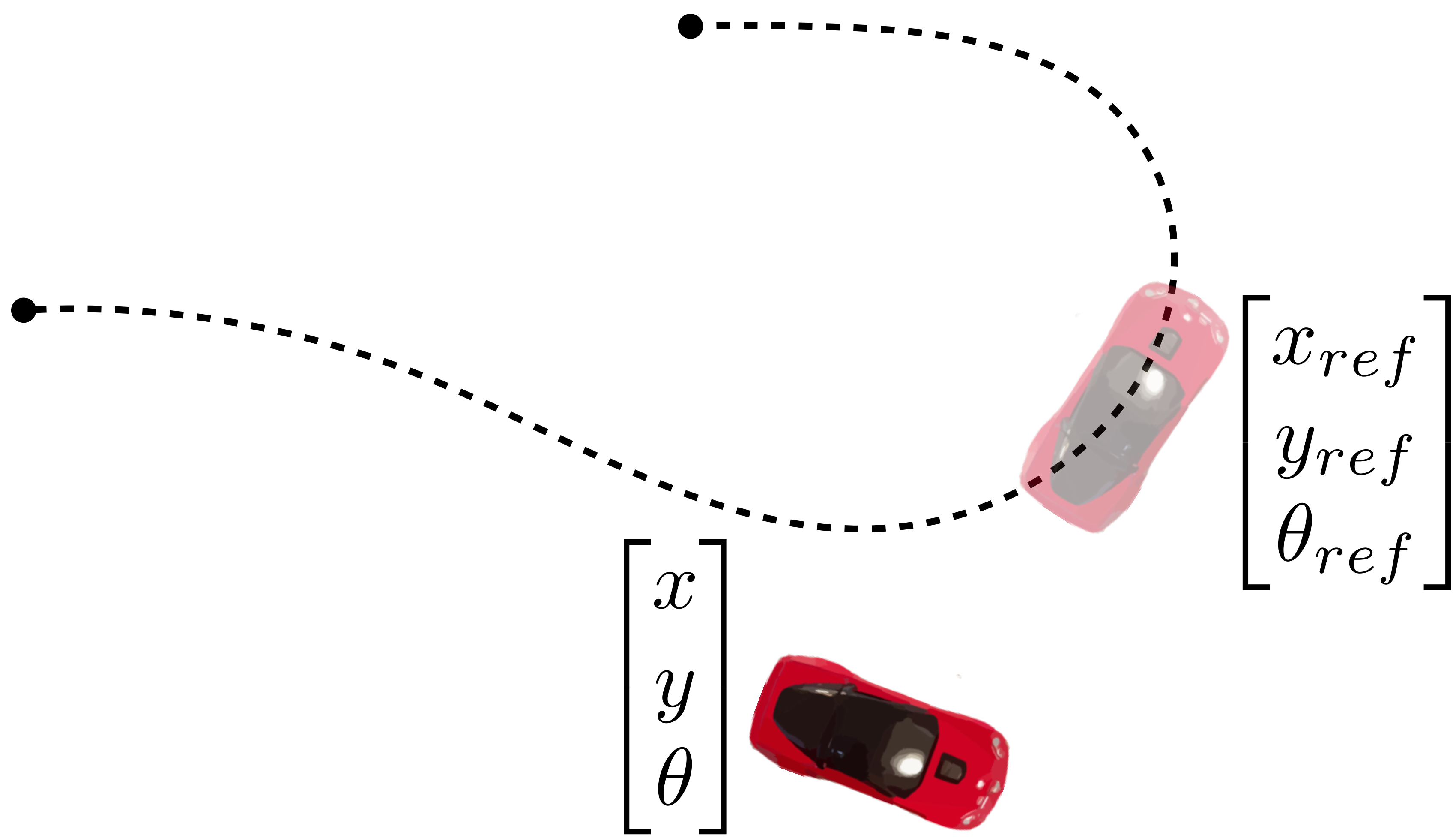
E.g. Pick nearest state / pick state L distance ahead

Step 3: Compute error to this state

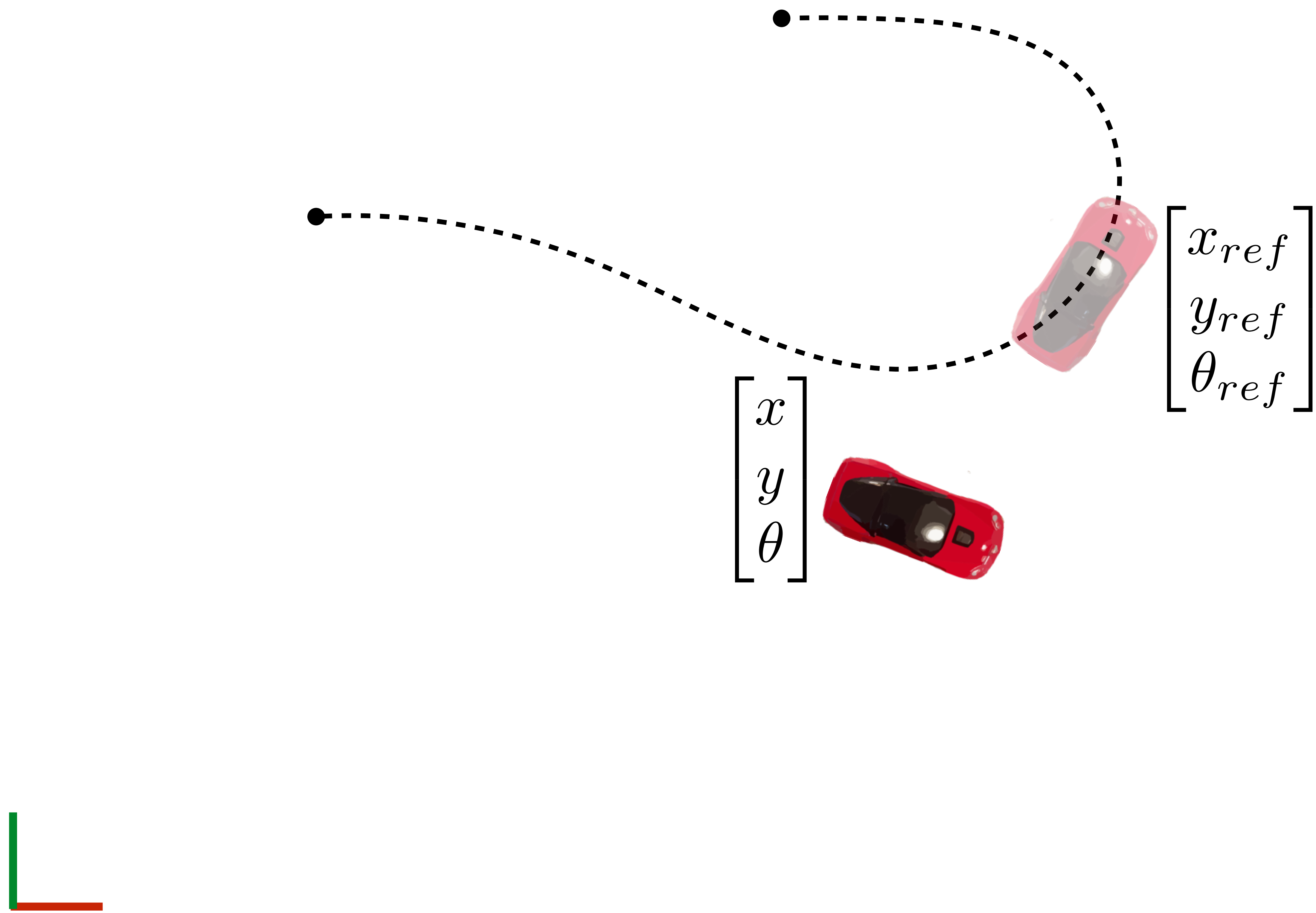
Error is simply the state of the car expressed in the frame of the reference (desired) state

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \Rightarrow \begin{bmatrix} e_{at} \\ e_{ct} \\ \theta_e \end{bmatrix}$$

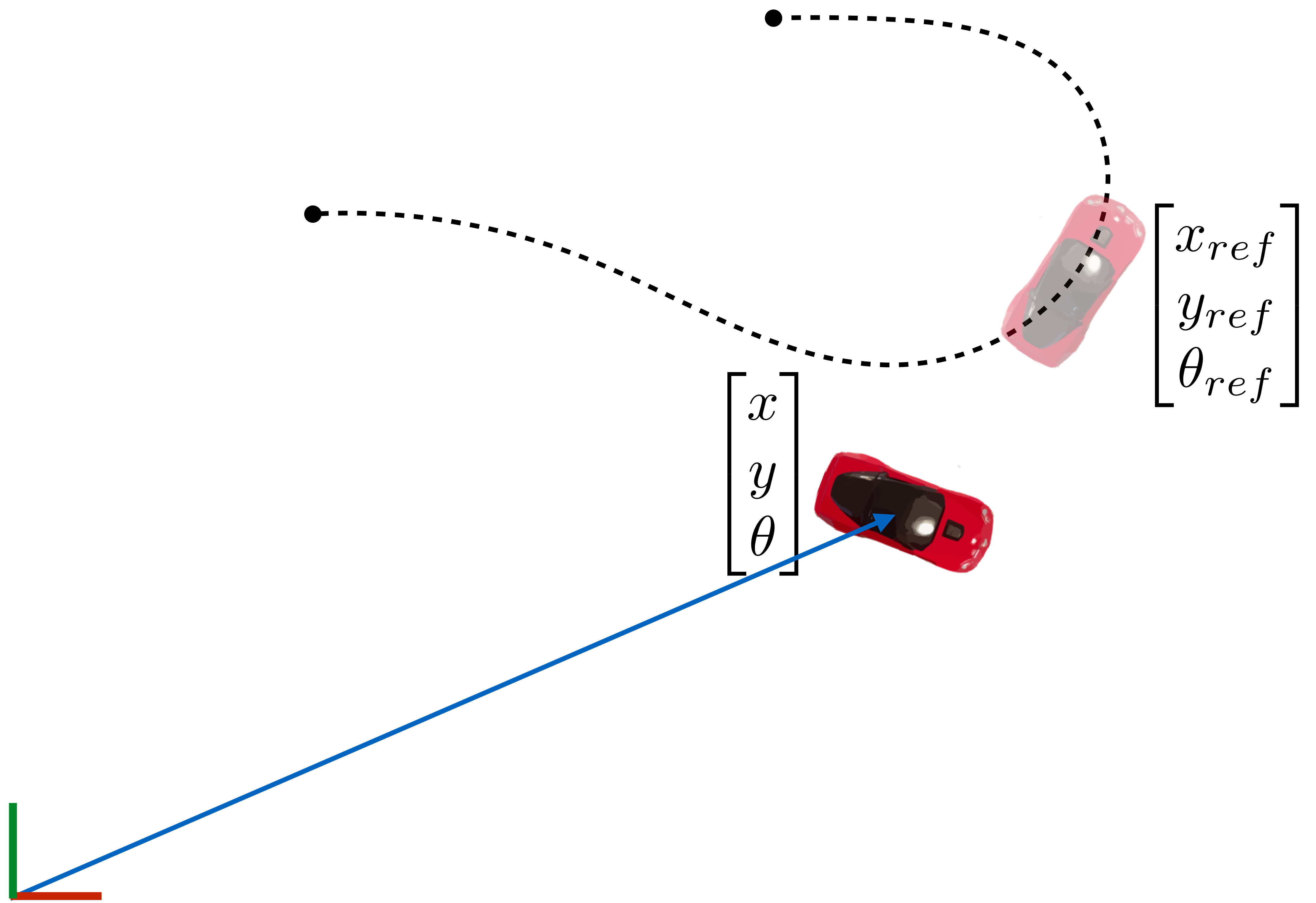
Step 3: Compute error to this state



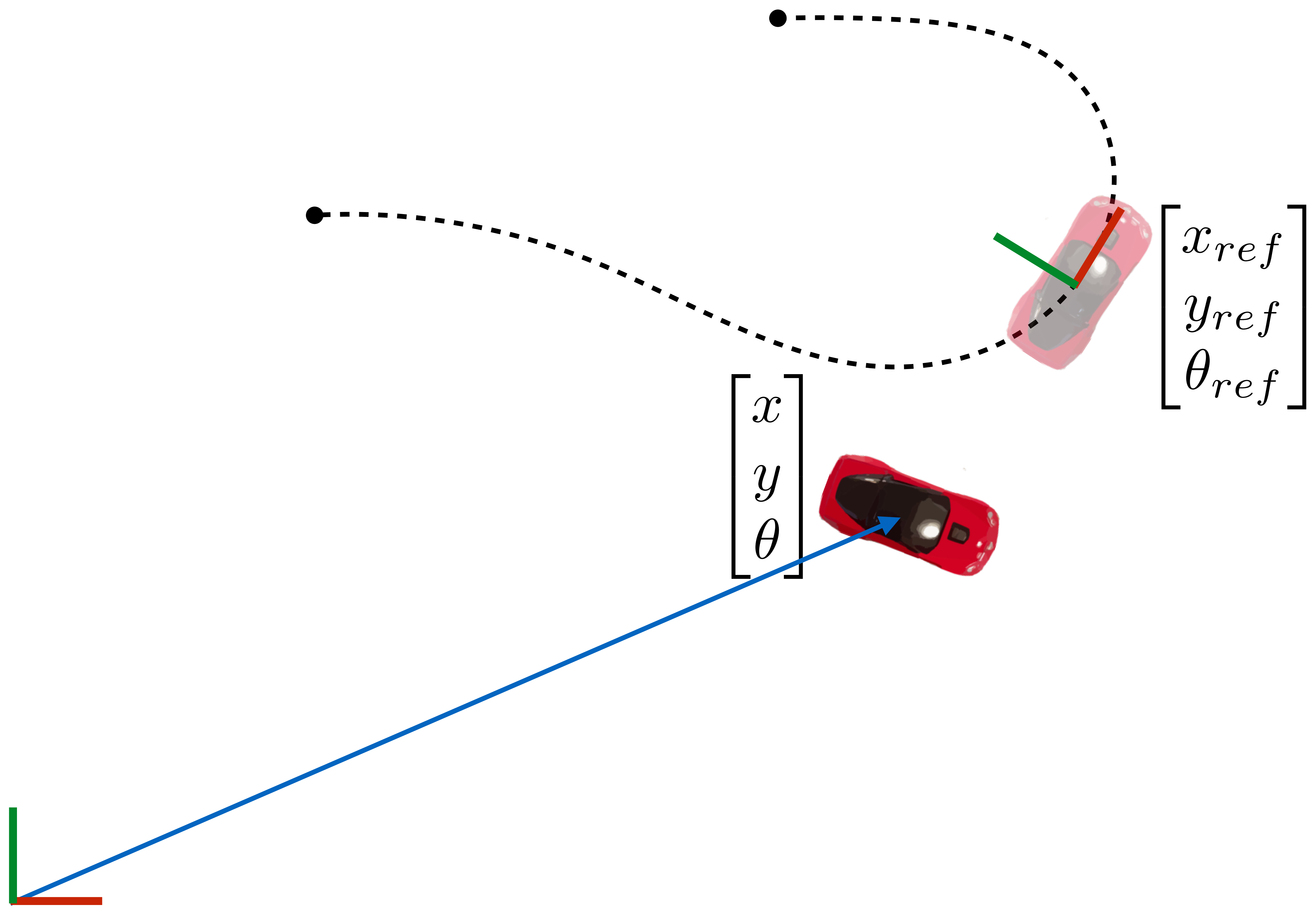
Step 3: Compute error to this state



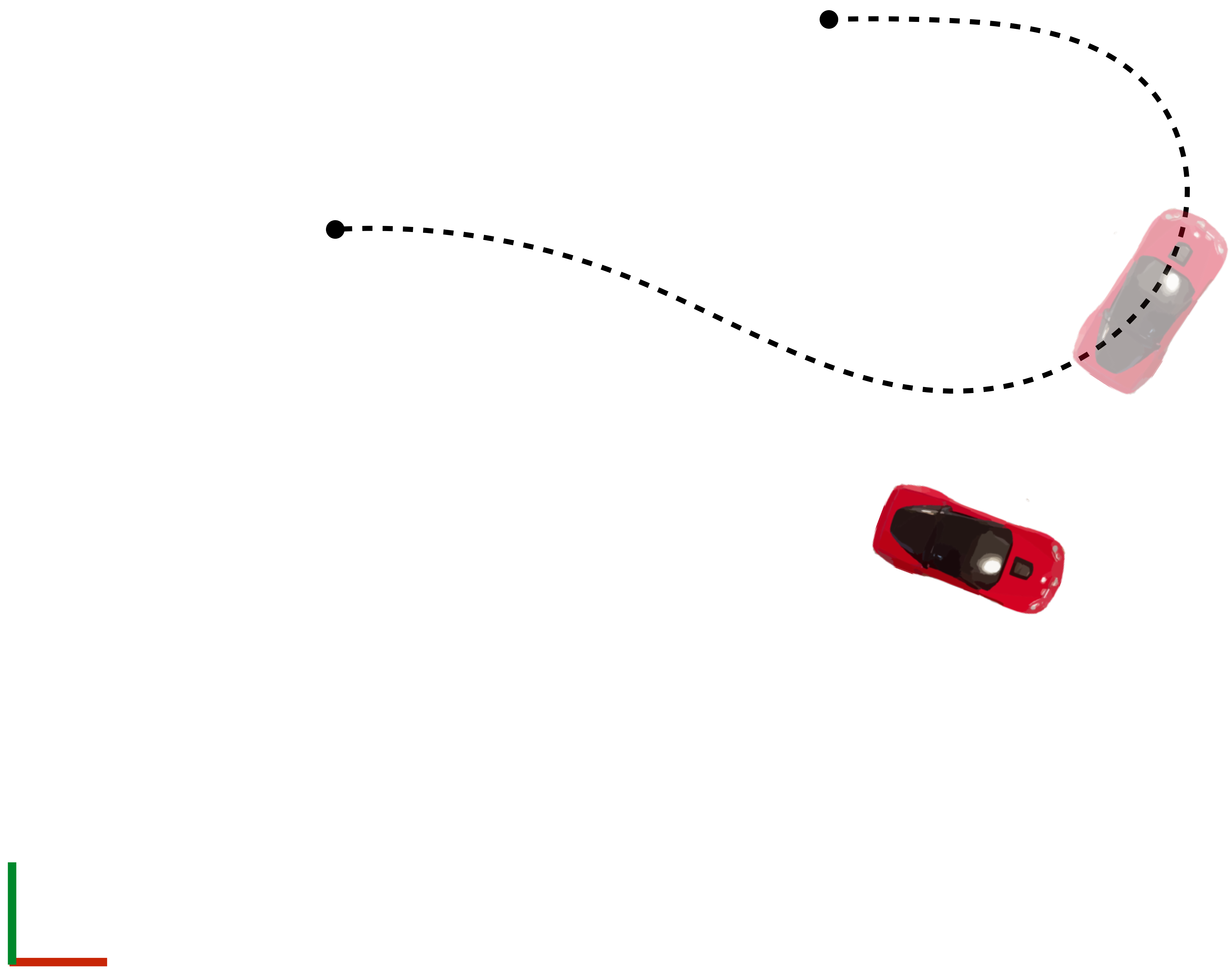
Step 3: Compute error to this state



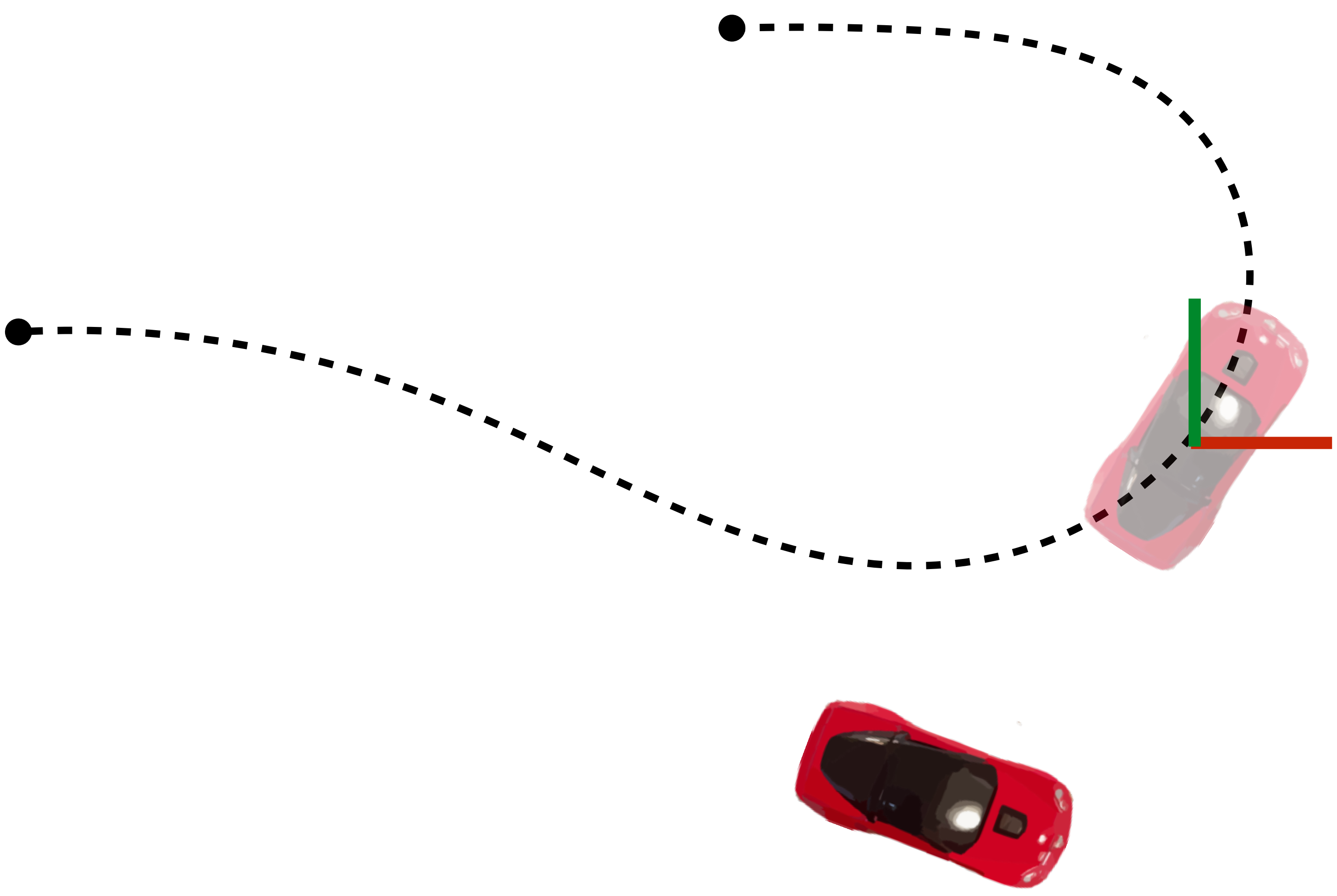
Step 3: Compute error to this state



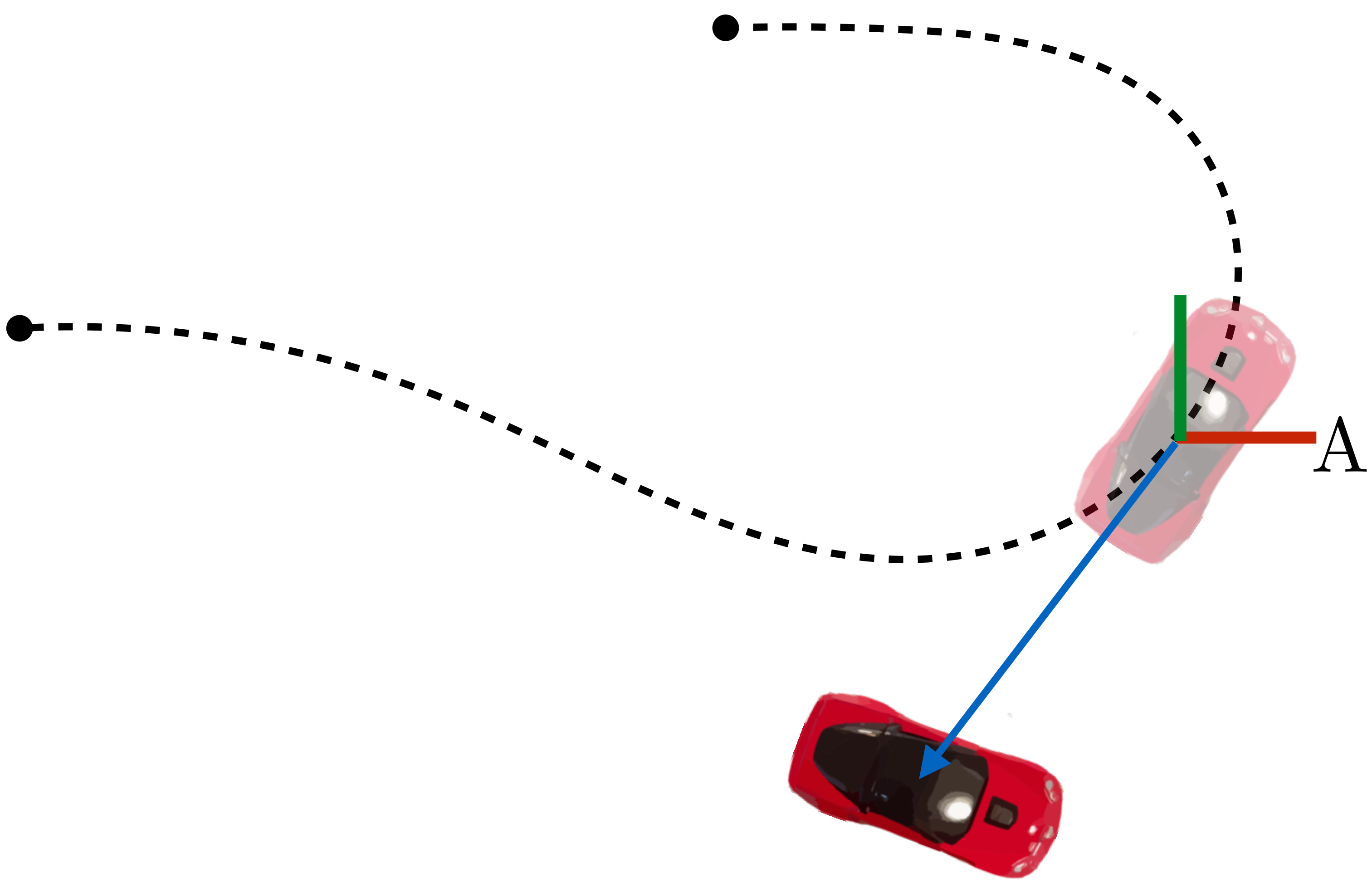
Step 3: Compute error to this state



Step 3: Compute error to this state



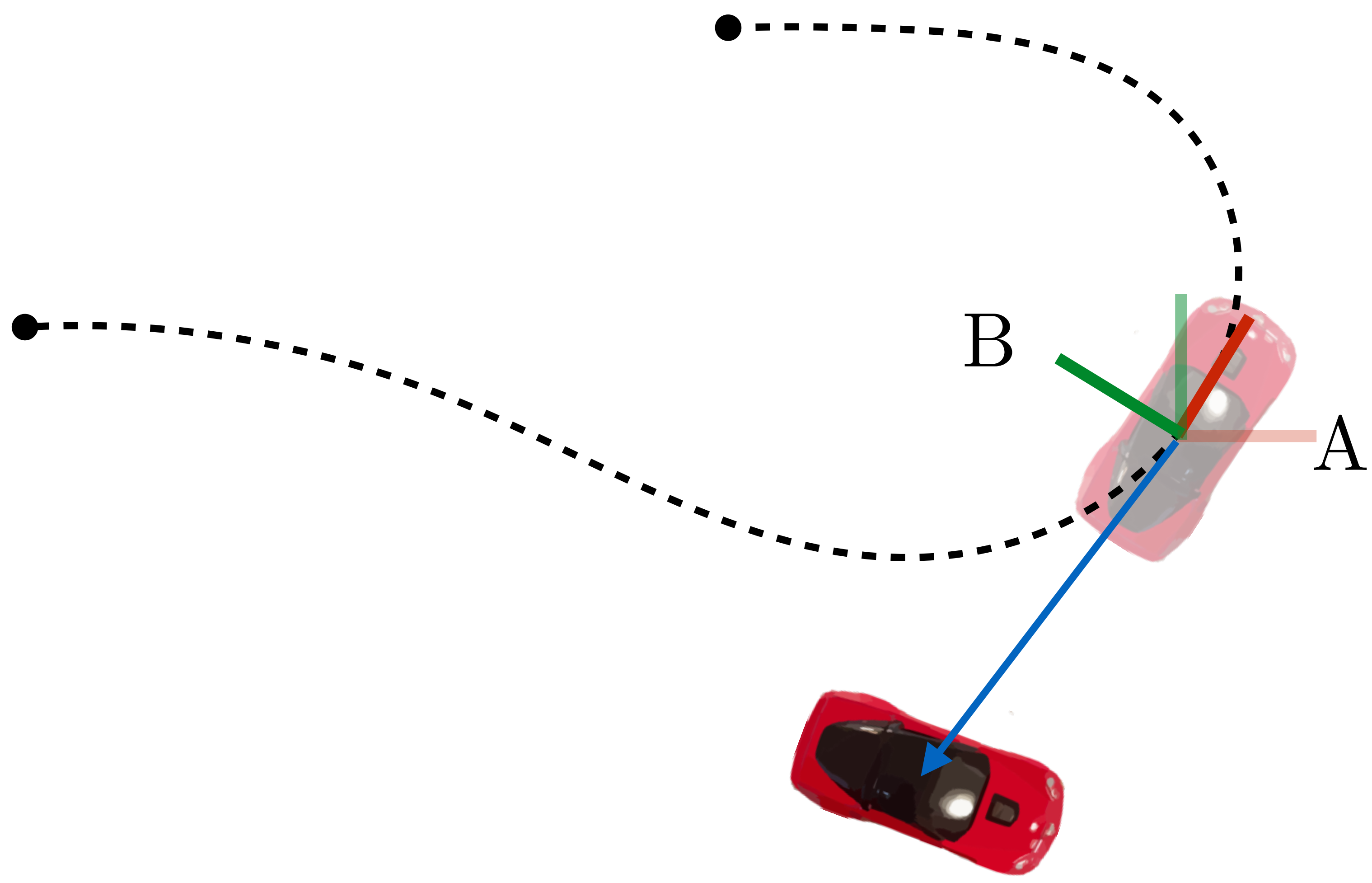
Step 3: Compute error to this state



Position in frame A

$$^{A}e = \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_{ref} \\ y_{ref} \end{bmatrix}$$

Step 3: Compute error to this state

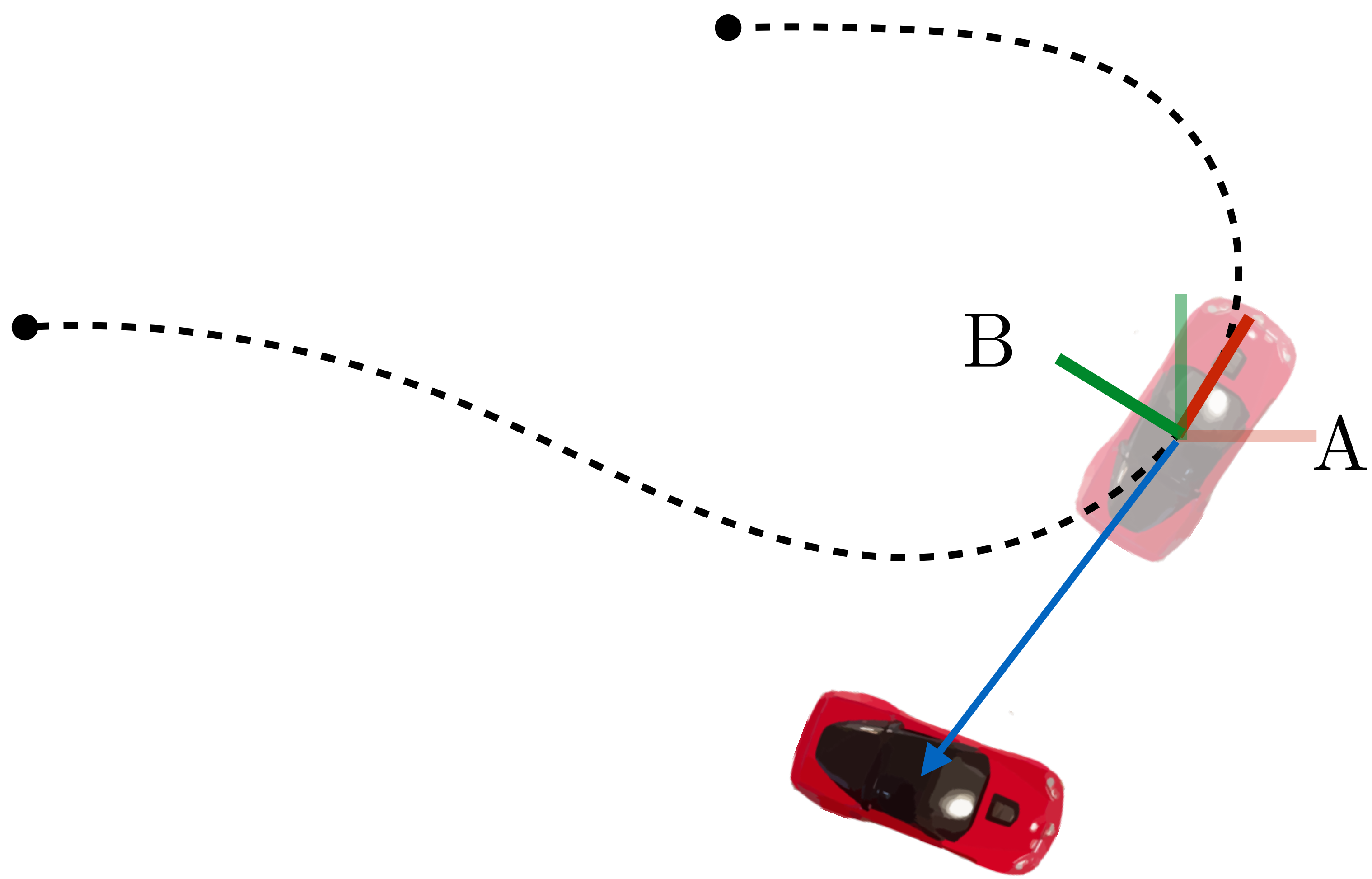


We want position in frame B

$$^{B}e = {}^{B}_{A}R \; {}^{A}e$$

(rotation of
A w.r.t B)

Step 3: Compute error to this state



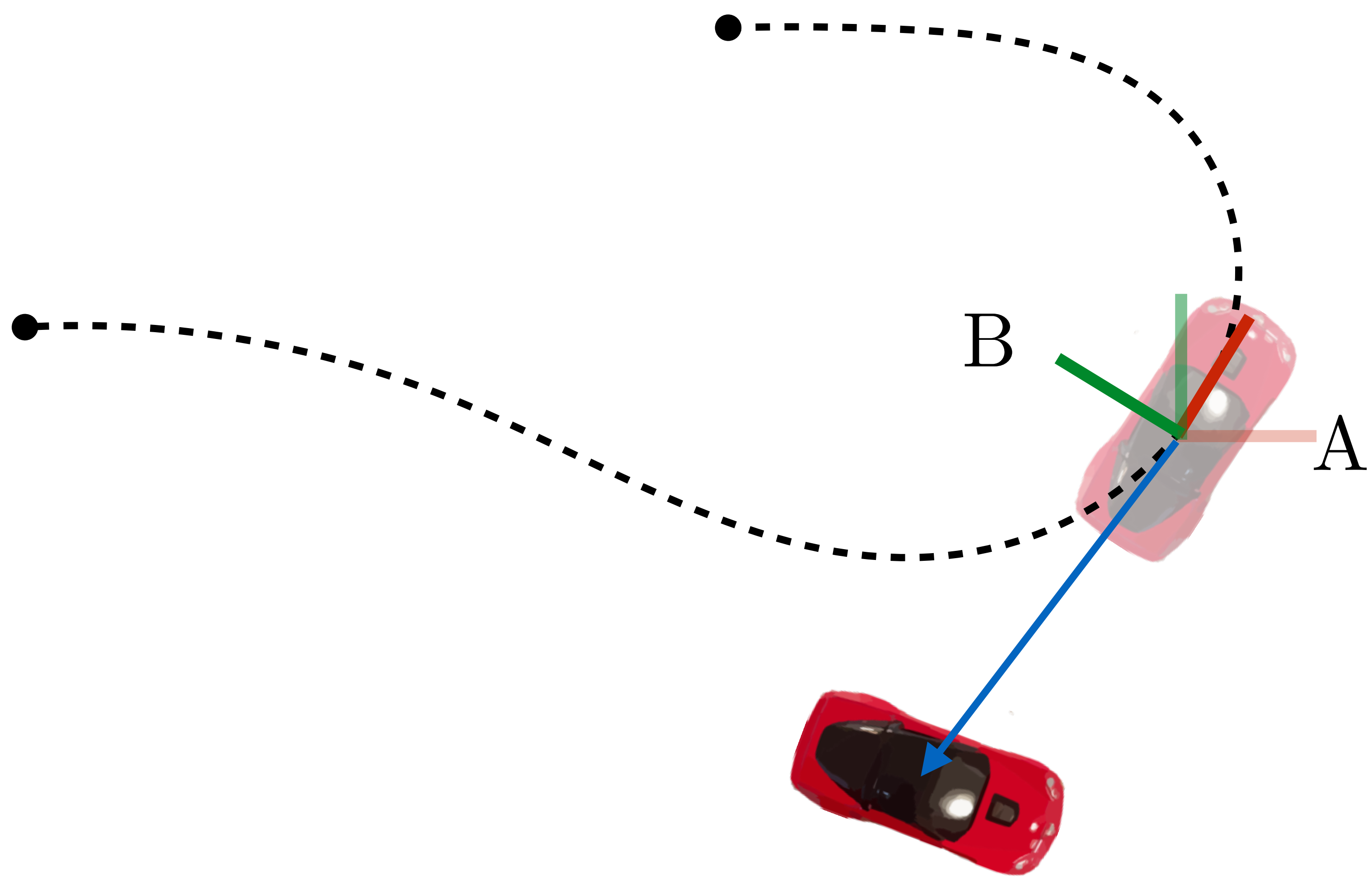
We want position in frame B

$$ {}^{B}e = {}^{B}_{A}R \; {}^{A}e = R(-\theta_{ref}) \left(\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_{ref} \\ y_{ref} \end{bmatrix} \right) $$

(rotation of A w.r.t B)

(rotation of A w.r.t B)

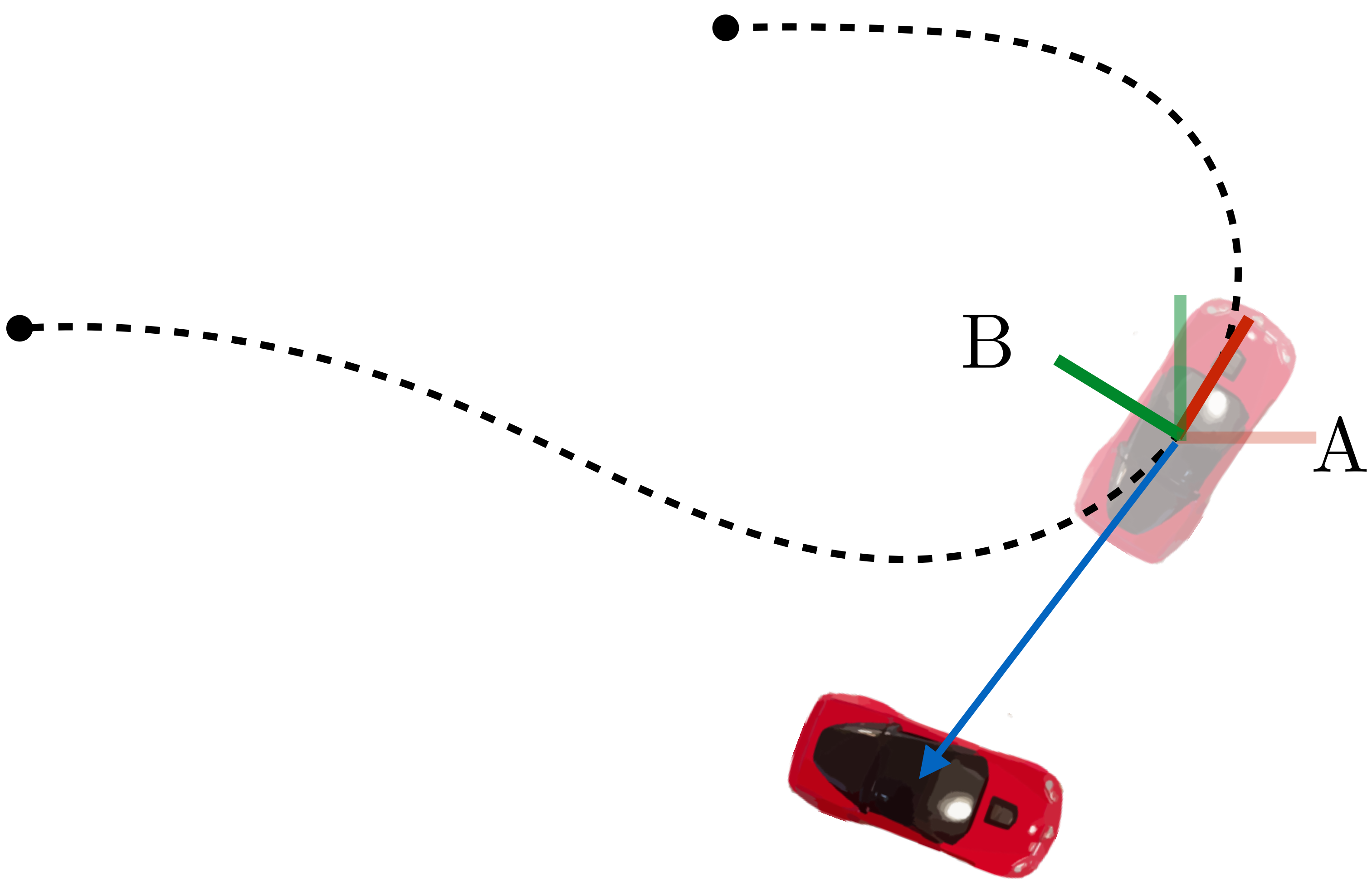
Step 3: Compute error to this state



We want position in frame B

$$^{B}e = \begin{bmatrix} e_{at} \\ e_{ct} \end{bmatrix} = \begin{bmatrix} \cos(\theta_{ref}) & \sin(\theta_{ref}) \\ -\sin(\theta_{ref}) & \cos(\theta_{ref}) \end{bmatrix} \left(\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_{ref} \\ y_{ref} \end{bmatrix} \right)$$

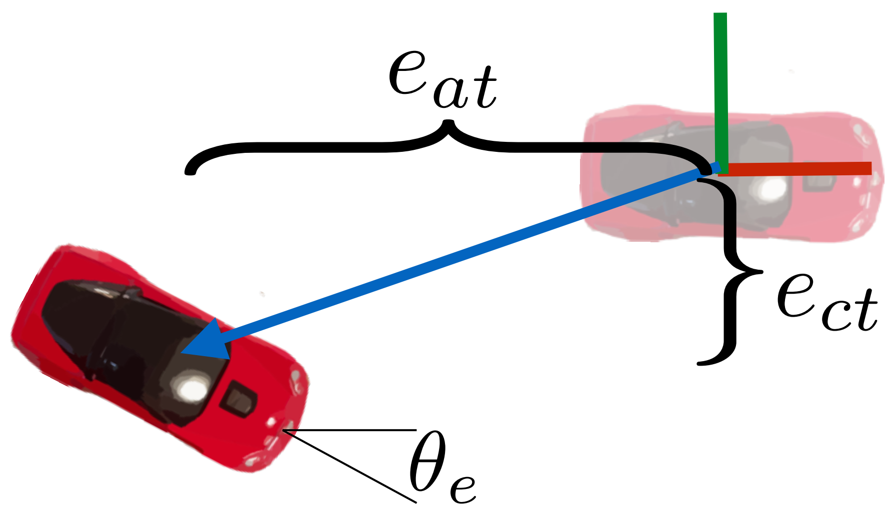
Step 3: Compute error to this state



We heading in frame B

$$\theta_e = \theta - \theta_{ref}$$

Step 3: Compute error to this state

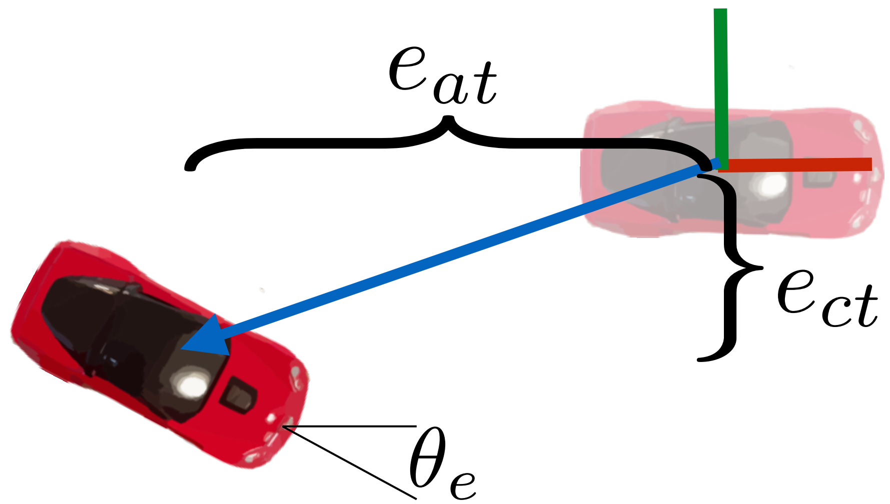


(Along-track) $e_{at} = \cos(\theta_{ref})(x - x_{ref}) + \sin(\theta_{ref})(y - y_{ref})$

(Cross-track) $e_{ct} = -\sin(\theta_{ref})(x - x_{ref}) + \cos(\theta_{ref})(y - y_{ref})$

(Heading) $\theta_e = \theta - \theta_{ref}$

Some things to note



1. We will only control steering angle; speed set to reference speed

2. Hence, no real control on along-track error. Ignore for now.

3. Some control laws will only minimize cross-track error, others both heading and cross-track error.

Step 4: Compute control law

Compute control action based on instantaneous error

$$u = K(e)$$

control error

Different laws have different trade-offs,
make different assumptions,
look at different errors

Different control laws

1. PID control

2. Pure-pursuit control

3. Lyapunov control

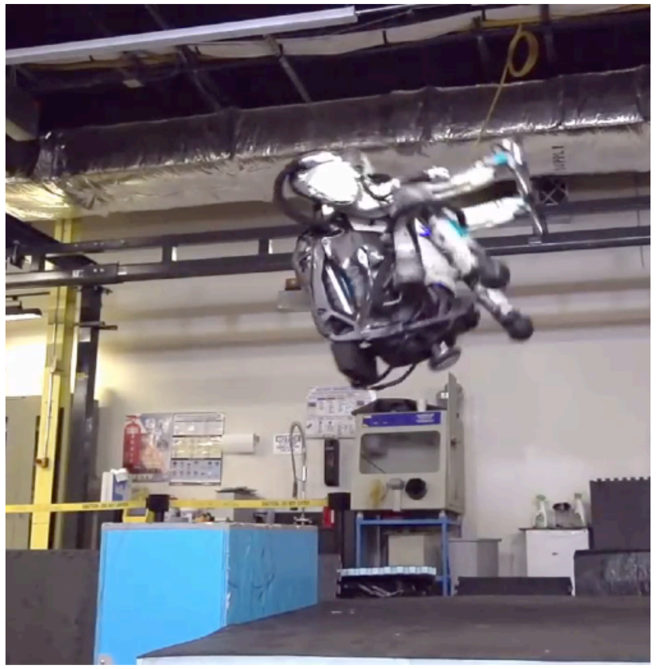
4. LQR

5. MPC

Proportional–integral–derivative (PID) controller

Used widely in industrial control from 1900s

Regulate temp, press, speed etc



Do not try this with PID!!!

PID control overview

Select a control law that tries to drive error to zero (and keep it there)

PID control overview

Select a control law that tries to drive error to zero (and keep it there)

$$u = -\left(K_p e_{ct} + K_i \int e_{ct}(t)dt + K_d \dot{e}_{ct} \right)$$

Proportional (current) | Integral (past) | Derivative (future)

Some intuition ...

$$u = -\left(K_p e_{ct} + K_i \int e_{ct}(t)dt + K_d \dot{e}_{ct}\right)$$

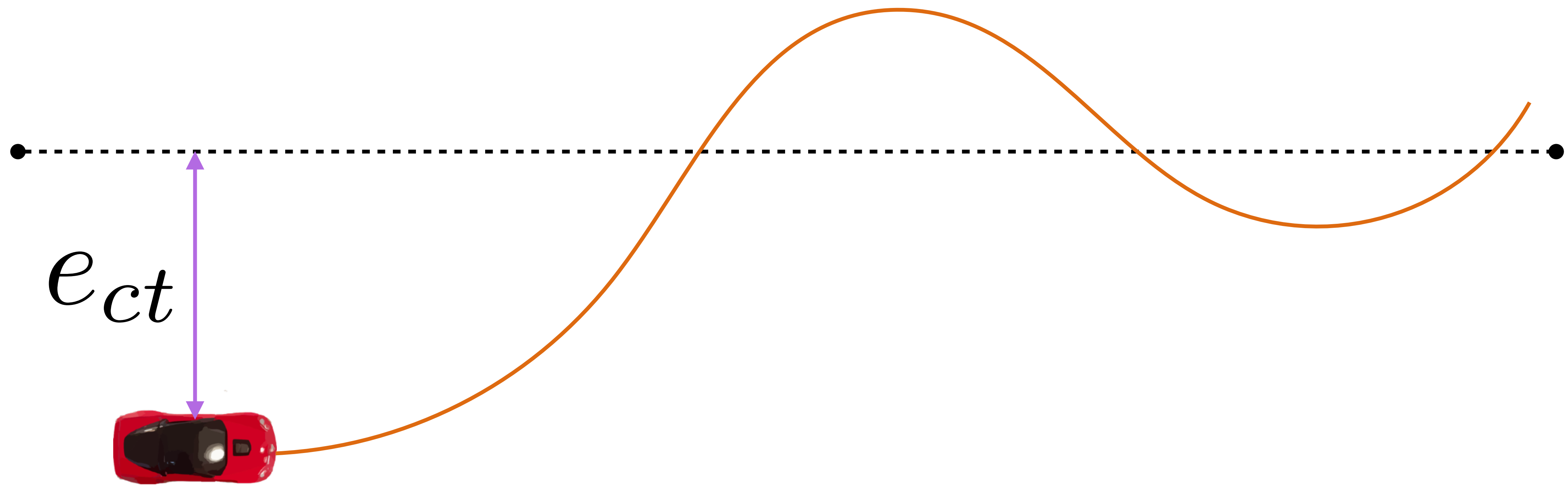
Proportional	Integral	Derivative
(current)	(past)	(future)

Proportional - get rid of the current error!

Integral - if I am accumulating error, try harder!

Derivative - if I am going to overshoot, slow down!

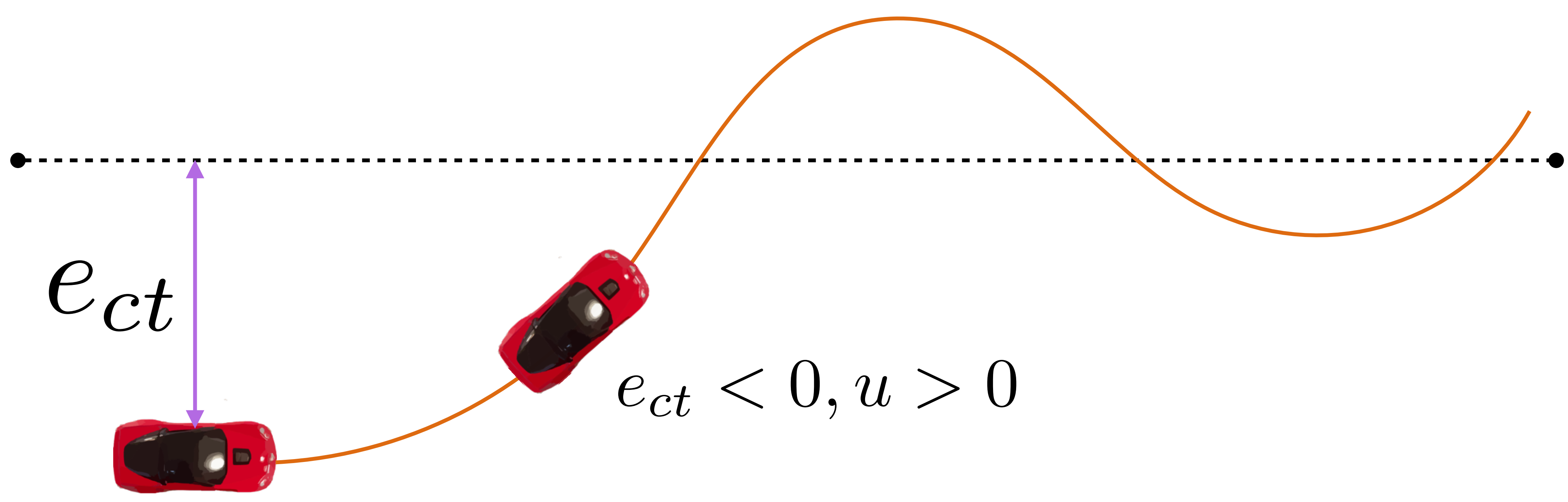
Proportional control



$$u = -K_p e_{ct}$$

(Gain)

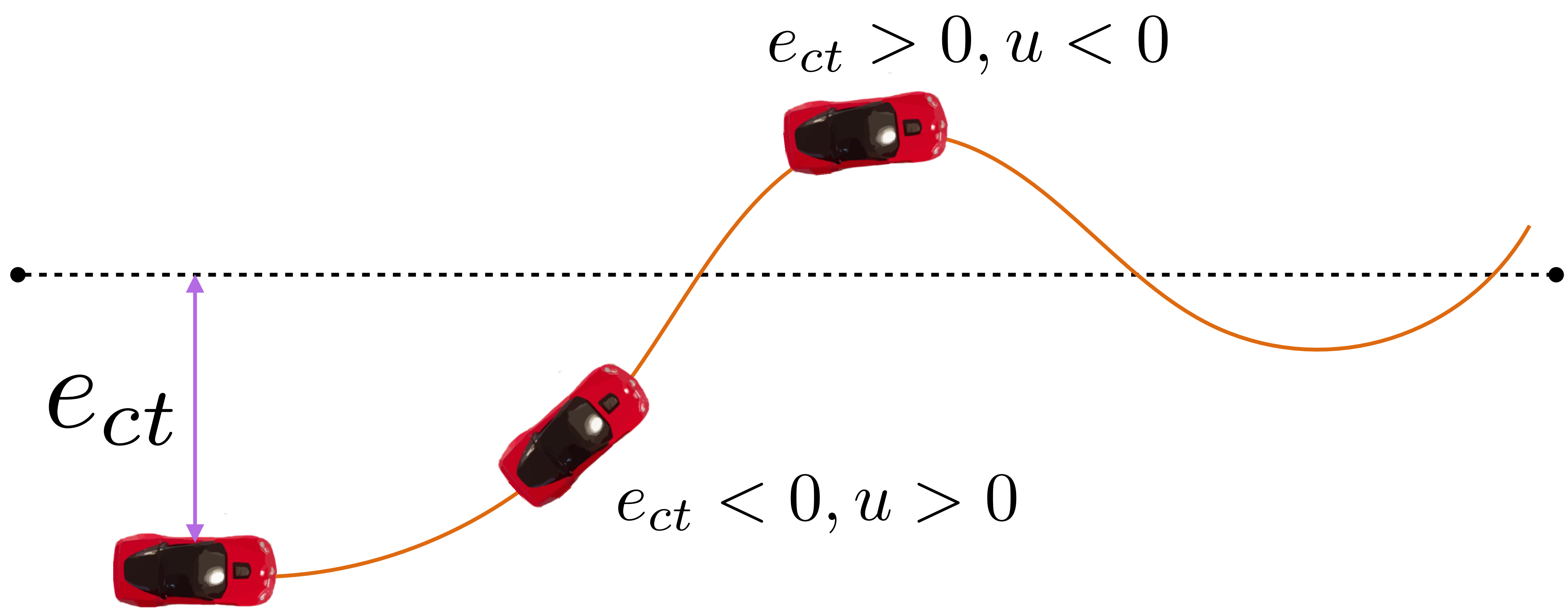
Proportional control



$$u = -K_p e_{ct}$$

(Gain)

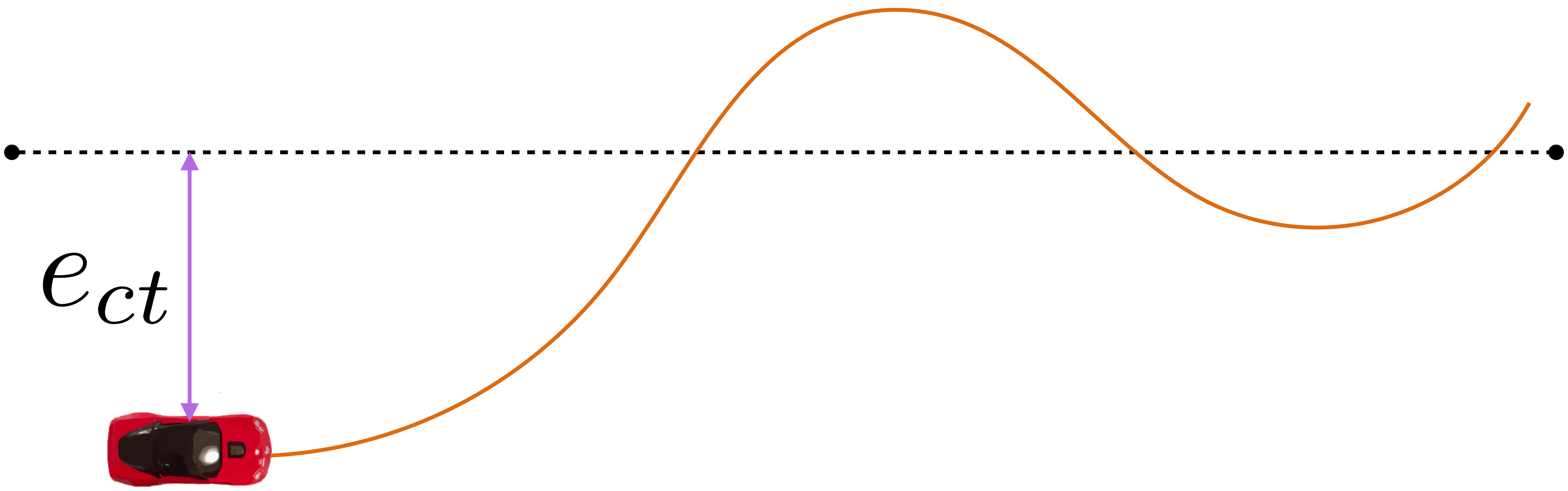
Proportional control



$$u = -K_p e_{ct}$$

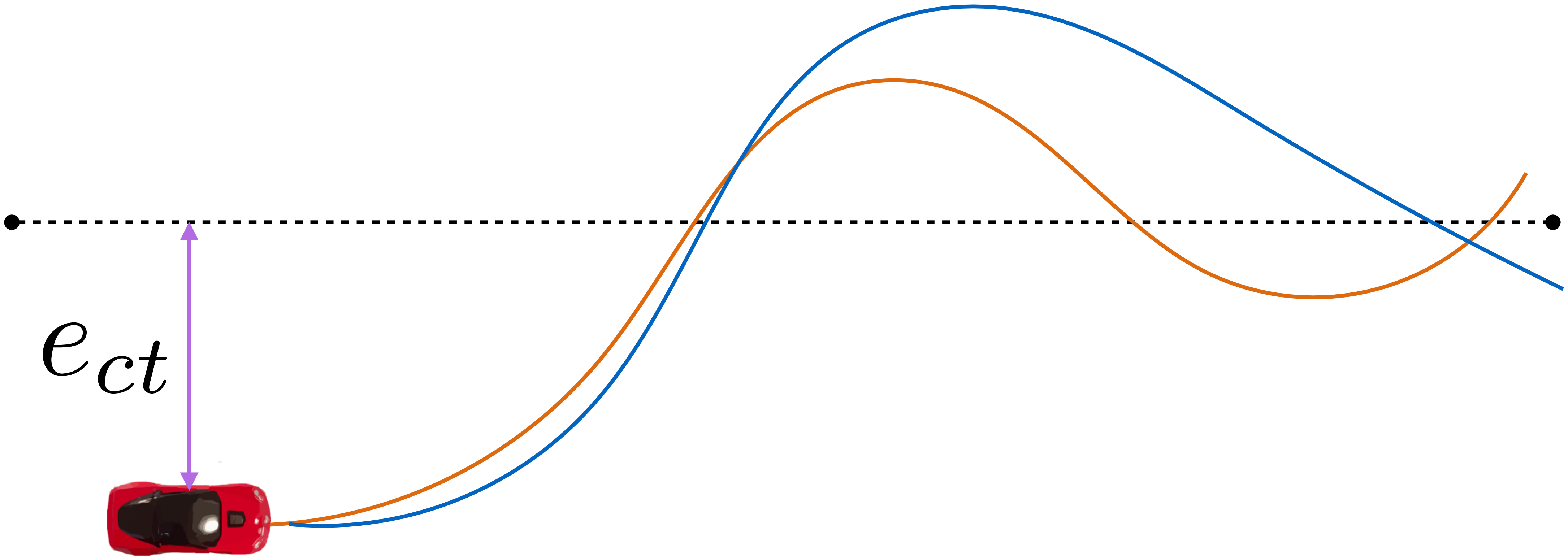
(Gain)

The proportional gain matters!



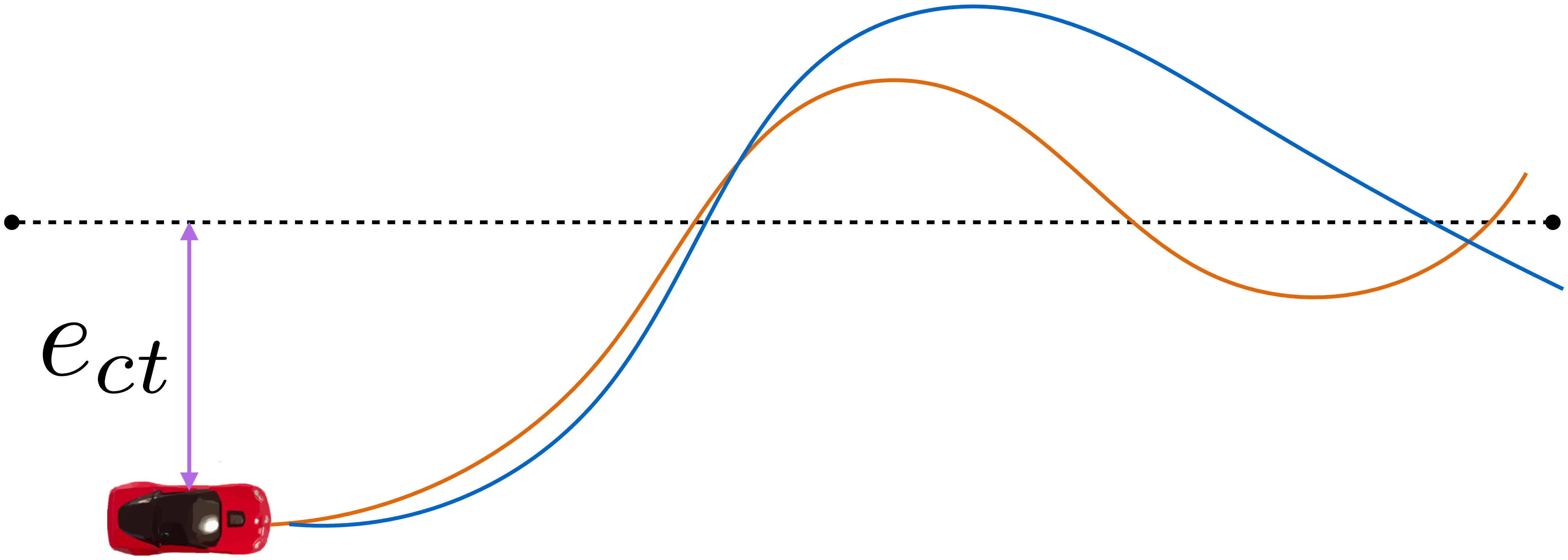
What happens when gain is low?

The proportional gain matters!



What happens when gain is low?

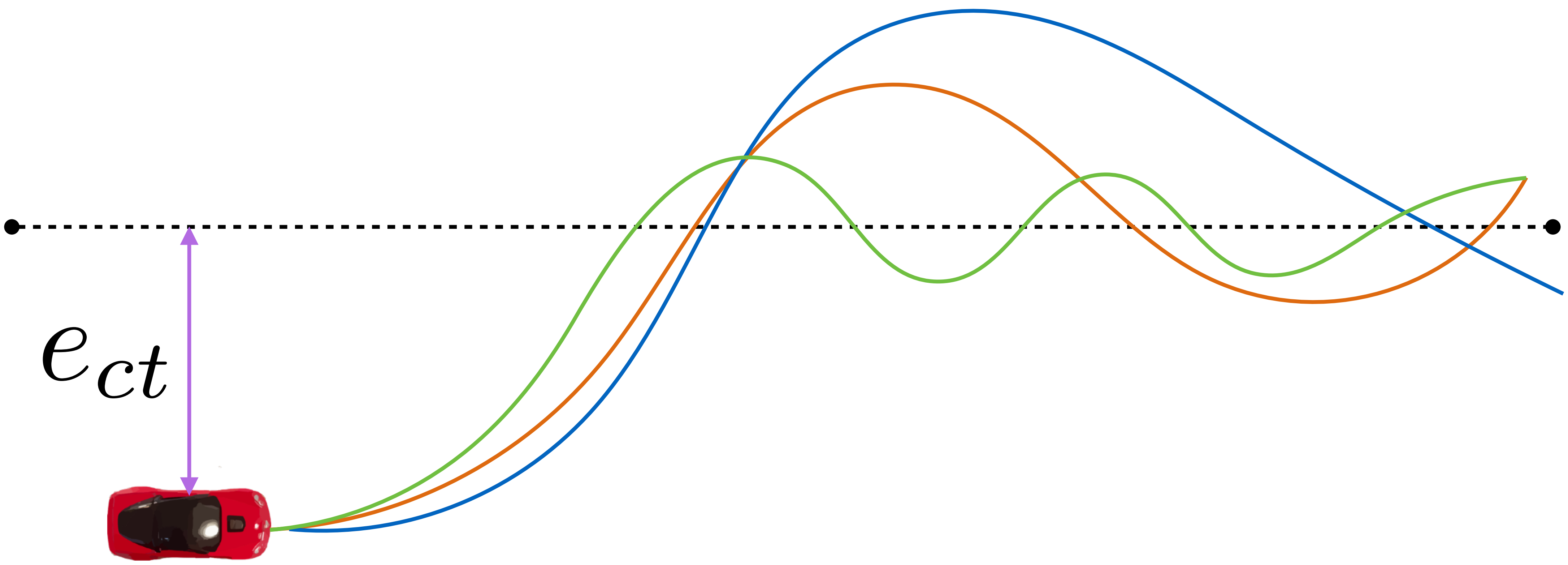
The proportional gain matters!



What happens when gain is low?

What happens when gain is high?

The proportional gain matters!

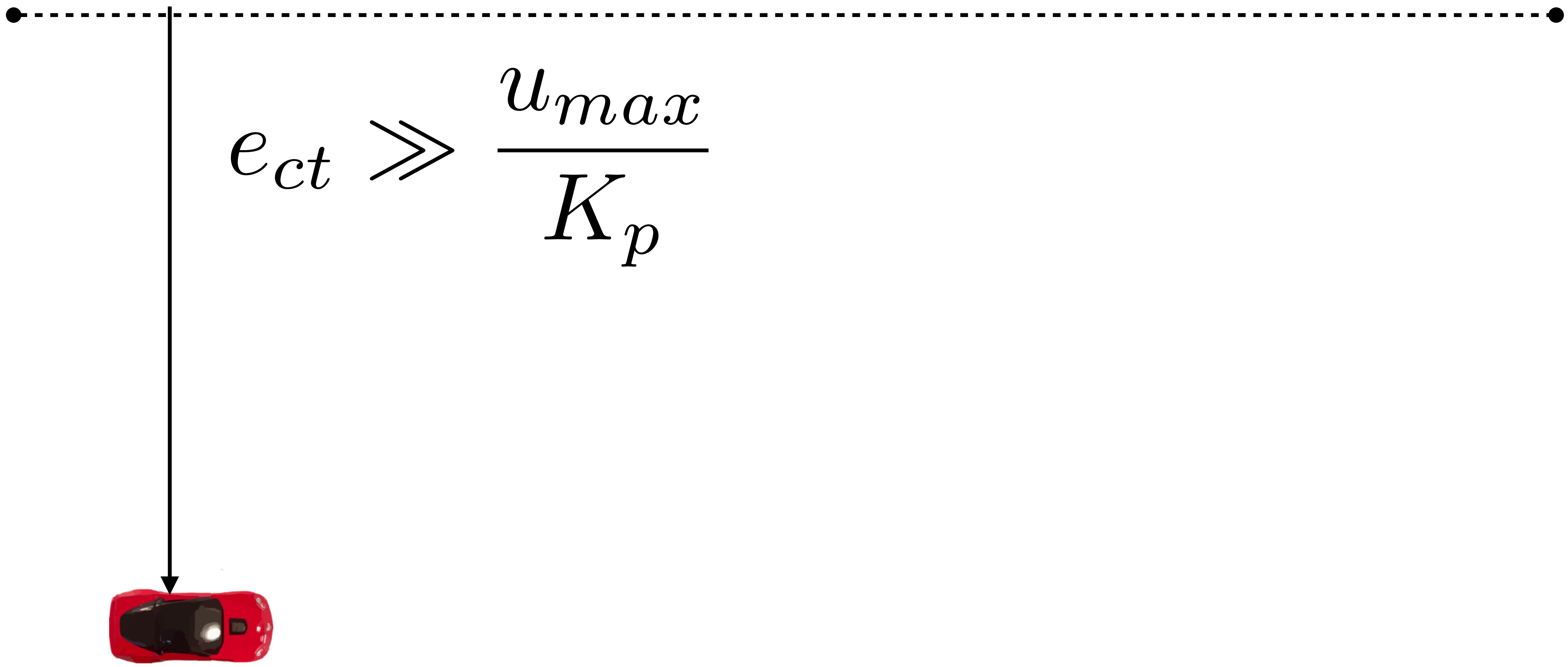


What happens when gain is low?

What happens when gain is high?

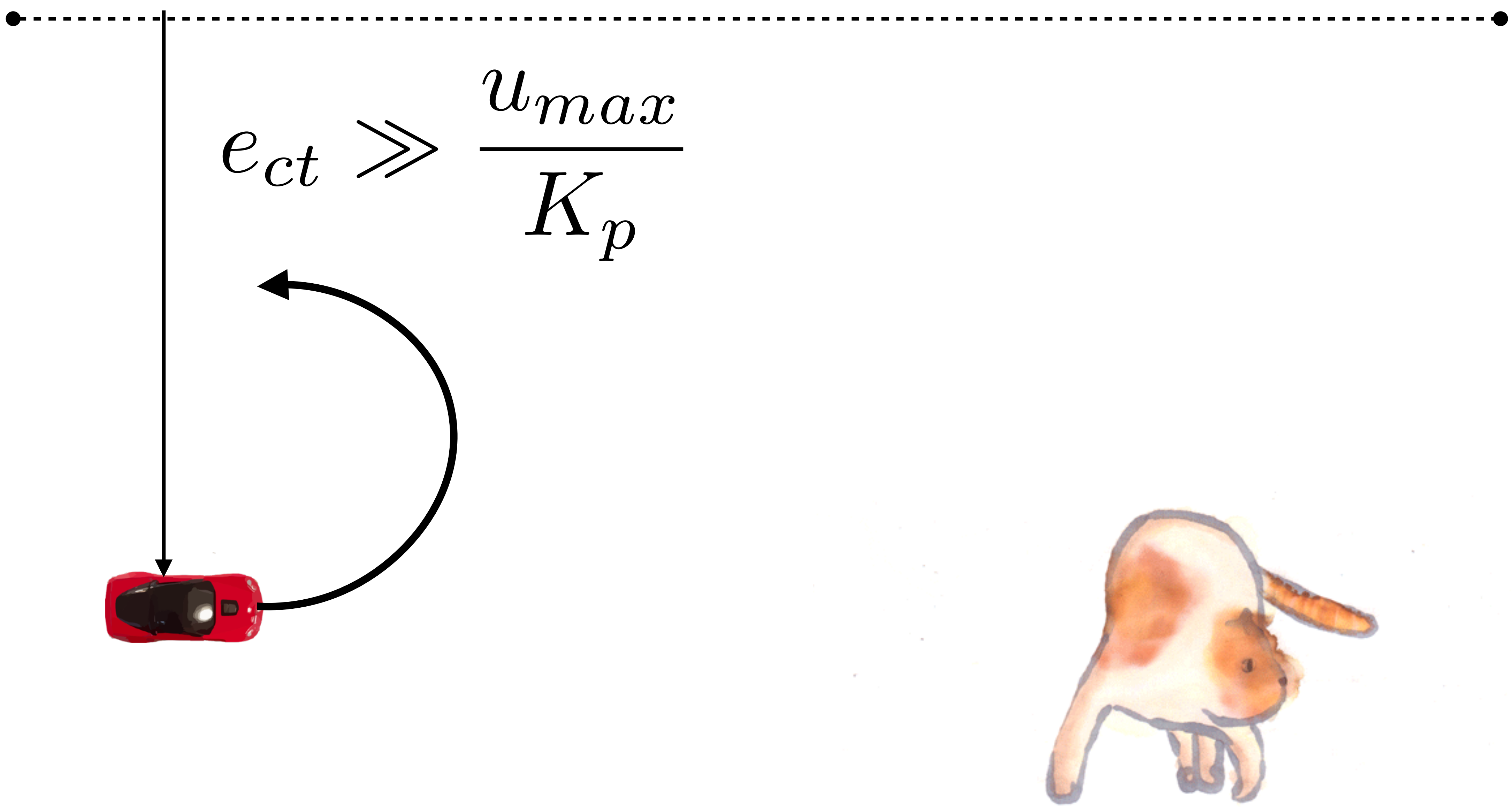
Proportional term

What happens when gain is too high?



Proportional term

What happens when gain is too high?

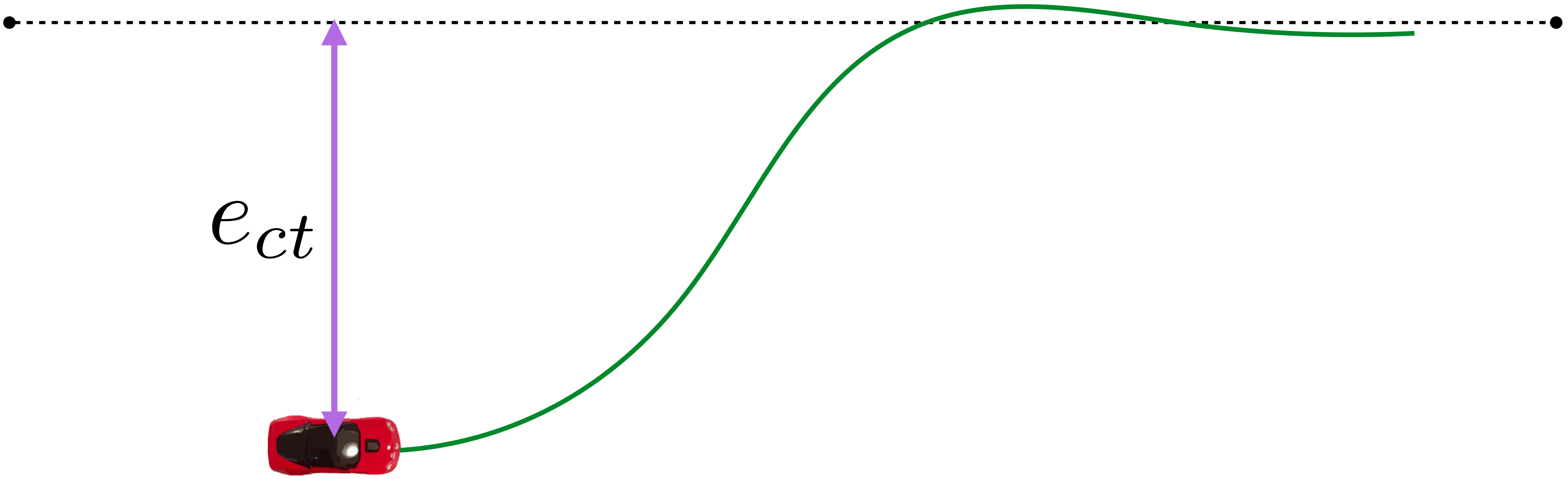


$$e_{ct} \gg \frac{u_{max}}{K_p}$$

Proportional derivative control

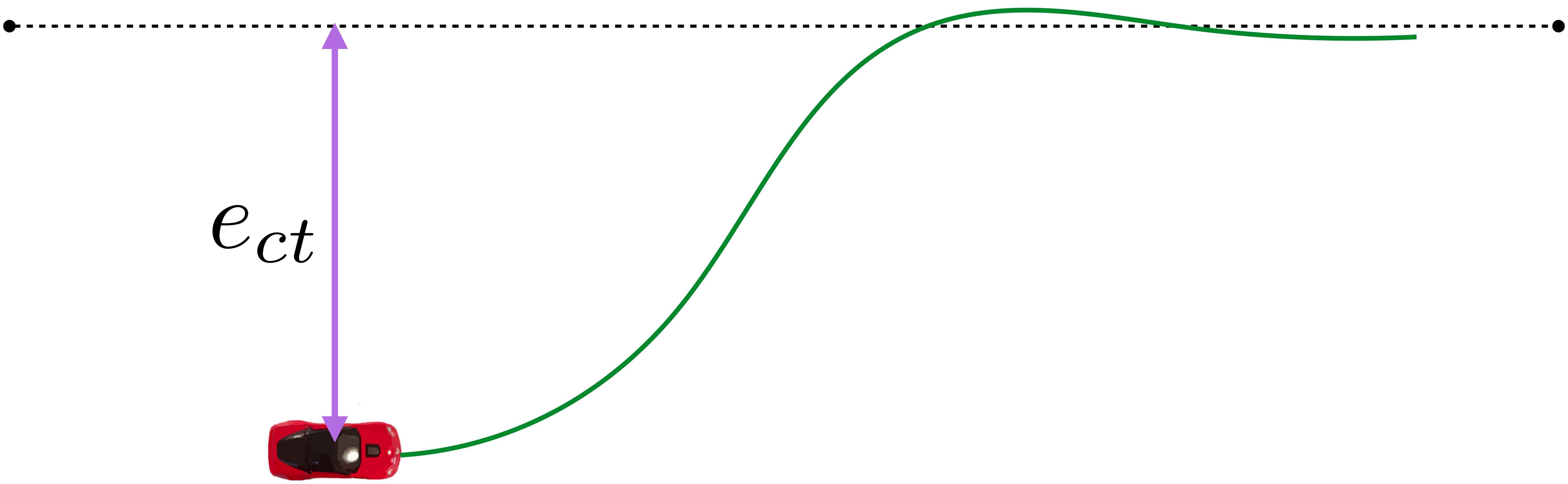
$$u = -\left(K_p e_{ct} + K_d \dot{e}_{ct}\right)$$

Proportional derivative control



$$u = -\left(K_p e_{ct} + K_d \dot{e}_{ct}\right)$$

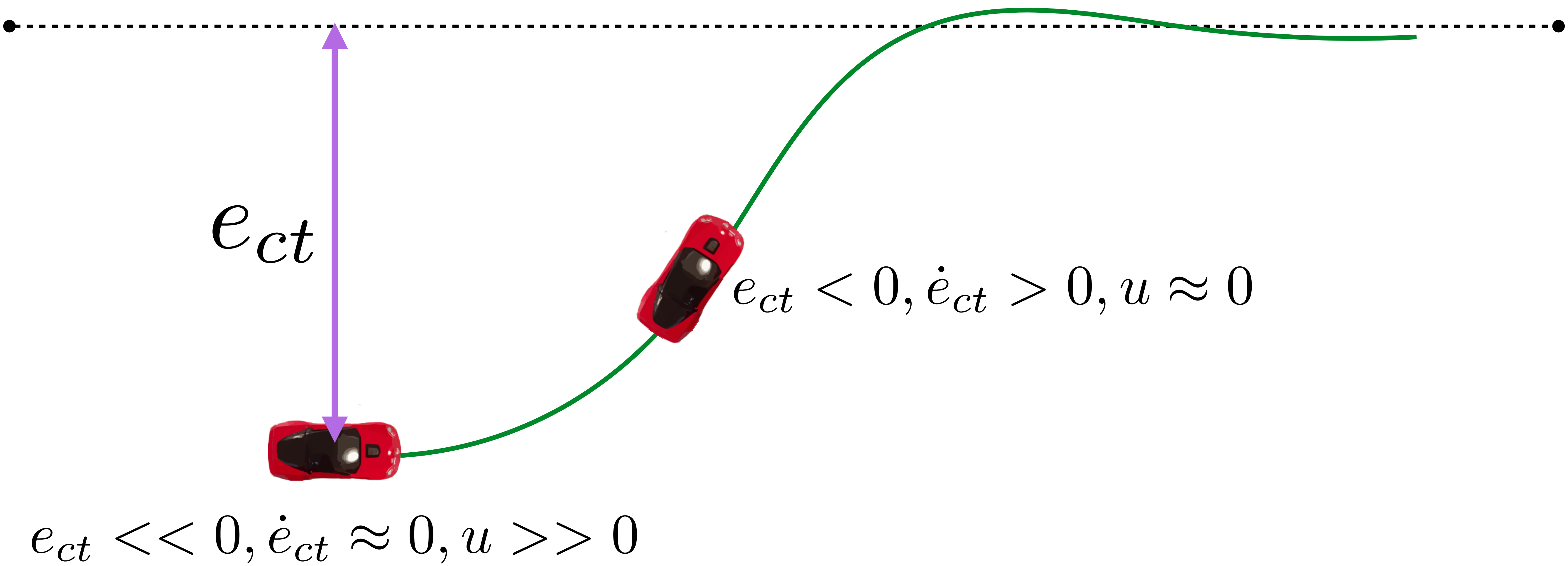
Proportional derivative control



$e_{ct} << 0, \dot{e}_{ct} \approx 0, u >> 0$

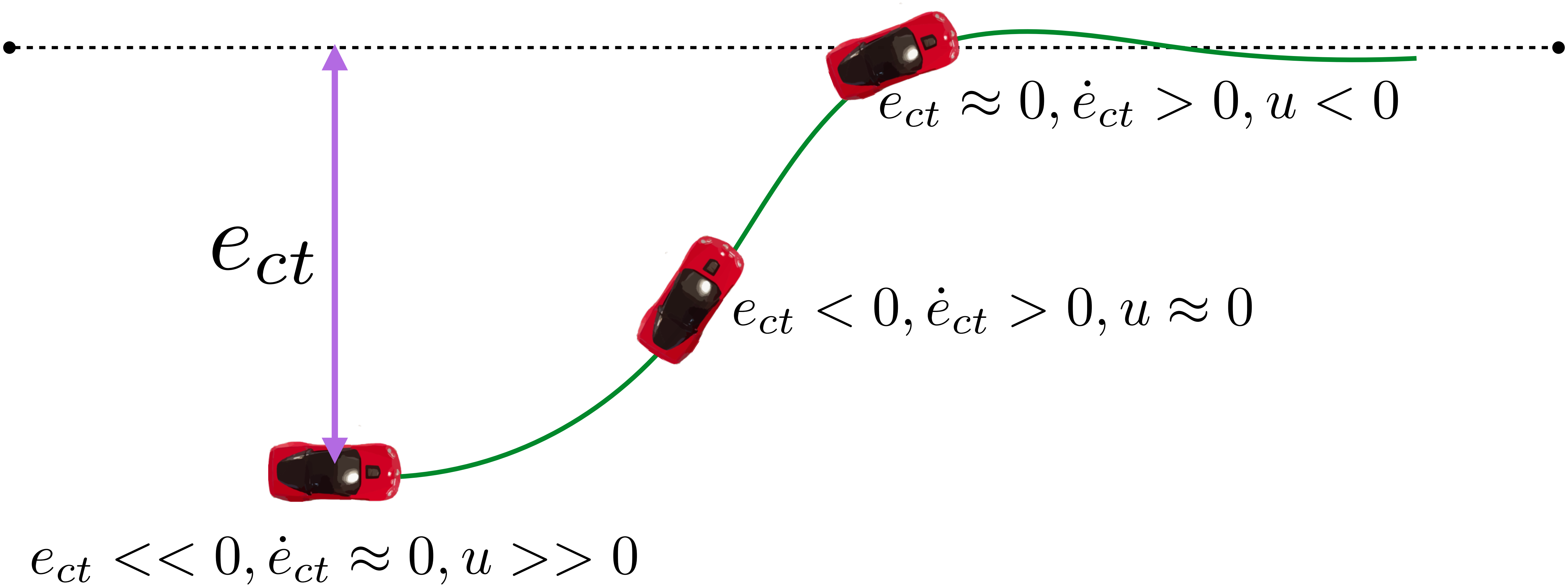
$$u = -(K_p e_{ct} + K_d \dot{e}_{ct})$$

Proportional derivative control



$$u = -(K_p e_{ct} + K_d \dot{e}_{ct})$$

Proportional derivative control



$$u = -(K_p e_{ct} + K_d \dot{e_{ct}})$$

How do you evaluate the derivative term?

How do you evaluate the derivative term?

Terrible way: Numerically differentiate error. Why is this a bad idea?

How do you evaluate the derivative term?

Terrible way: Numerically differentiate error. Why is this a bad idea?

Smart way: Analytically compute the derivative of the cross track error

How do you evaluate the derivative term?

Terrible way: Numerically differentiate error. Why is this a bad idea?

Smart way: Analytically compute the derivative of the cross track error

$$e_{ct} = -\sin(\theta_{ref})(x - x_{ref}) + \cos(\theta_{ref})(y - y_{ref})$$

How do you evaluate the derivative term?

Terrible way: Numerically differentiate error. Why is this a bad idea?

Smart way: Analytically compute the derivative of the cross track error

$$e_{ct} = -\sin(\theta_{ref})(x - x_{ref}) + \cos(\theta_{ref})(y - y_{ref})$$

$$\begin{aligned}\dot{e}_{ct} &= -\sin(\theta_{ref})\dot{x} + \cos(\theta_{ref})\dot{y} \\ &= -\sin(\theta_{ref})V\cos(\theta) + \cos(\theta_{ref})V\sin(\theta) \\ &= V\sin(\theta - \theta_{ref}) = V\sin(\theta_e)\end{aligned}$$

How do you evaluate the derivative term?

Terrible way: Numerically differentiate error. Why is this a bad idea?

Smart way: Analytically compute the derivative of the cross track error

$$e_{ct} = -\sin(\theta_{ref})(x - x_{ref}) + \cos(\theta_{ref})(y - y_{ref})$$

$$\begin{aligned}\dot{e}_{ct} &= -\sin(\theta_{ref})\dot{x} + \cos(\theta_{ref})\dot{y} \\ &= -\sin(\theta_{ref})V\cos(\theta) + \cos(\theta_{ref})V\sin(\theta) \\ &= V\sin(\theta - \theta_{ref}) = V\sin(\theta_e)\end{aligned}$$

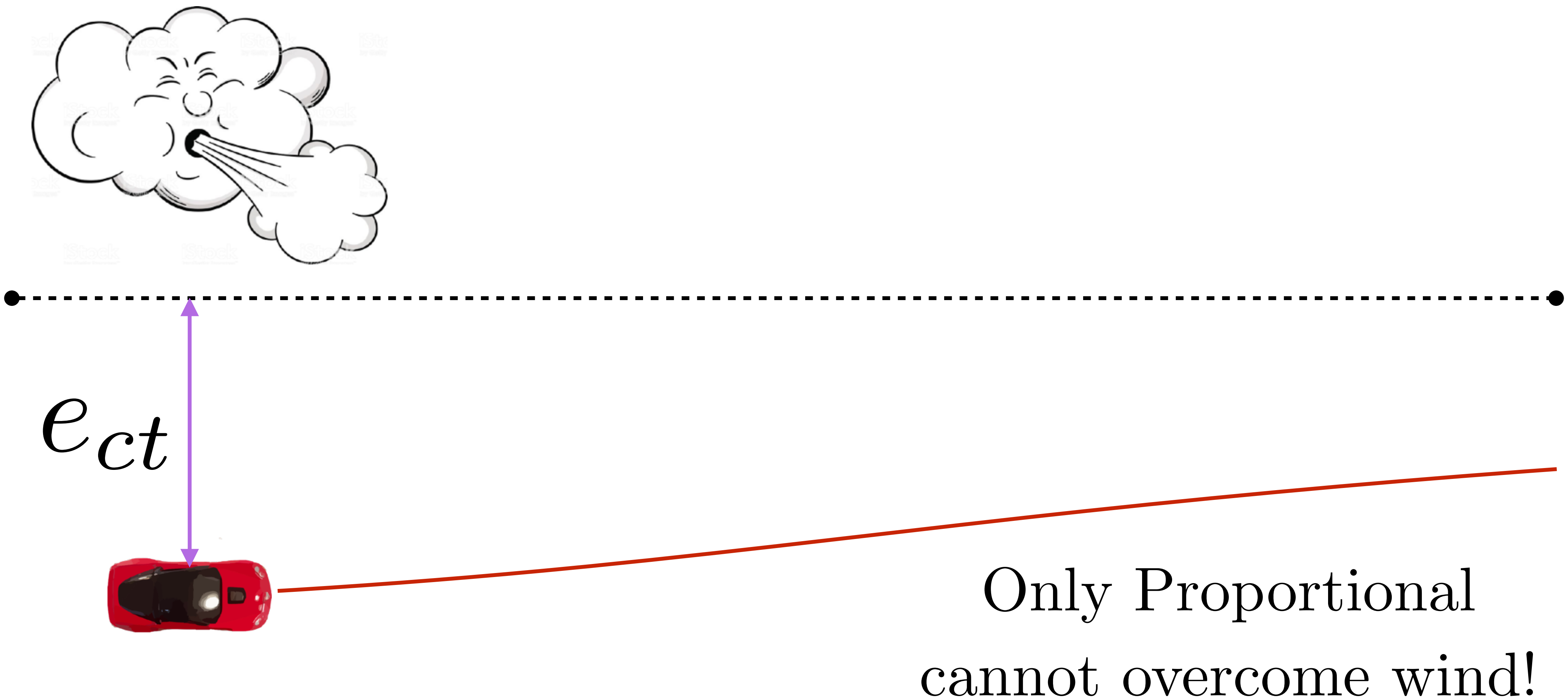
New control law! Penalize error in cross track and in heading

$$u = -(K_p e_{ct} + K_d V \sin\theta_e)$$

Proportional integral control

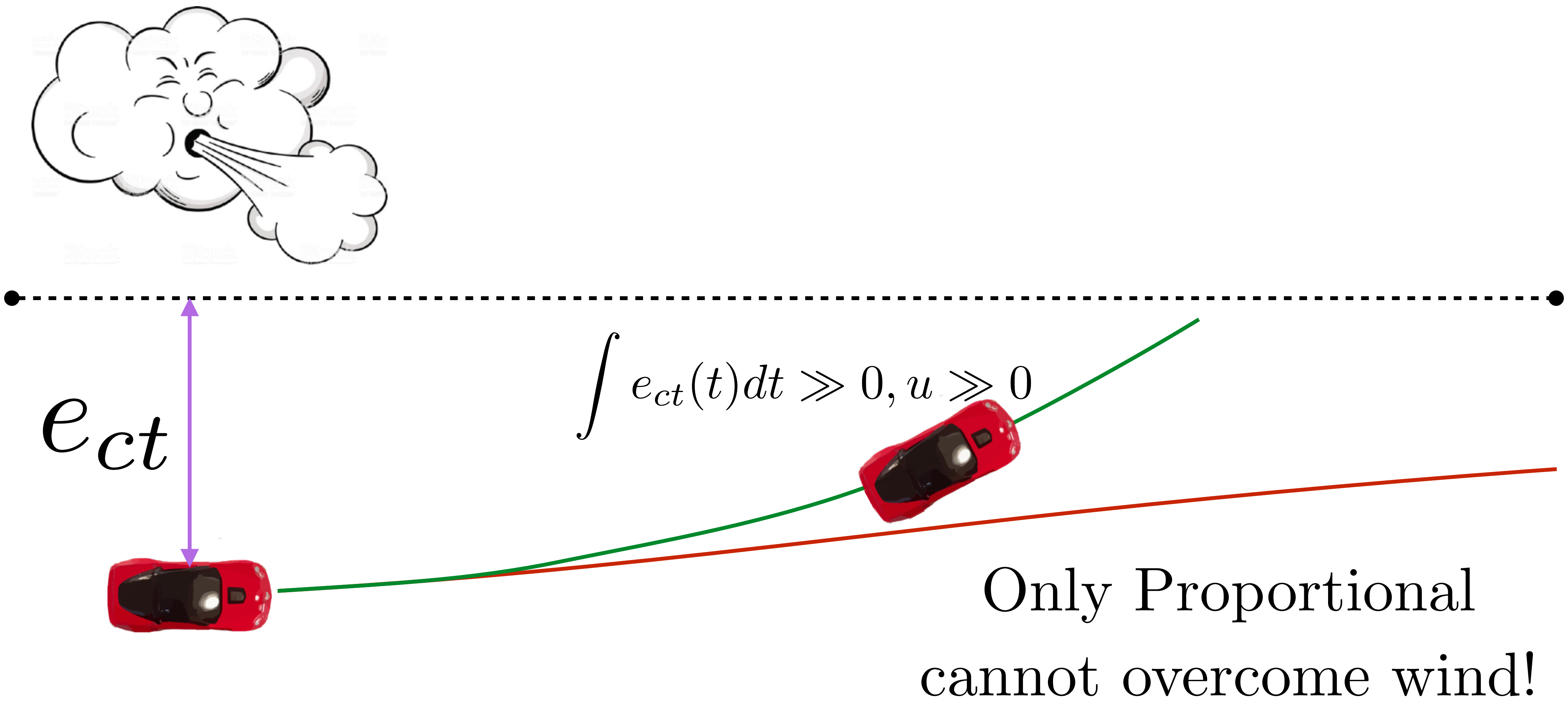
$$u = -\left(K_p e_{ct} + K_i \int e_{ct}(t)dt\right)$$

Proportional integral control



$$u = -\left(K_p e_{ct} + K_i \int e_{ct}(t)dt \right)$$

Proportional integral control



$$u = -\left(K_p e_{ct} + K_i \int e_{ct}(t)dt\right)$$

Different control laws

1. PID control

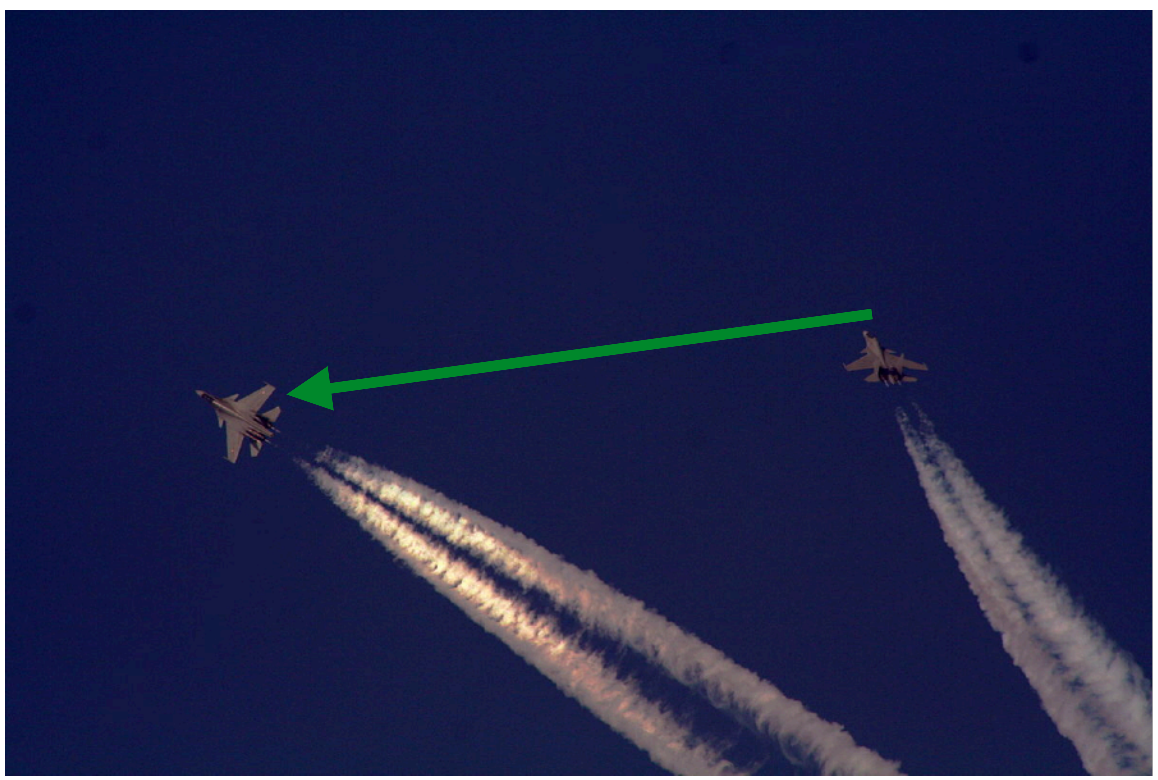
2. Pure-pursuit control

3. Lyapunov control

4. LQR

5. MPC

Pure Pursuit Control



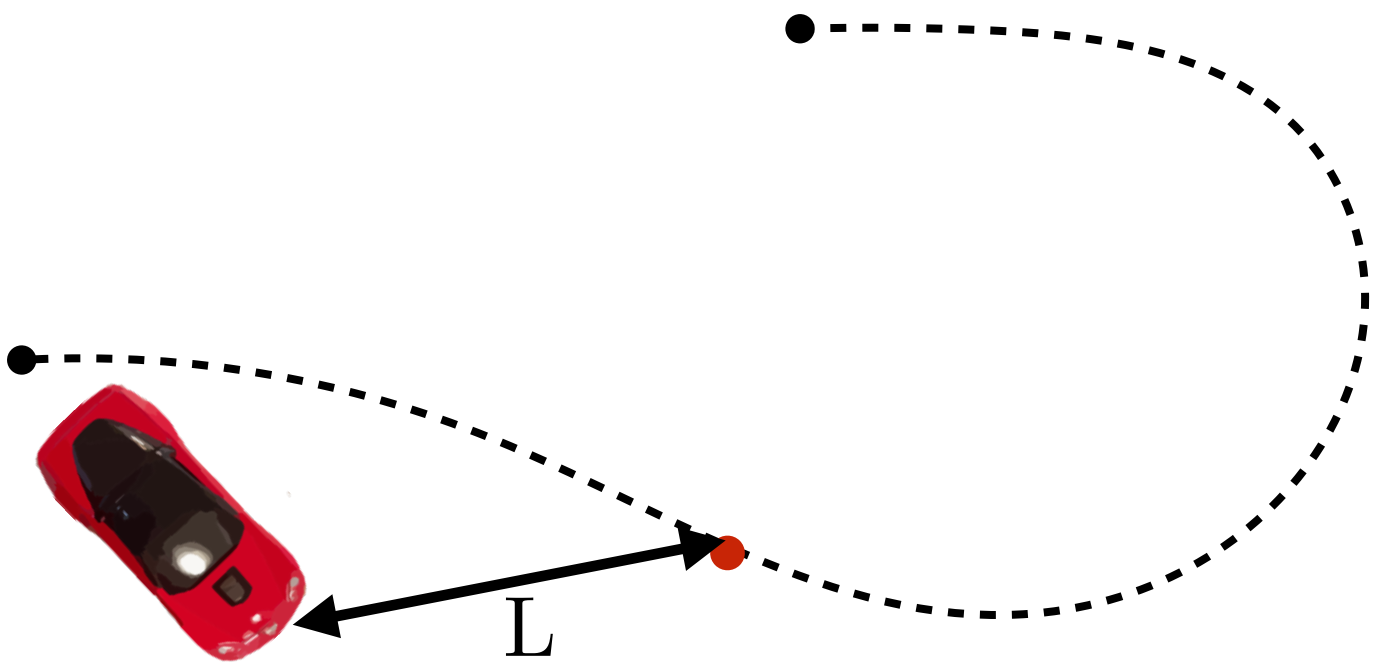
Aerial combat in which aircraft pursues another aircraft by pointing its nose directly towards it

Similar to carrot on a stick!

Key Idea:

The car is always moving in a circular arc

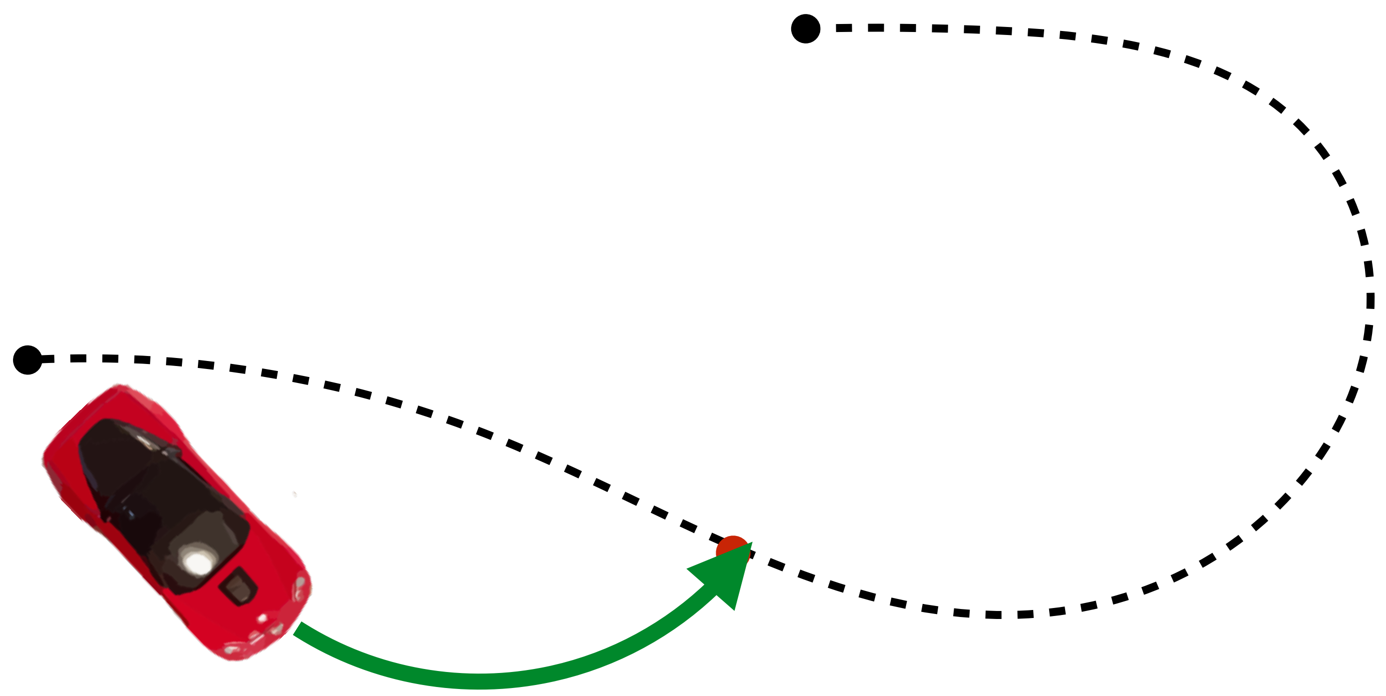
Consider a reference at a lookahead distance



$$\left\| \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_{ref} \\ y_{ref} \end{bmatrix} \right\| = L$$

Problem: Can we solve for a steering angle that guarantees that the car will pass through the reference?

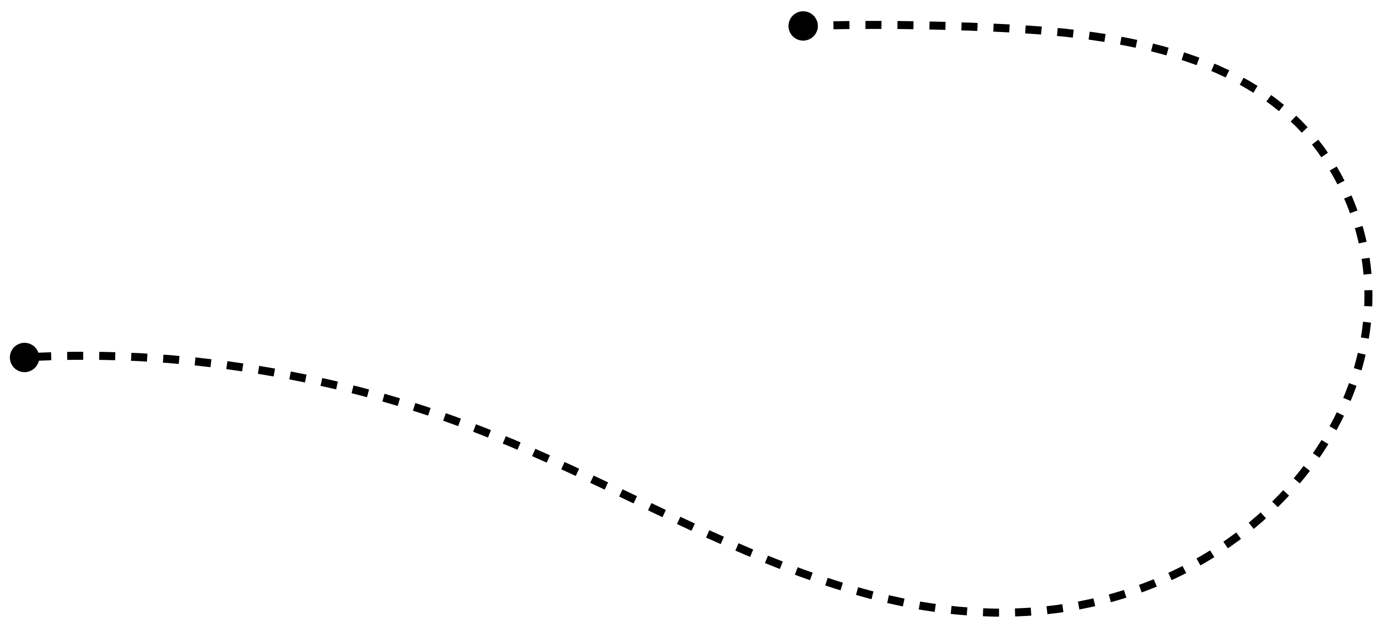
Solution: Compute a circular arc



We can always solve for a arc that
passes through a lookahead point

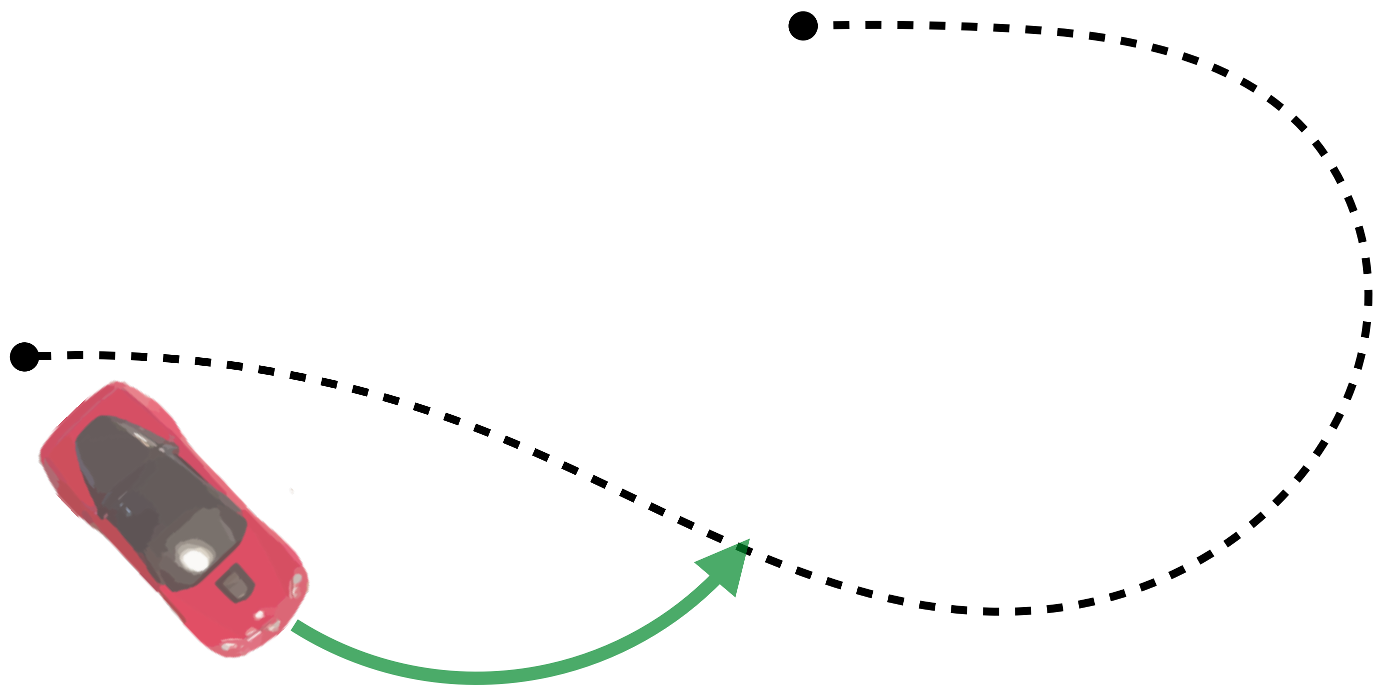
Note: As the car moves forward, the point keeps moving

Pure pursuit: Keep chasing looakahead



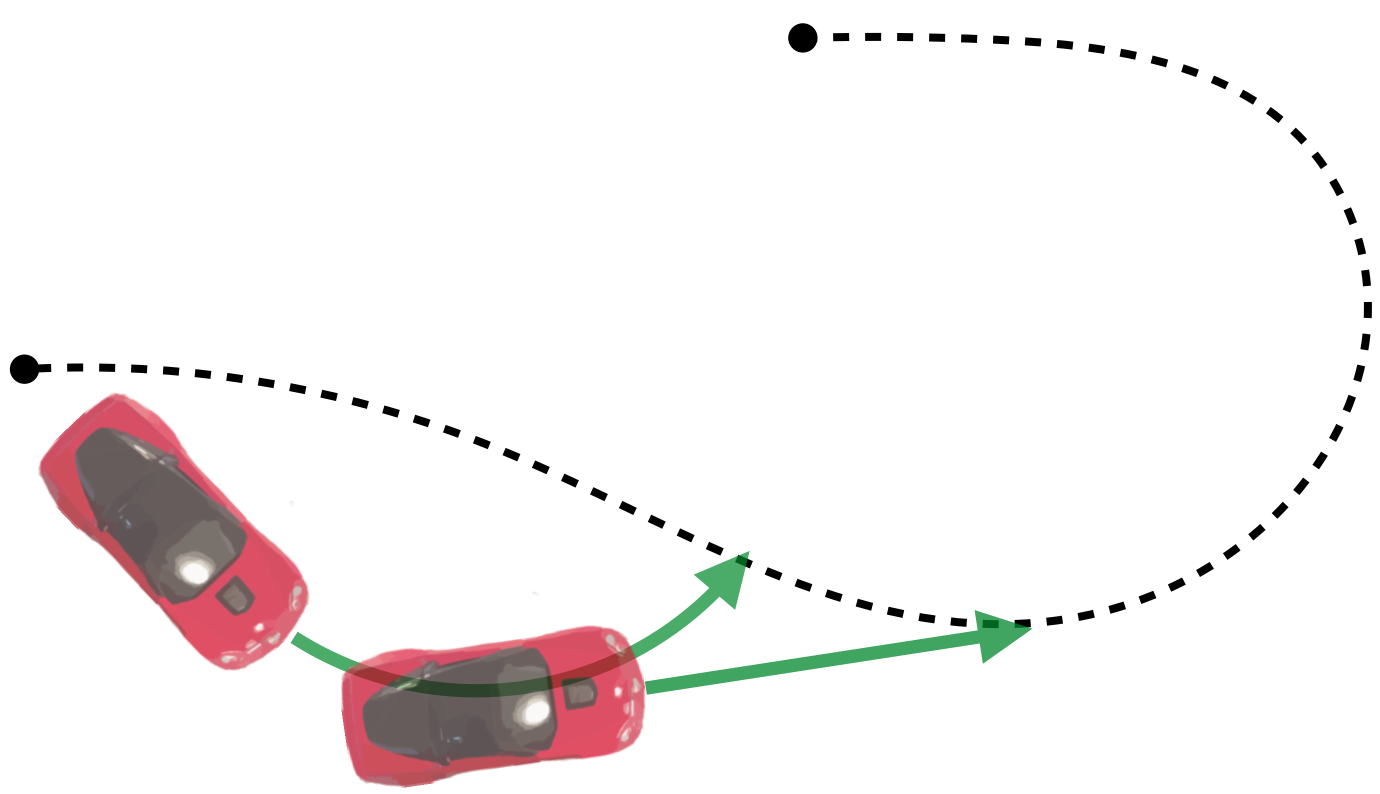
1. Find a lookahead and compute arc
2. Move along the arc
3. Go to step 1

Pure pursuit: Keep chasing looakahead



1. Find a lookahead and compute arc
2. Move along the arc
3. Go to step 1

Pure pursuit: Keep chasing looakahead

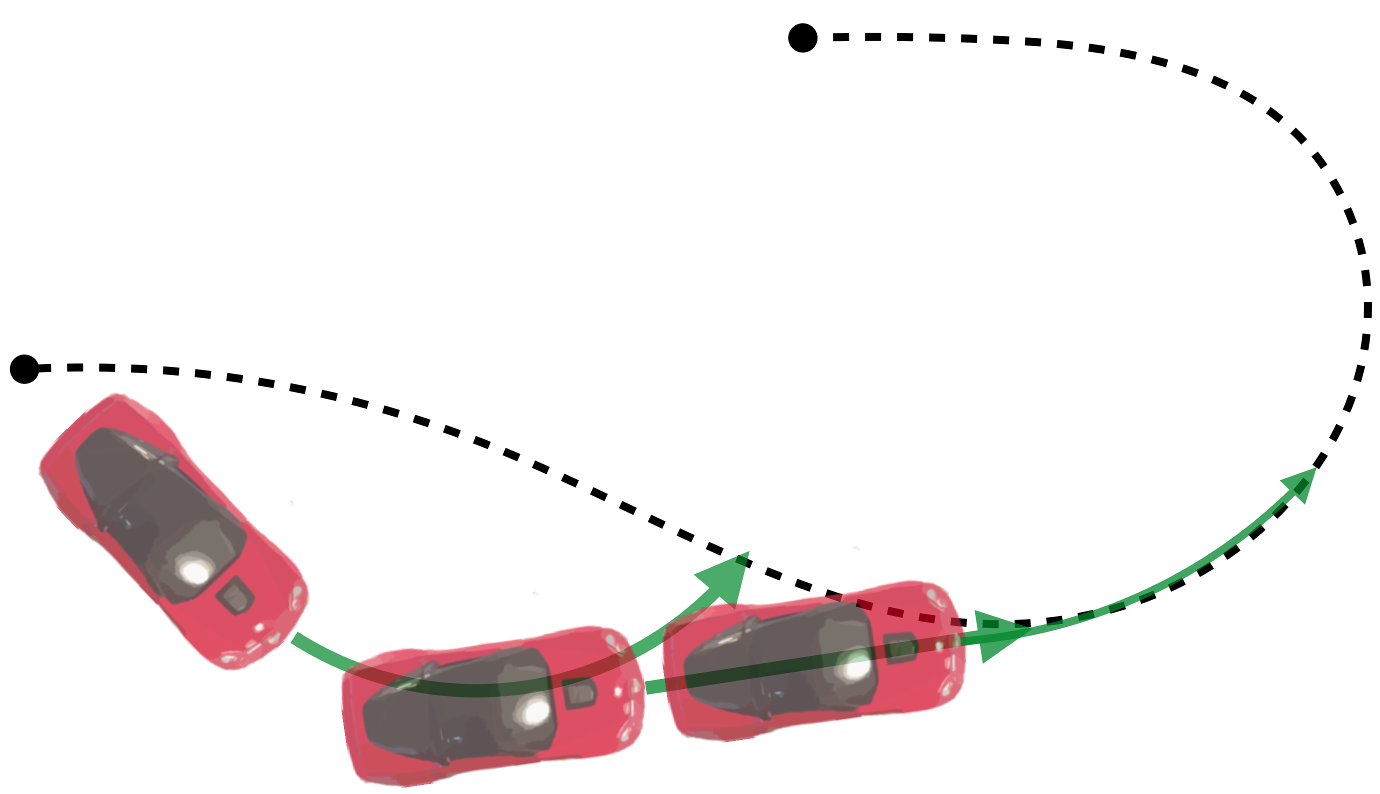


1. Find a lookahead and compute arc

2. Move along the arc

3. Go to step 1

Pure pursuit: Keep chasing looakahead

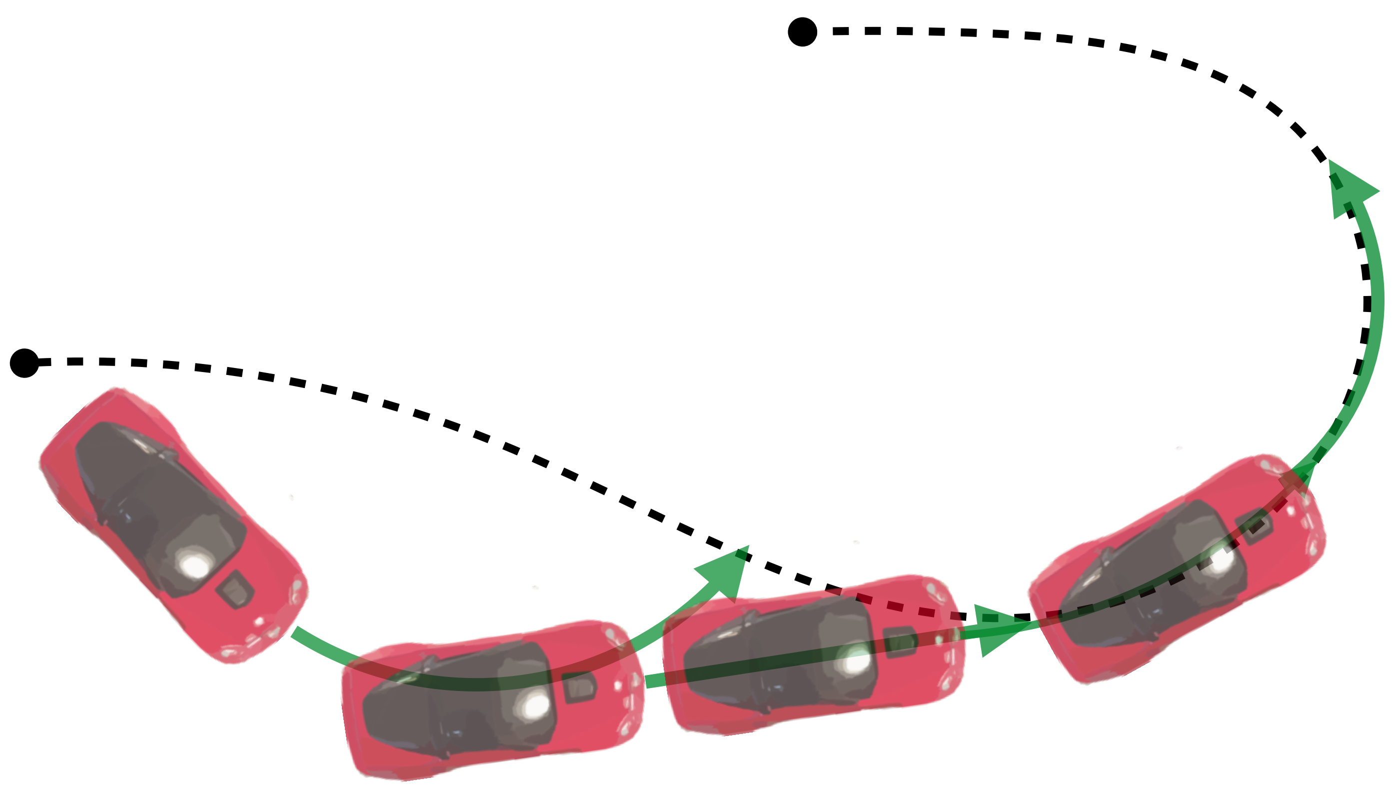


1. Find a lookahead and compute arc

2. Move along the arc

3. Go to step 1

Pure pursuit: Keep chasing looakahead

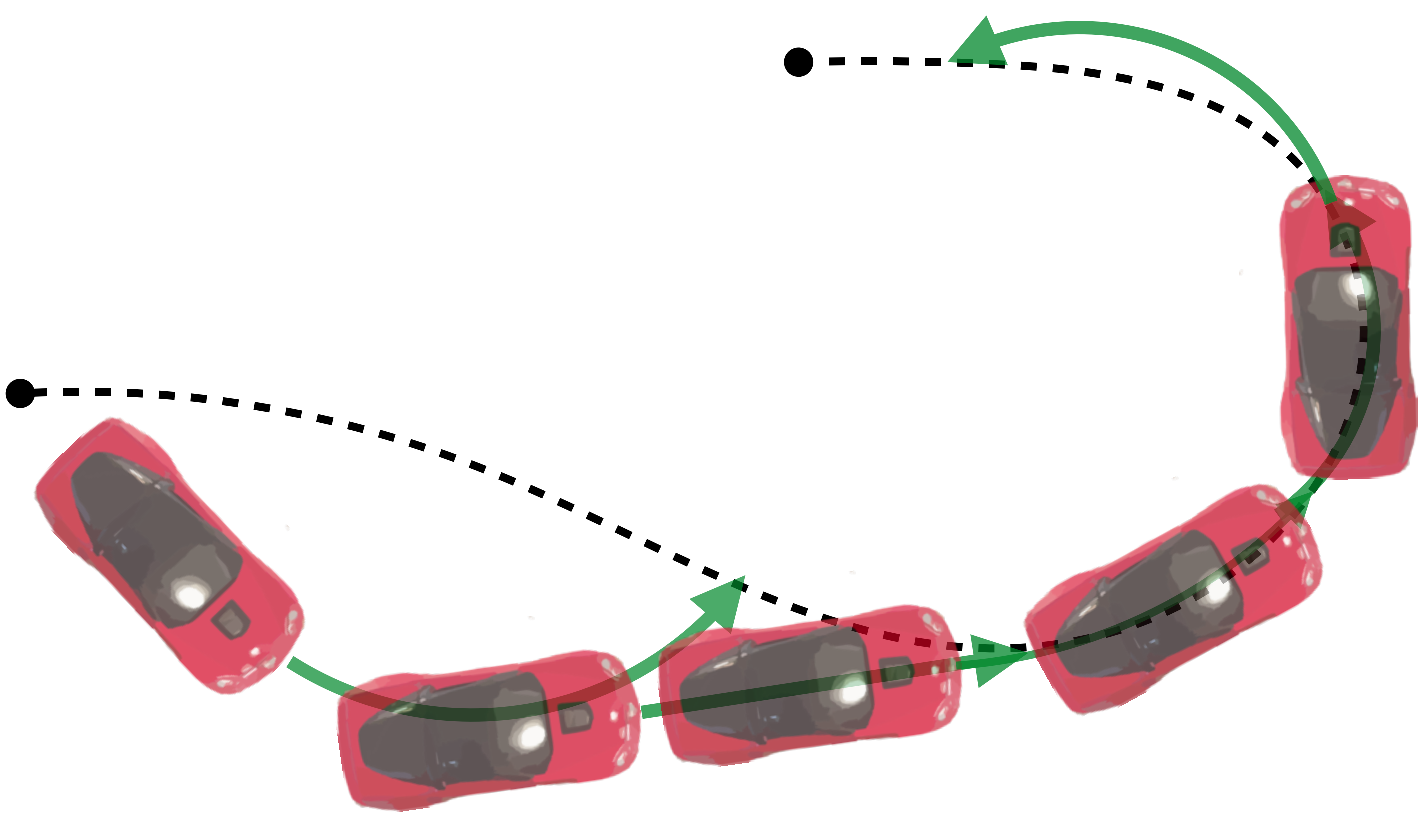


1. Find a lookahead and compute arc

2. Move along the arc

3. Go to step 1

Pure pursuit: Keep chasing looakahead

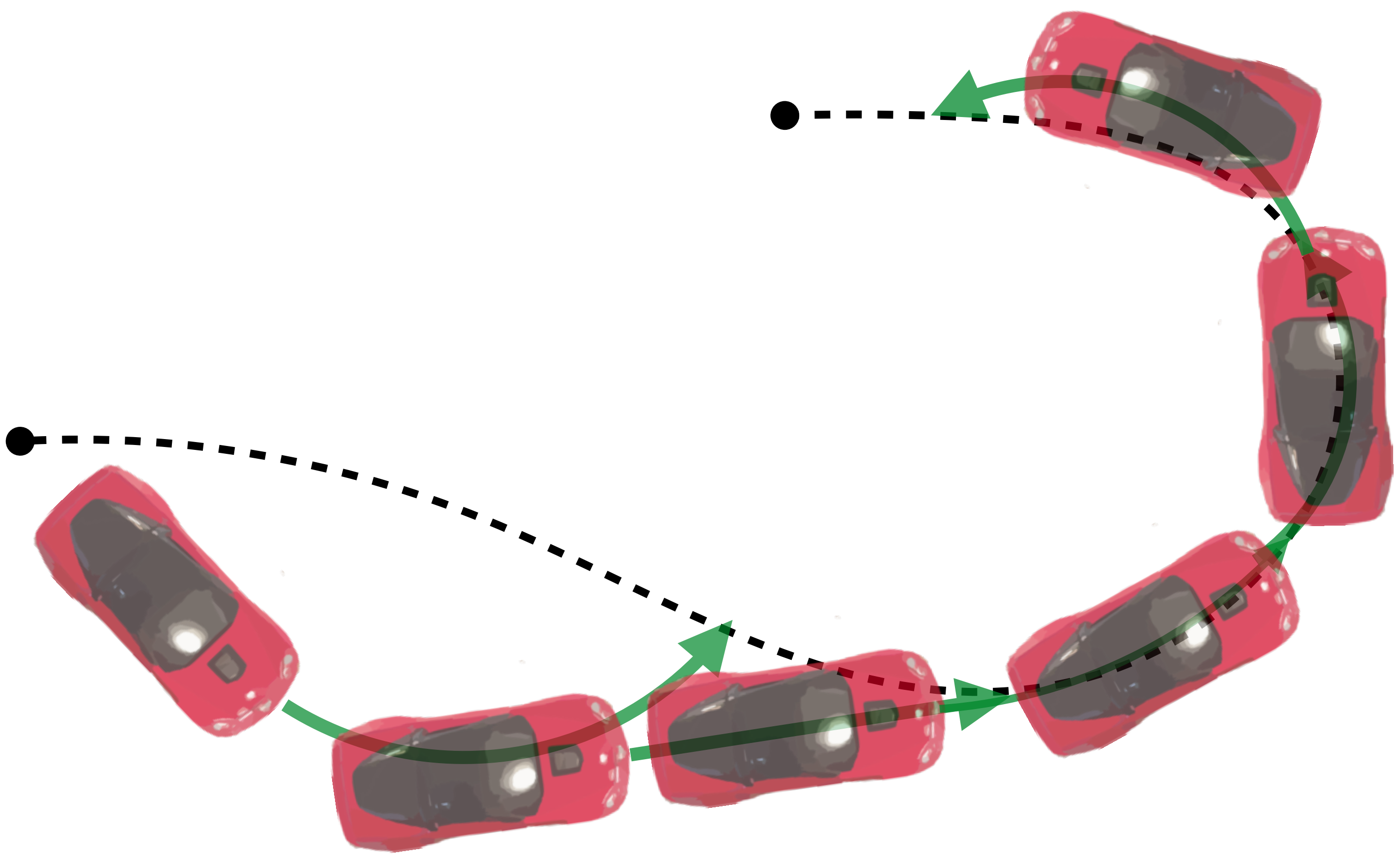


1. Find a lookahead and compute arc

2. Move along the arc

3. Go to step 1

Pure pursuit: Keep chasing looakahead

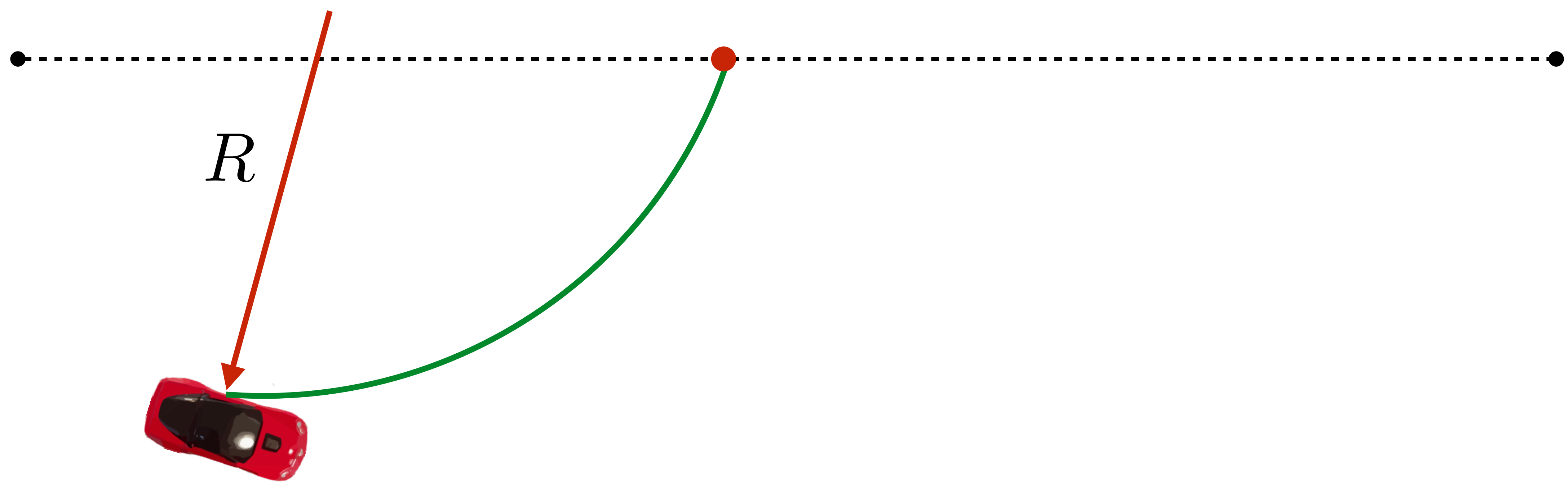


1. Find a lookahead and compute arc

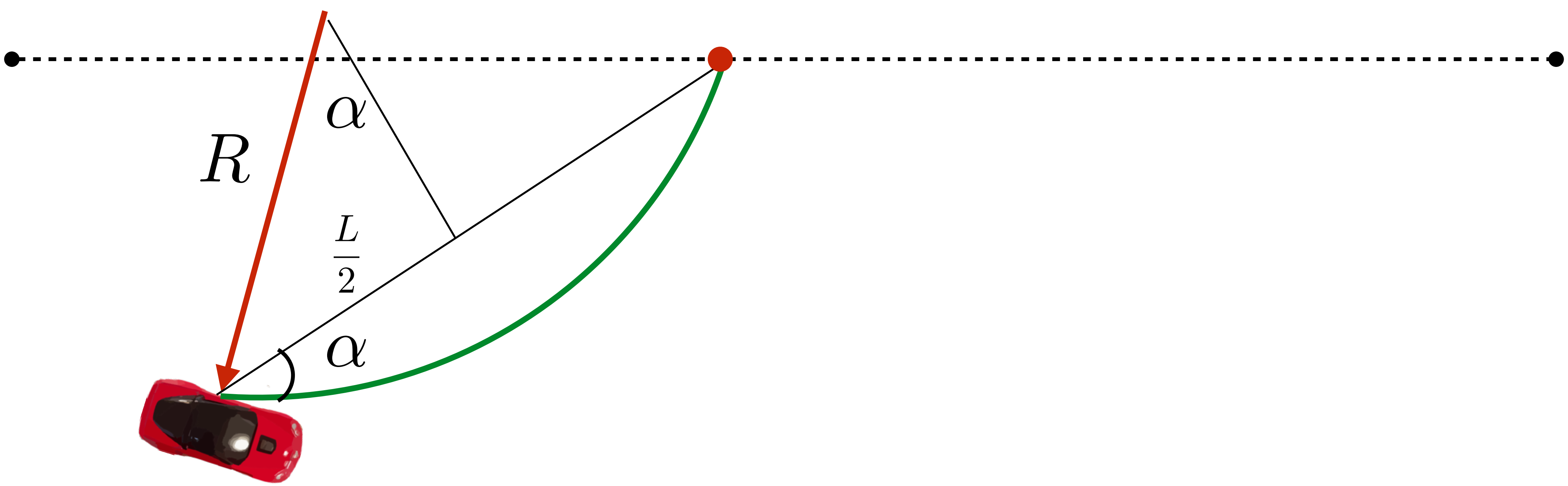
2. Move along the arc

3. Go to step 1

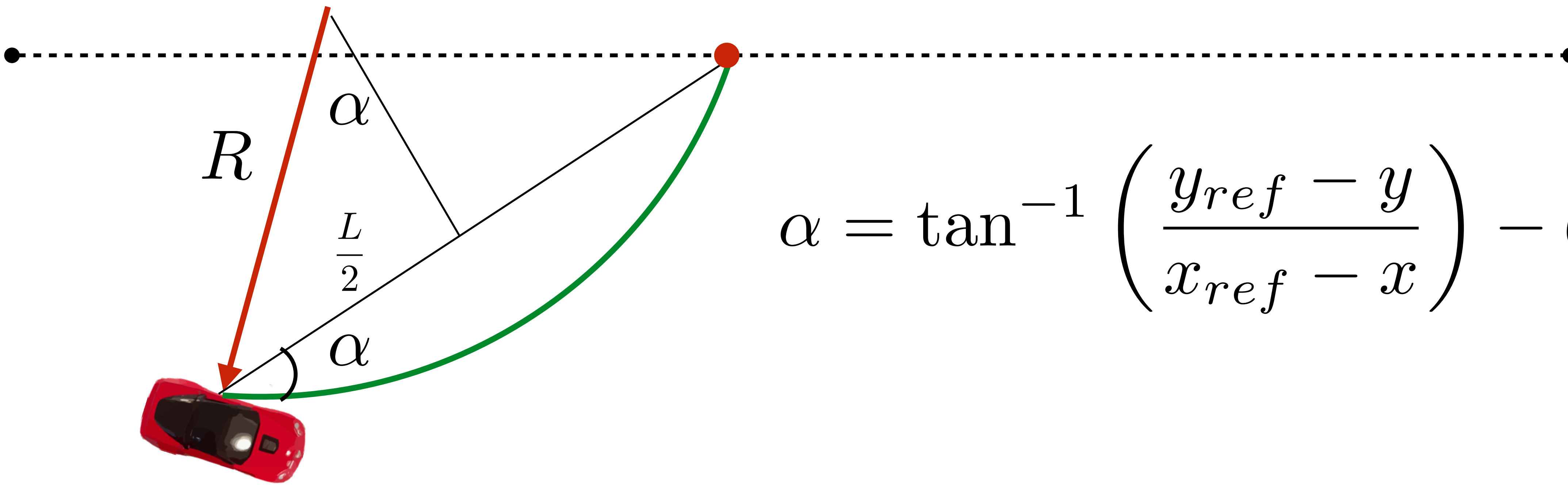
Control law derivation: Solve for arc



Control law derivation: Solve for arc

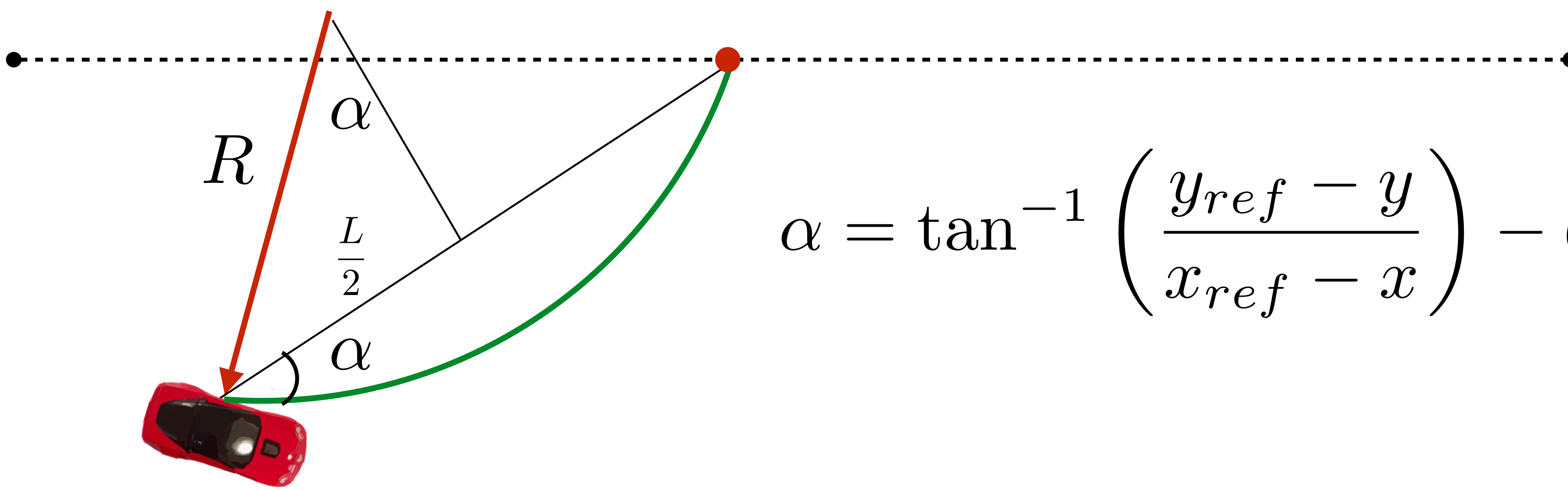


Control law derivation: Solve for arc



$$\alpha = \tan^{-1}\left(\frac{y_{ref} - y}{x_{ref} - x}\right) - \theta$$

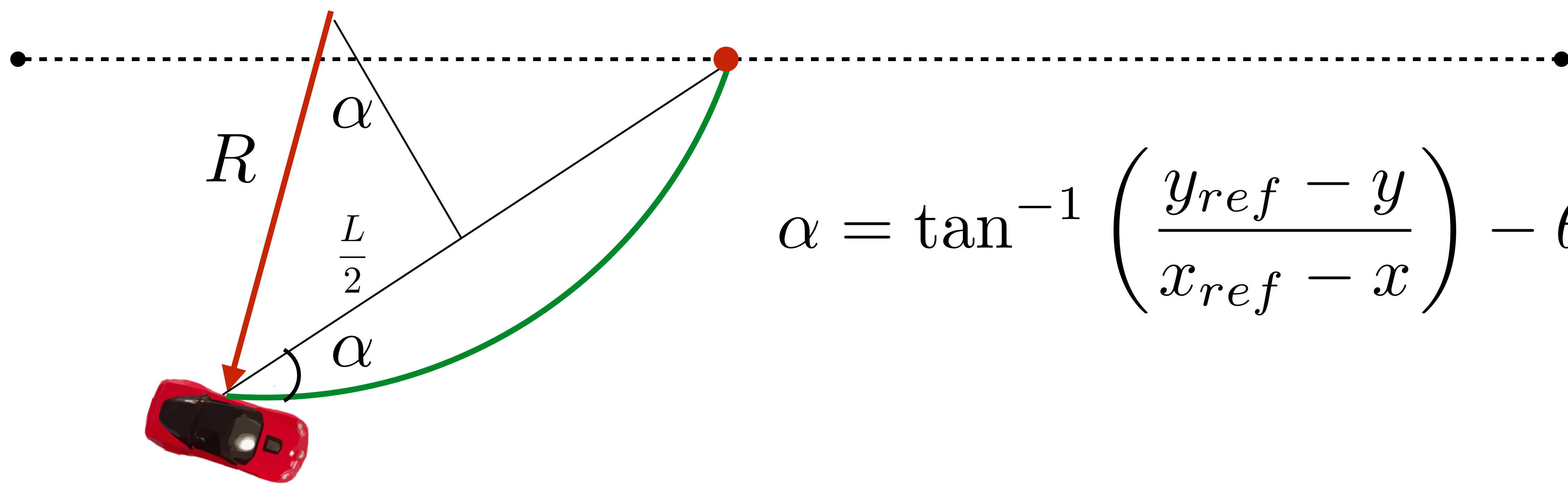
Control law derivation: Solve for arc



$$\alpha = \tan^{-1}\left(\frac{y_{ref} - y}{x_{ref} - x}\right) - \theta$$

$$\sin\alpha = \frac{L}{2R}$$

Control law derivation: Solve for arc

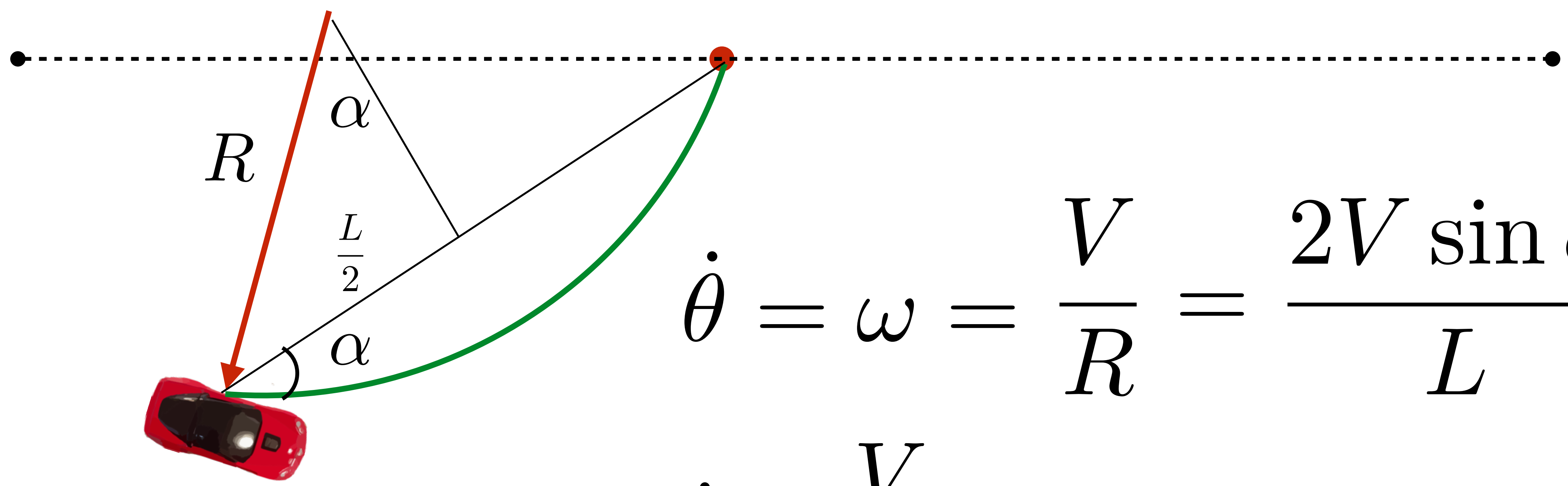


$$\alpha = \tan^{-1}\left(\frac{y_{ref} - y}{x_{ref} - x}\right) - \theta$$

$$\sin\alpha = \frac{L}{2R}$$

$$R = \frac{L}{2\sin\alpha}$$

Control law derivation

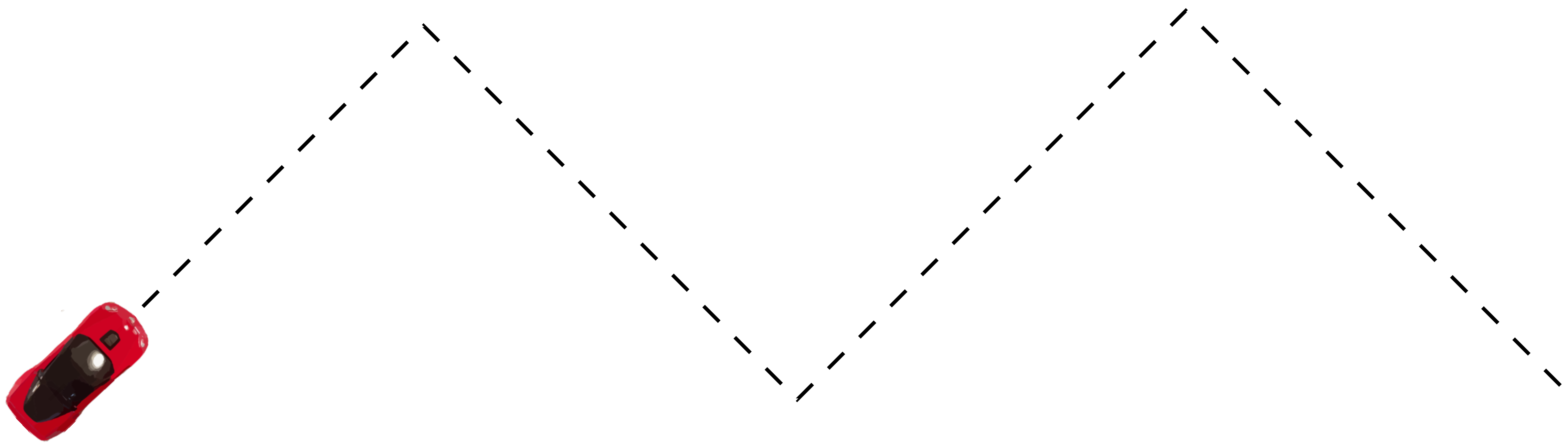


$$\dot{\theta} = \omega = \frac{V}{R} = \frac{2V \sin\alpha}{L}$$

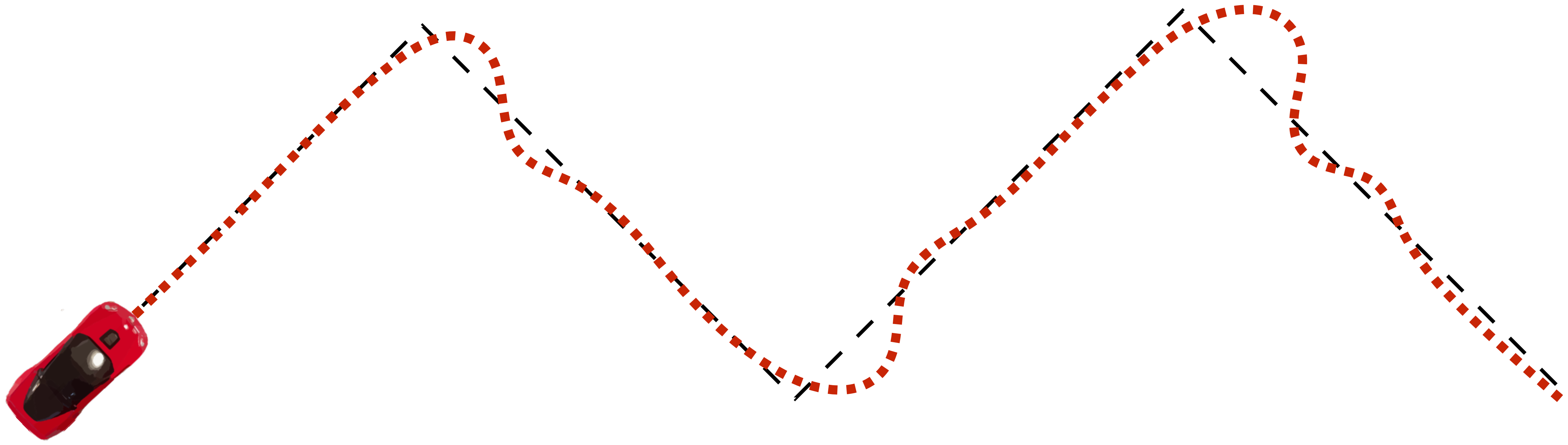
$$\dot{\theta} = \frac{V}{B} \tan u$$

$$\boxed{u = \tan^{-1}\left(\frac{2B \sin\alpha}{L}\right)}$$

Question: How do I choose L?



Question: How do I choose L?



Question: How do I choose L?