PID and Pure Pursuit Control

Instructor: Chris Mavrogiannis

TAs: Kay Ke, Gilwoo Lee, Matt Schmittle

*Slides based on or adapted from Sanjiban Choudhury

Logistics

1. Office hours poll

2. Mid-quarter course evaluation

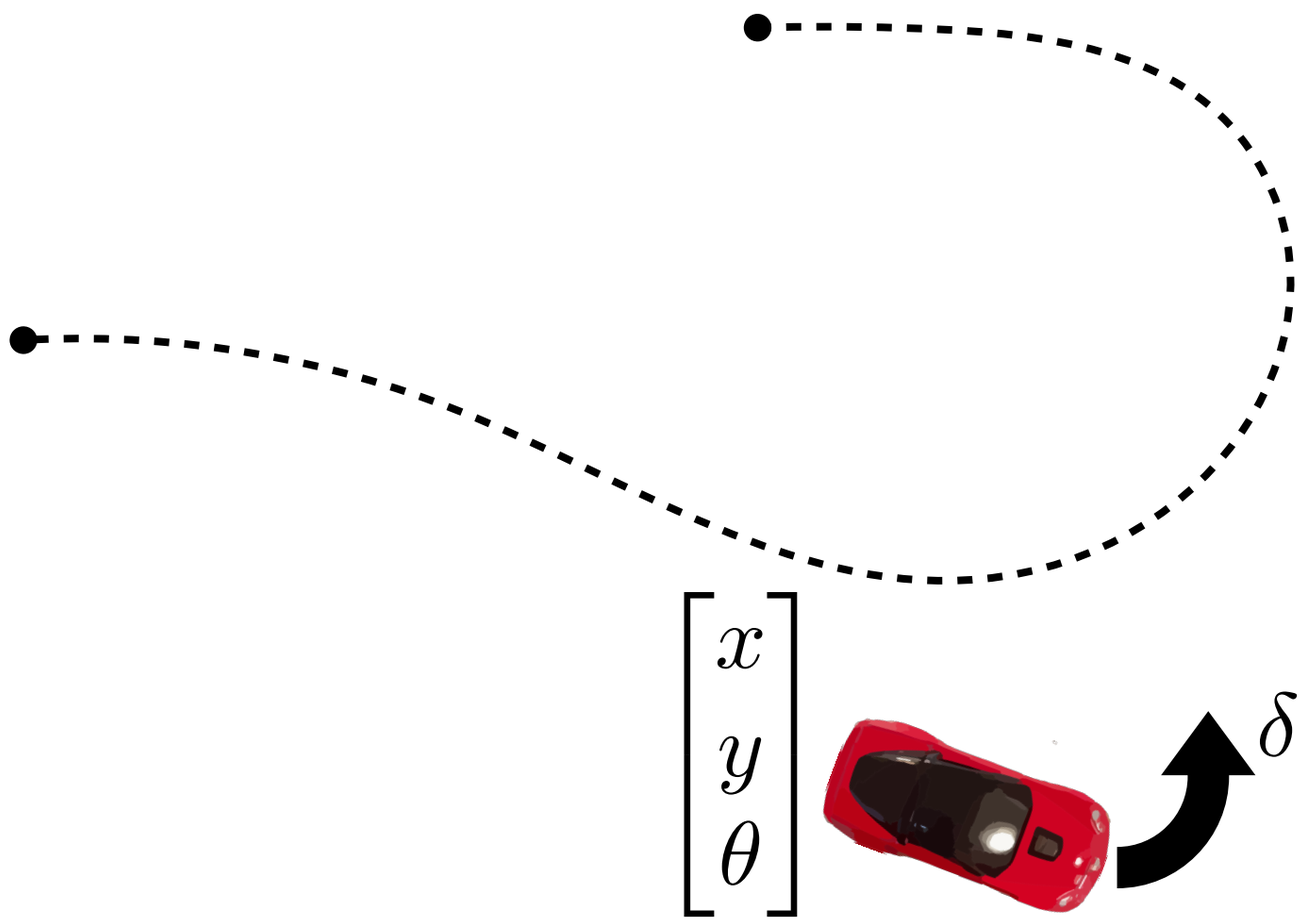
3. Apologies for errors/typos in assignments

4. **Crucial** that you attend office hours

5. Simulation vs Real World

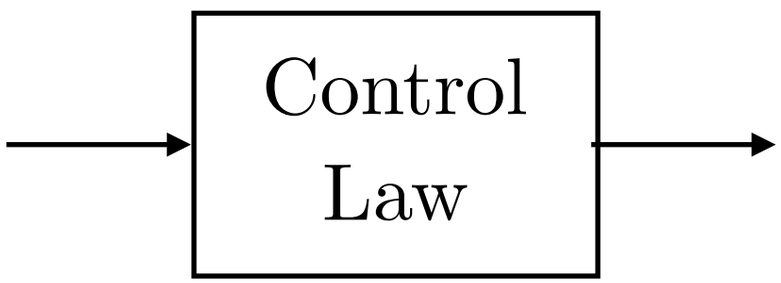
The Control API

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



Input

1. Reference path
2. Current state



Output

Control action

Steps to designing a controller

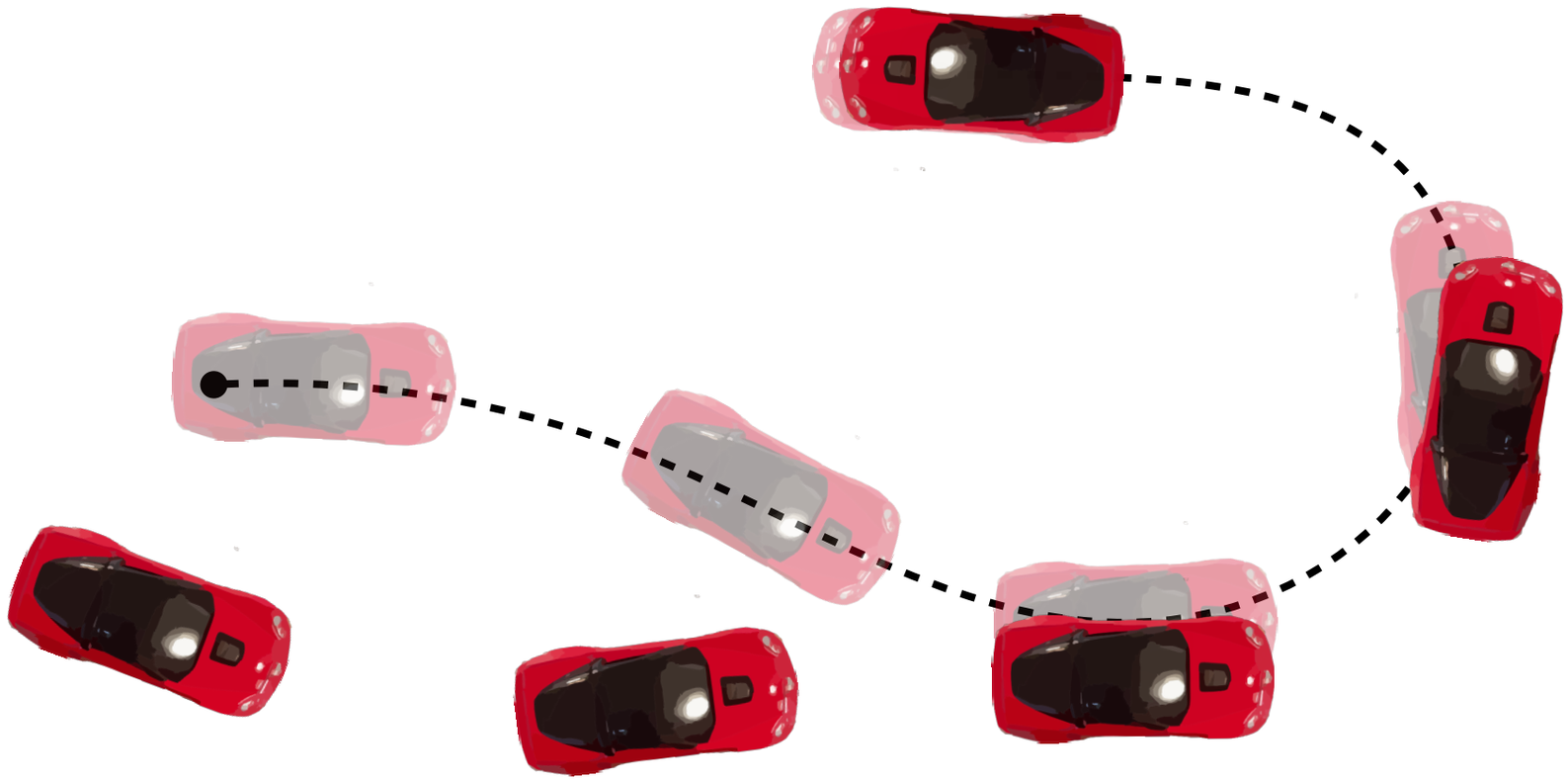
1. Get a reference path / trajectory to track

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)

Steps to designing a controller

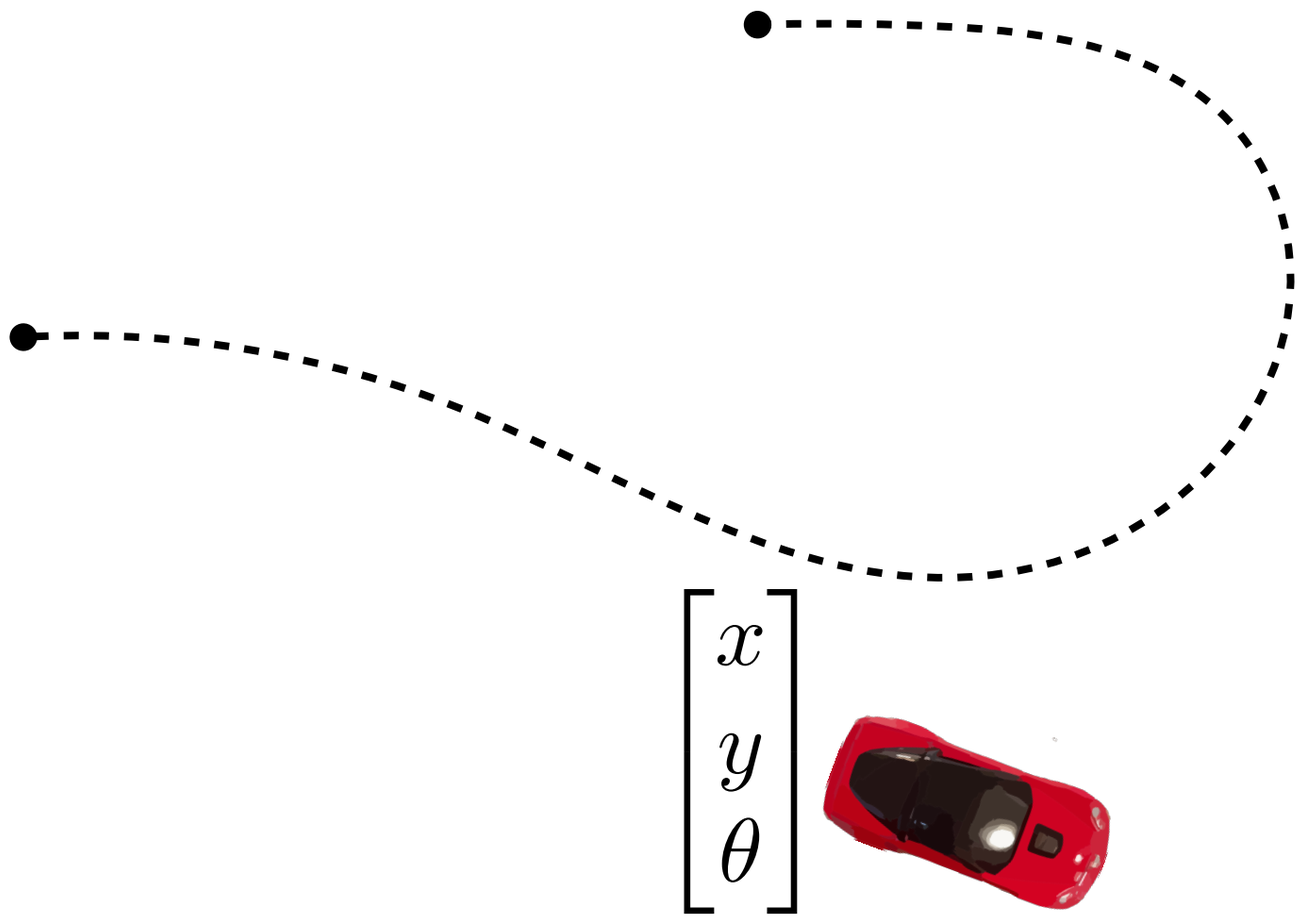
1. Get a reference path / trajectory to track

2. Pick a point on the reference

3. Compute error to reference point

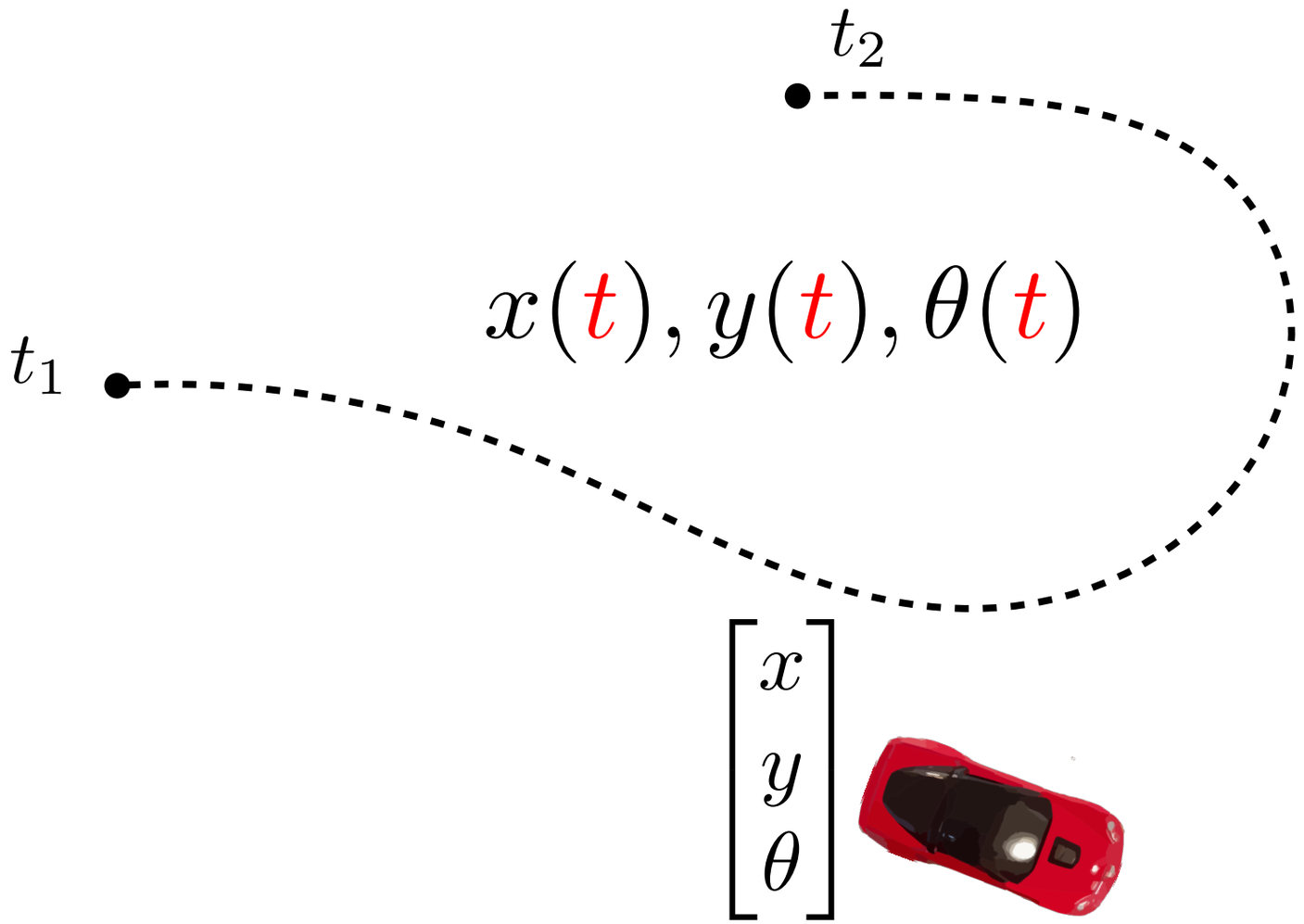
4. Compute control law to minimize error

Step 1: Get a reference path



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

How do we define a reference?

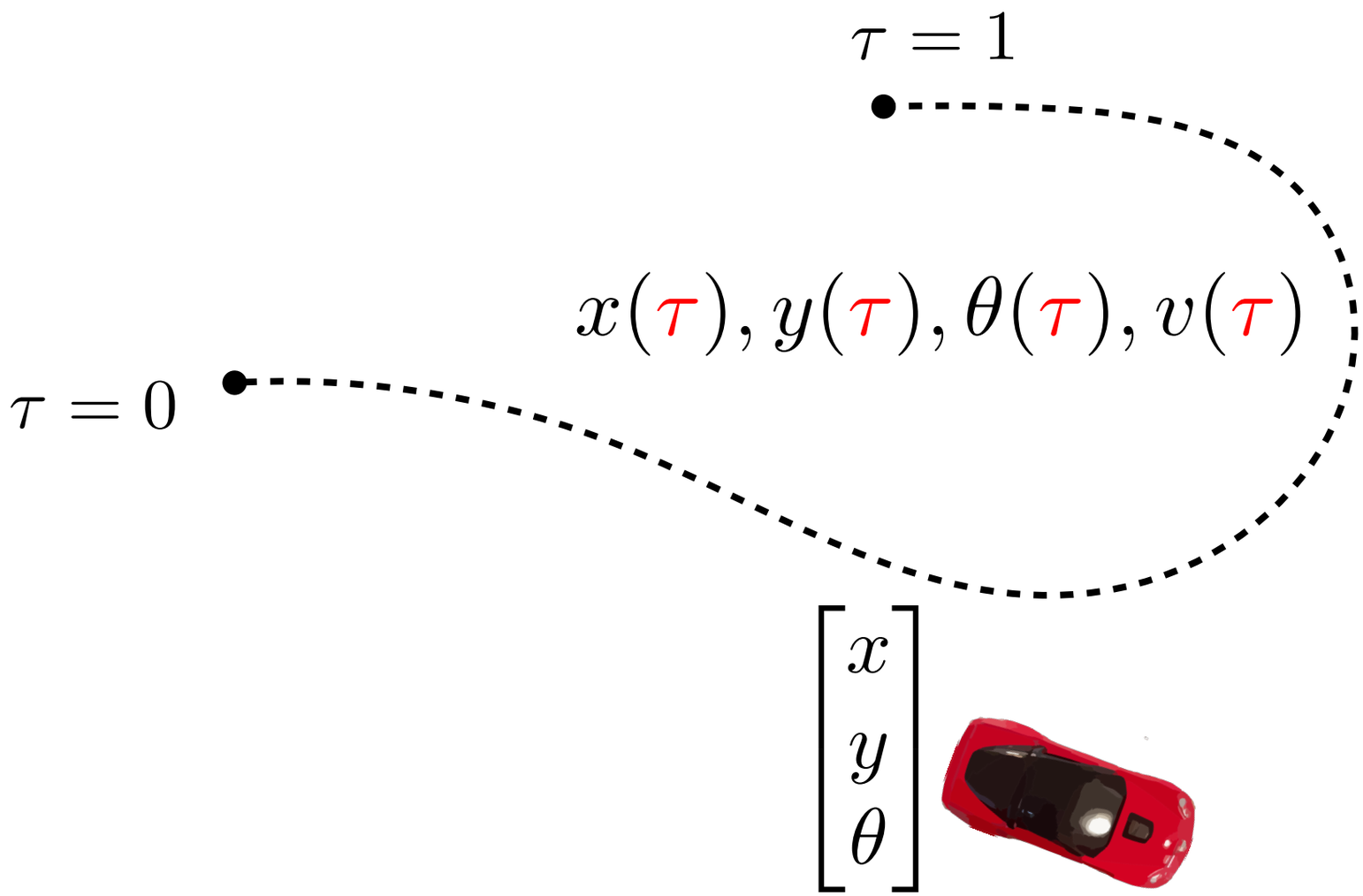


Option 1: Time-parameterized *trajectory*

Pro: Useful if we want the robot to respect time constraints

Con: Sometimes we care only about deviation from reference

How do we define a reference?

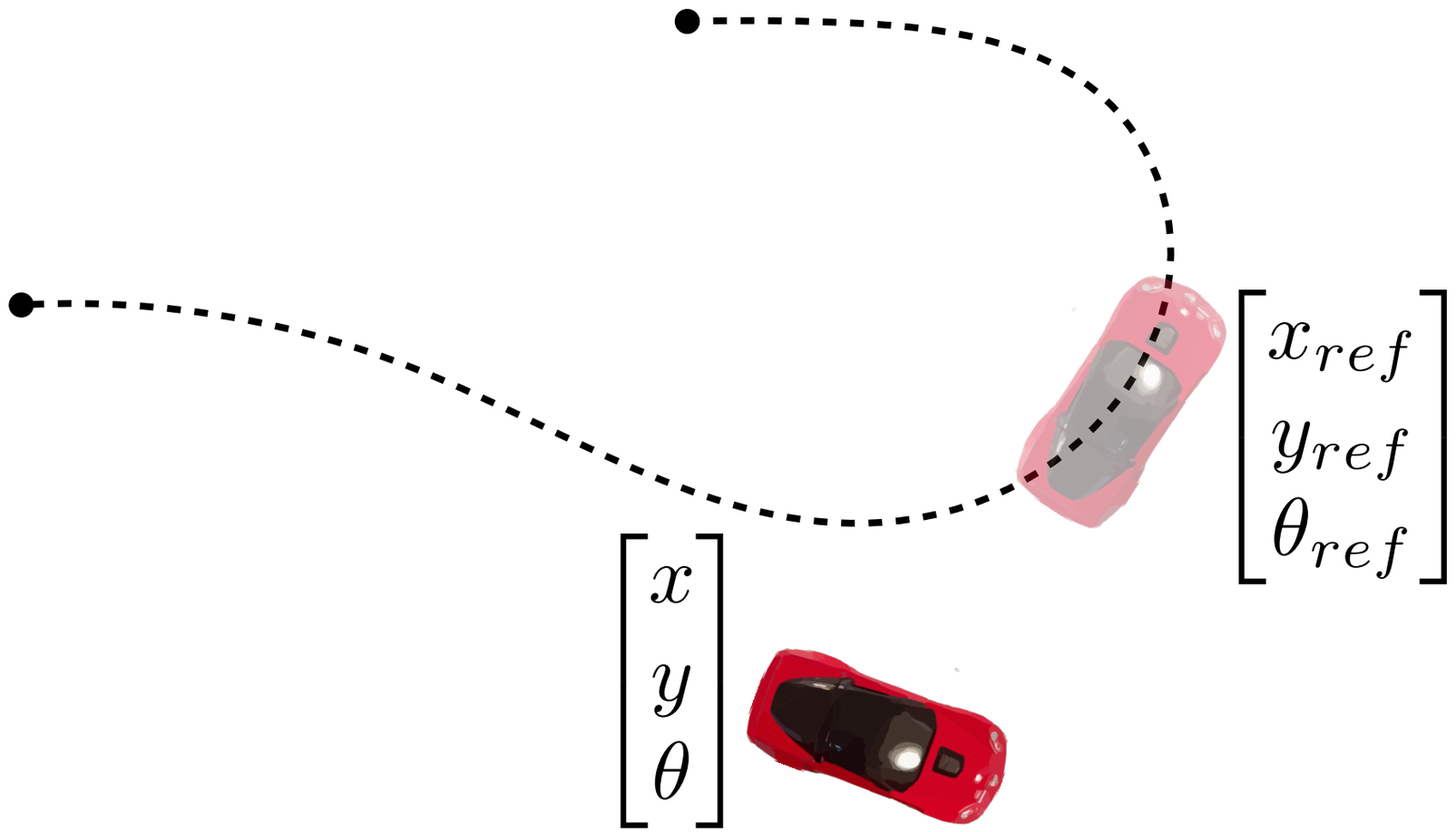


Option 2: Index-parametrized *path*

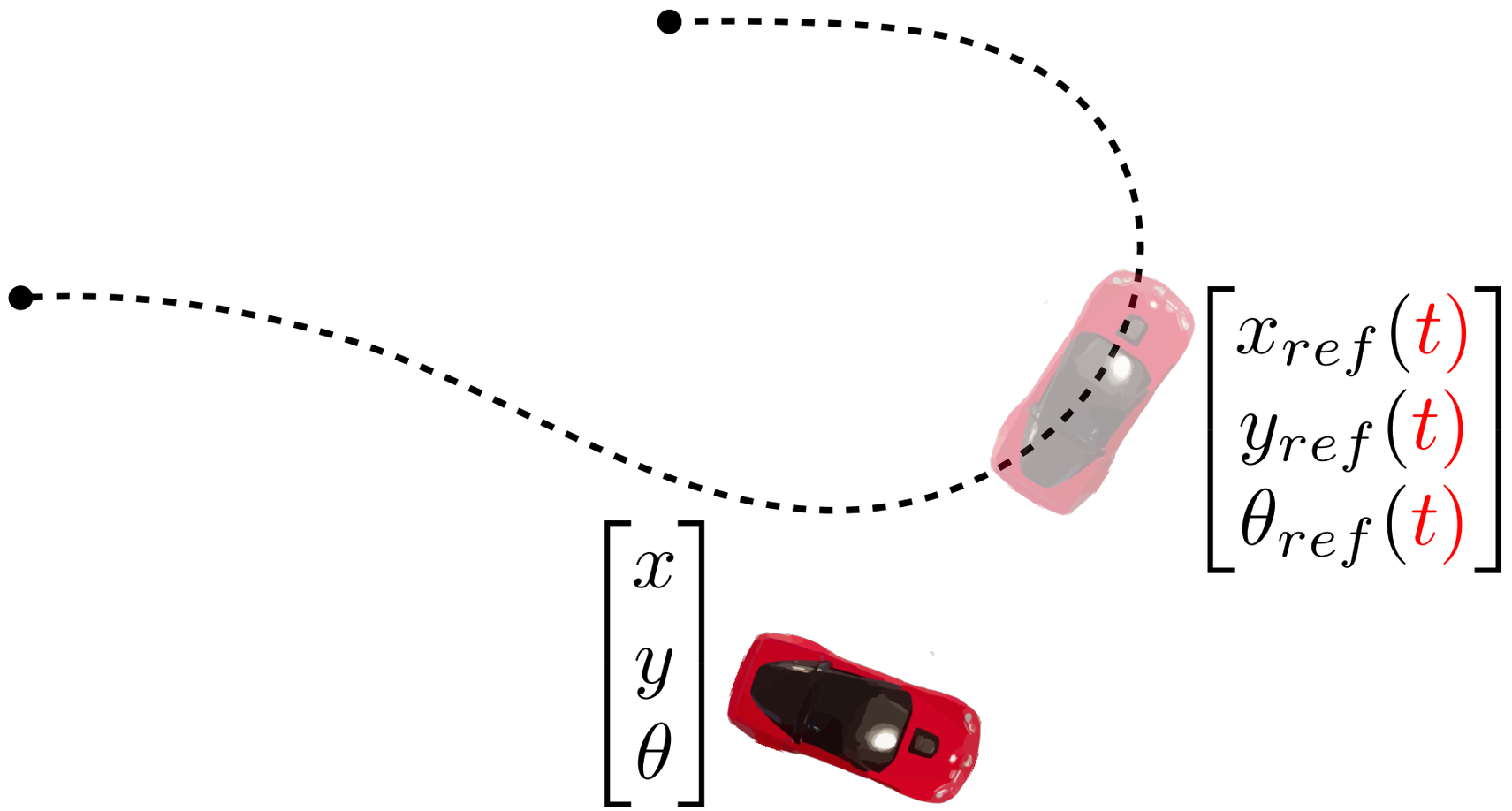
Pro: Useful for conveying the shape you want the robot to follow

Con: Can't control when robot will reach a point

Step 2: Pick a reference (desired) state



How do we pick a reference?



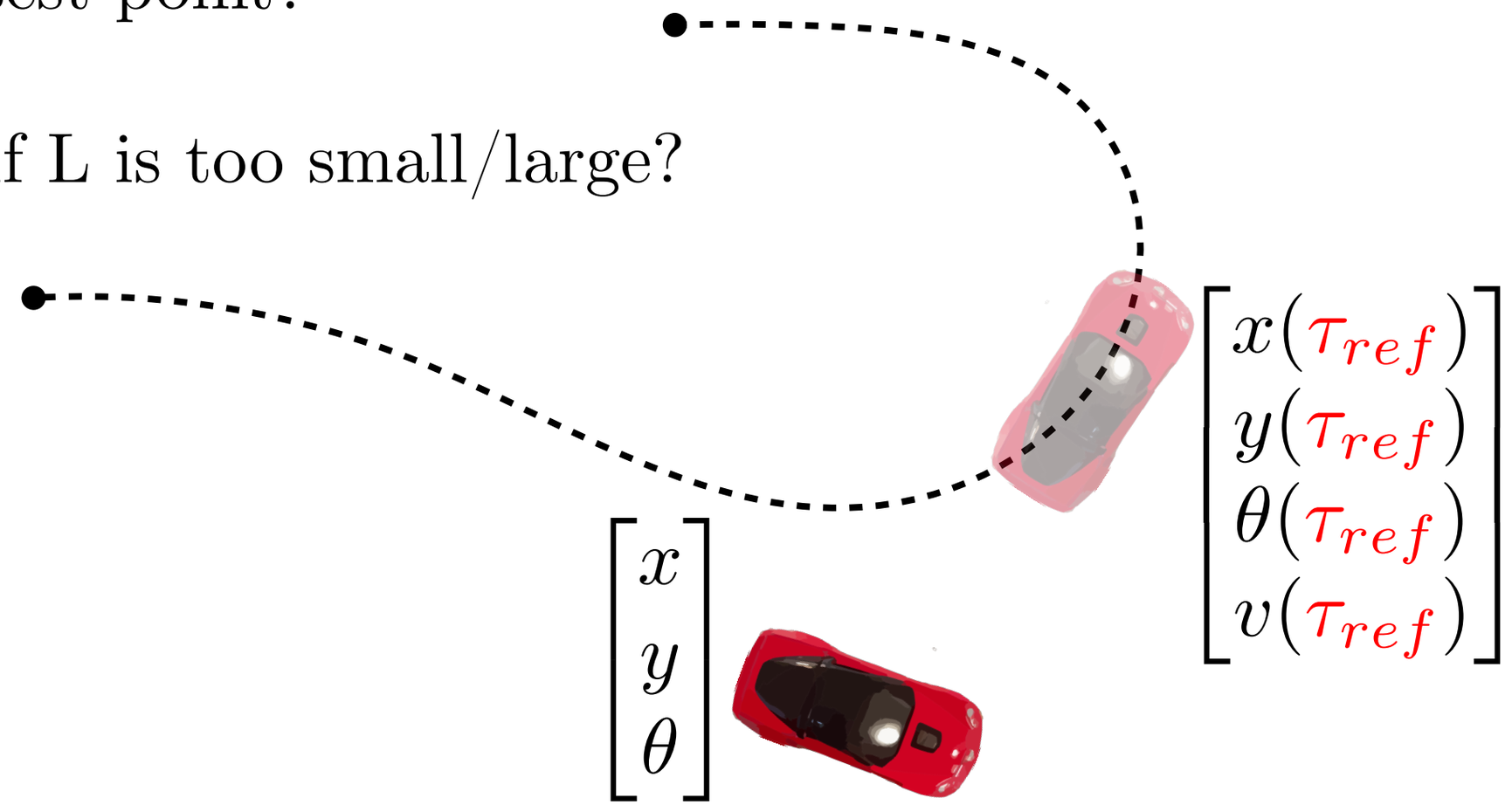
Option 1: Time-parameterized trajectory

Any problems with that?

How do we pick a reference?

Issues with Closest point?

What happens if L is too small/large?

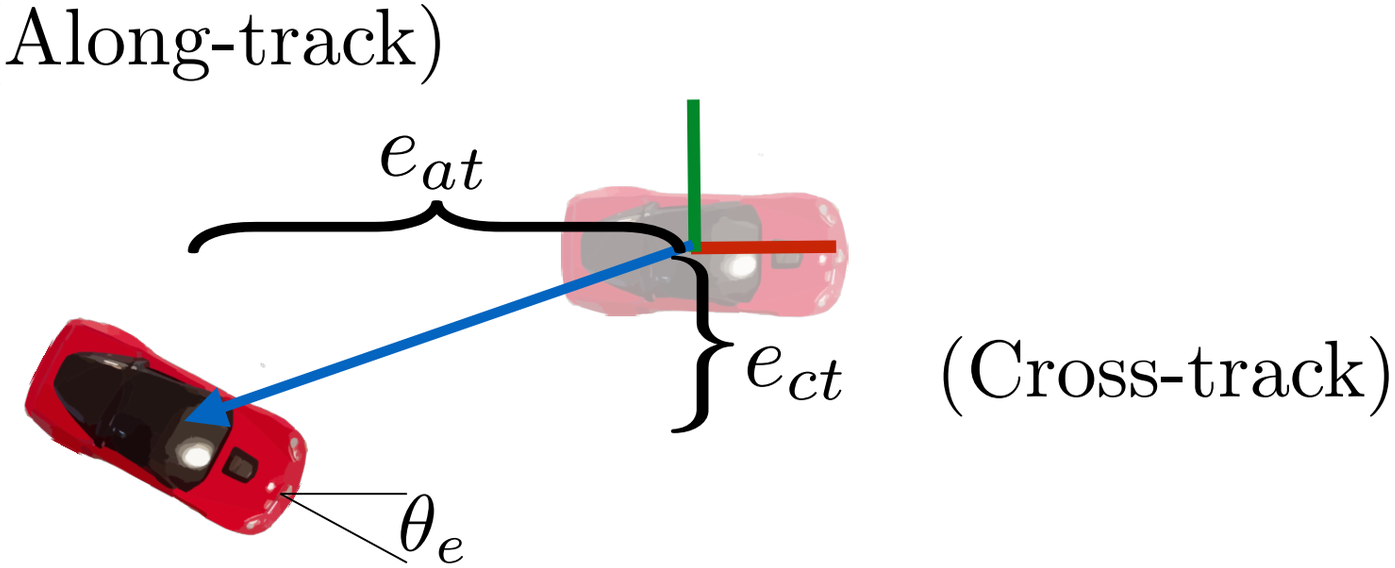


Option 2: Index-parameterized path

Closest point $\quad \tau_{ref} = \arg\min_{\tau} || \begin{bmatrix} x & y \end{bmatrix}^T - \begin{bmatrix} x(\tau) & y(\tau) \end{bmatrix}^T ||$

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

Step 3: Compute error to this state

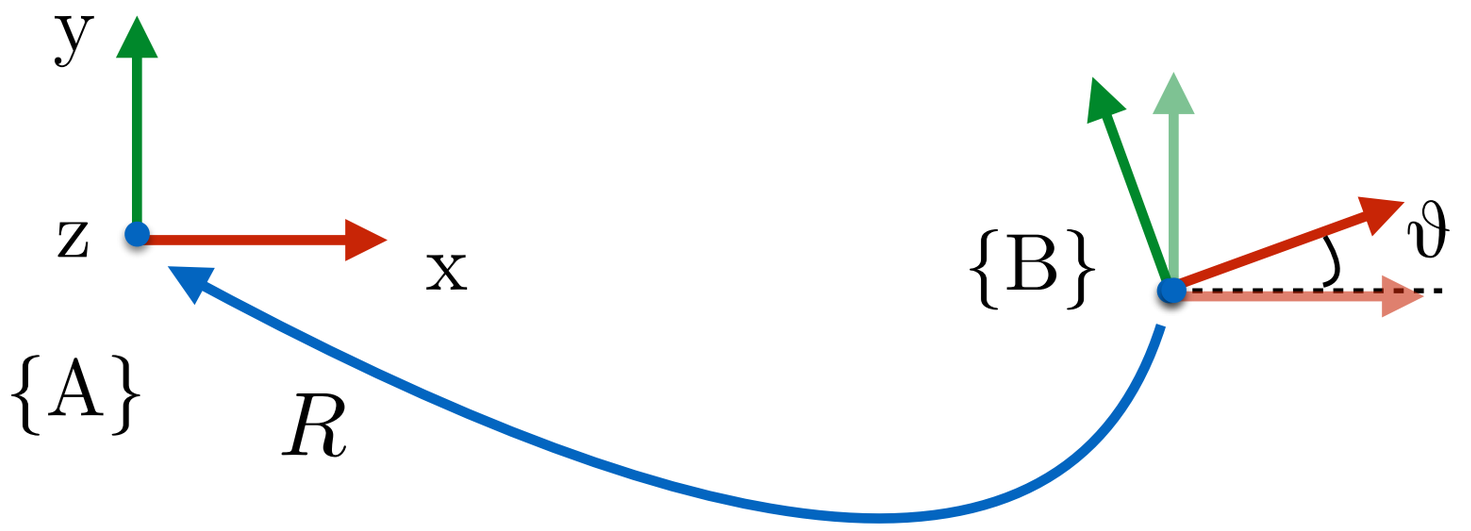


Step 3: Compute error to this state

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

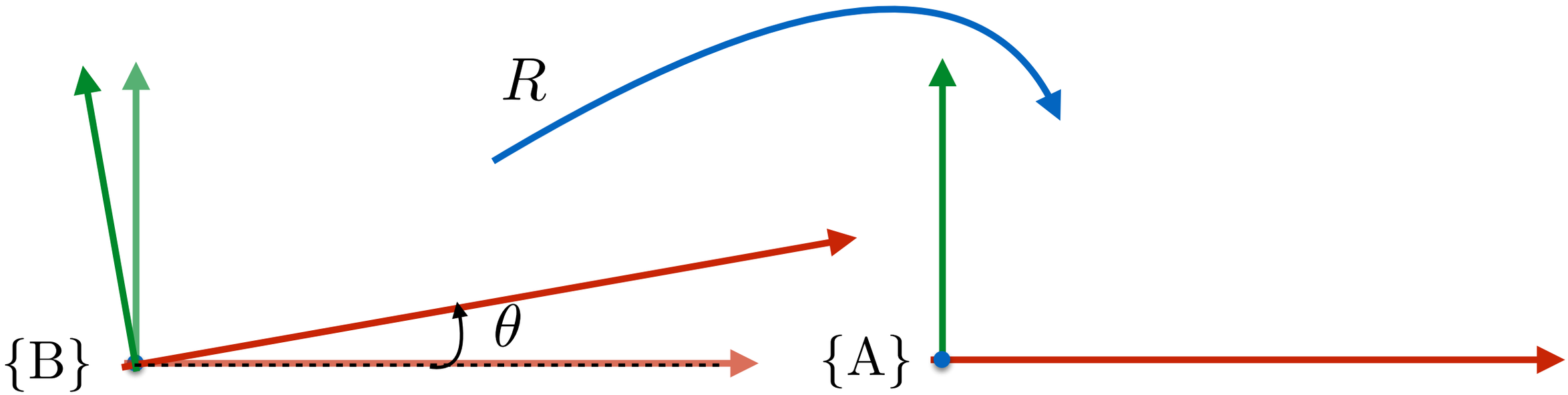
$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \rightarrow \begin{bmatrix} e_{at} \\ e_{ct} \\ \theta_e \end{bmatrix}$$

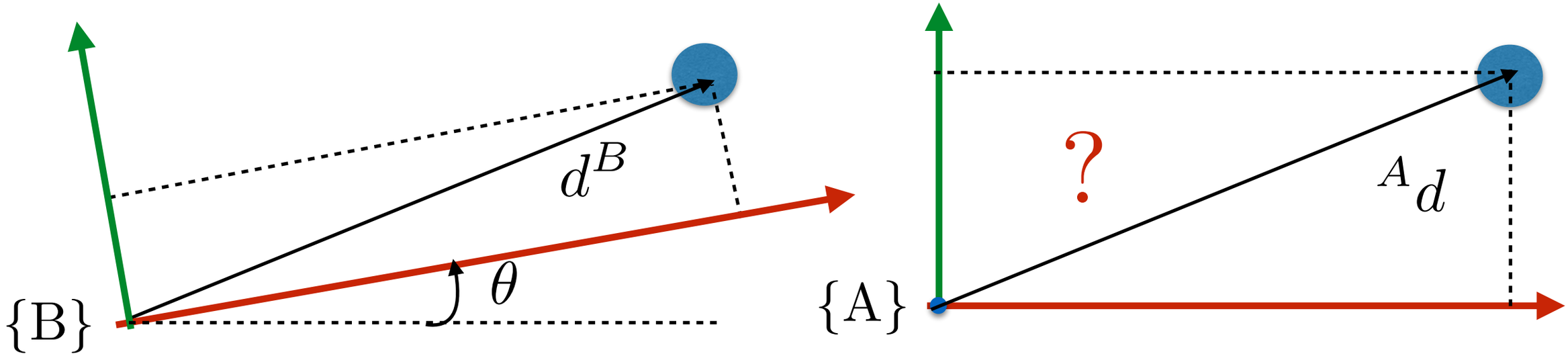
Aside: Rotation Matrices (Plane)



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

Express Position to desired Frame

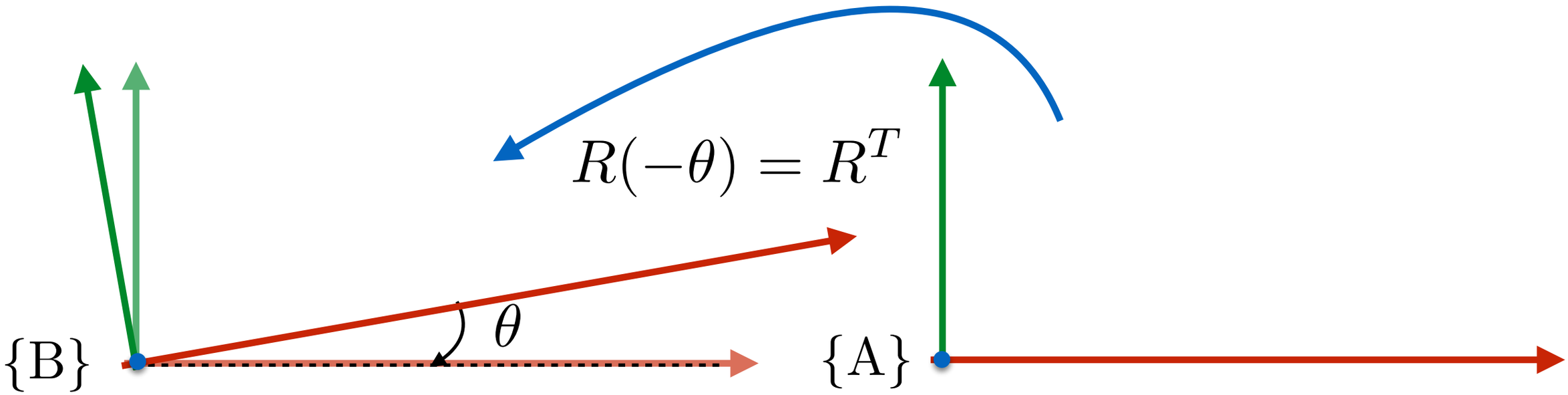


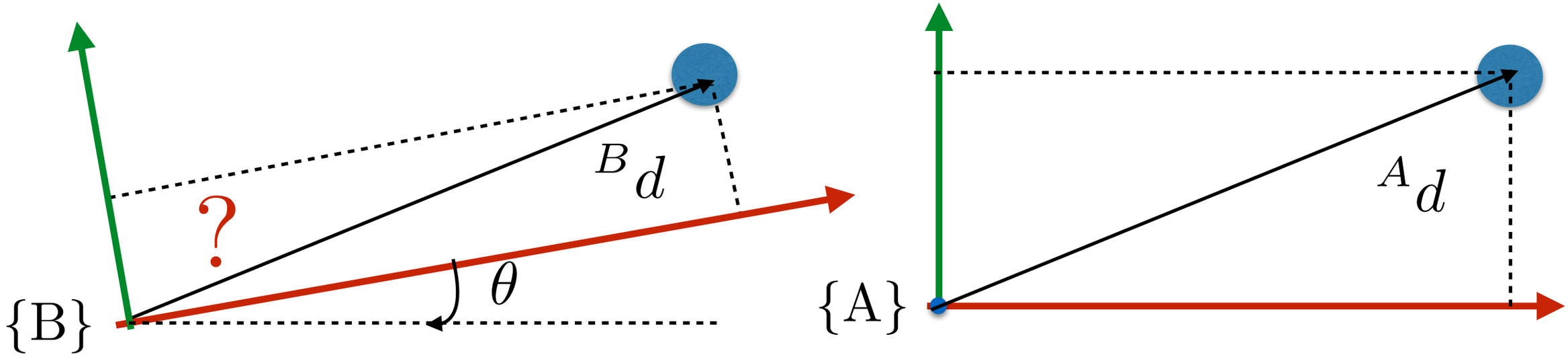


$$R = R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$${}^A d = R^B d$$

Inverse Transformation

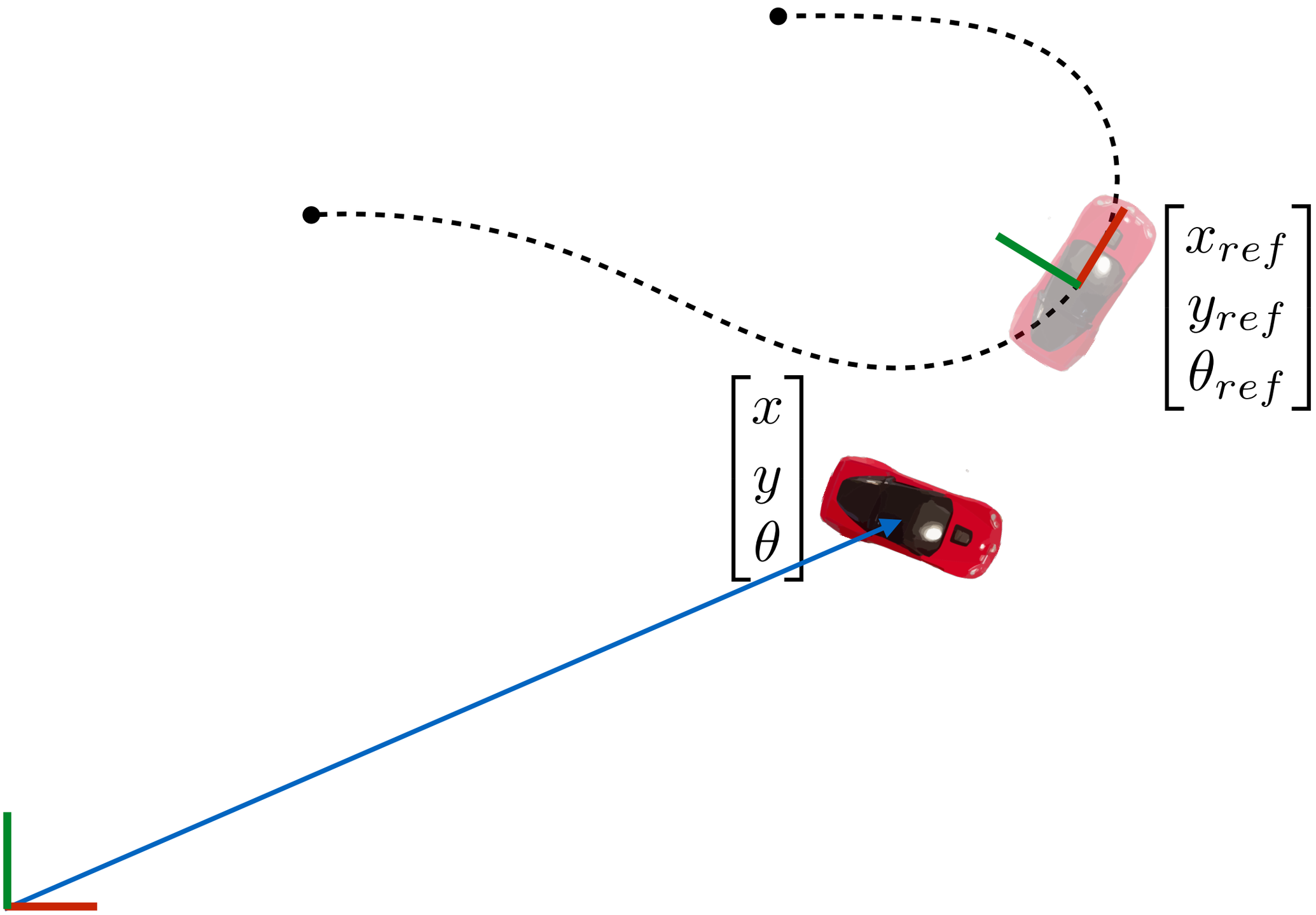




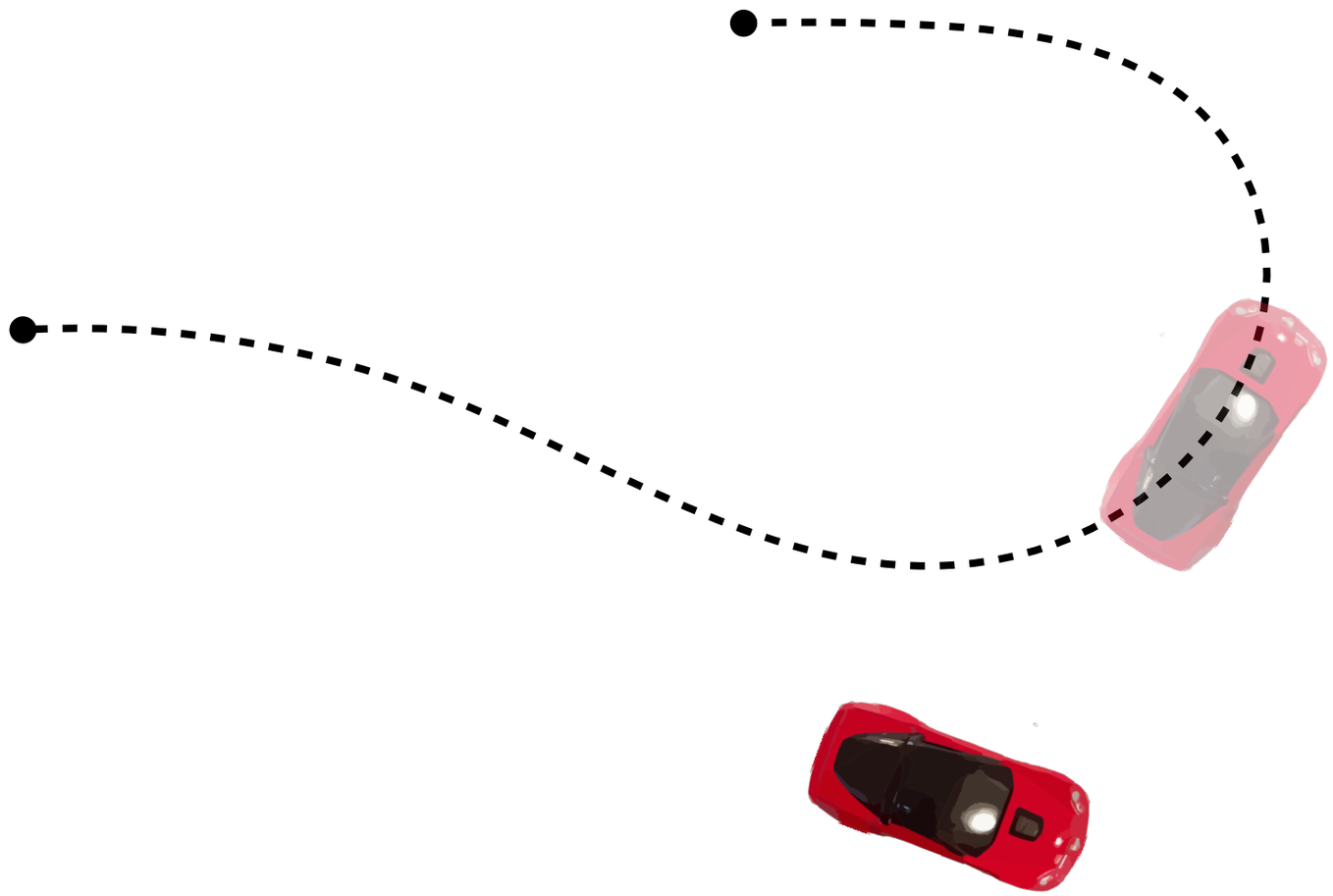
$$R^T = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

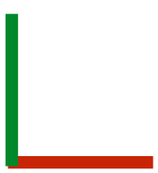
$${}^B d = R^T {}^A d$$

Step 3: Compute error to this state

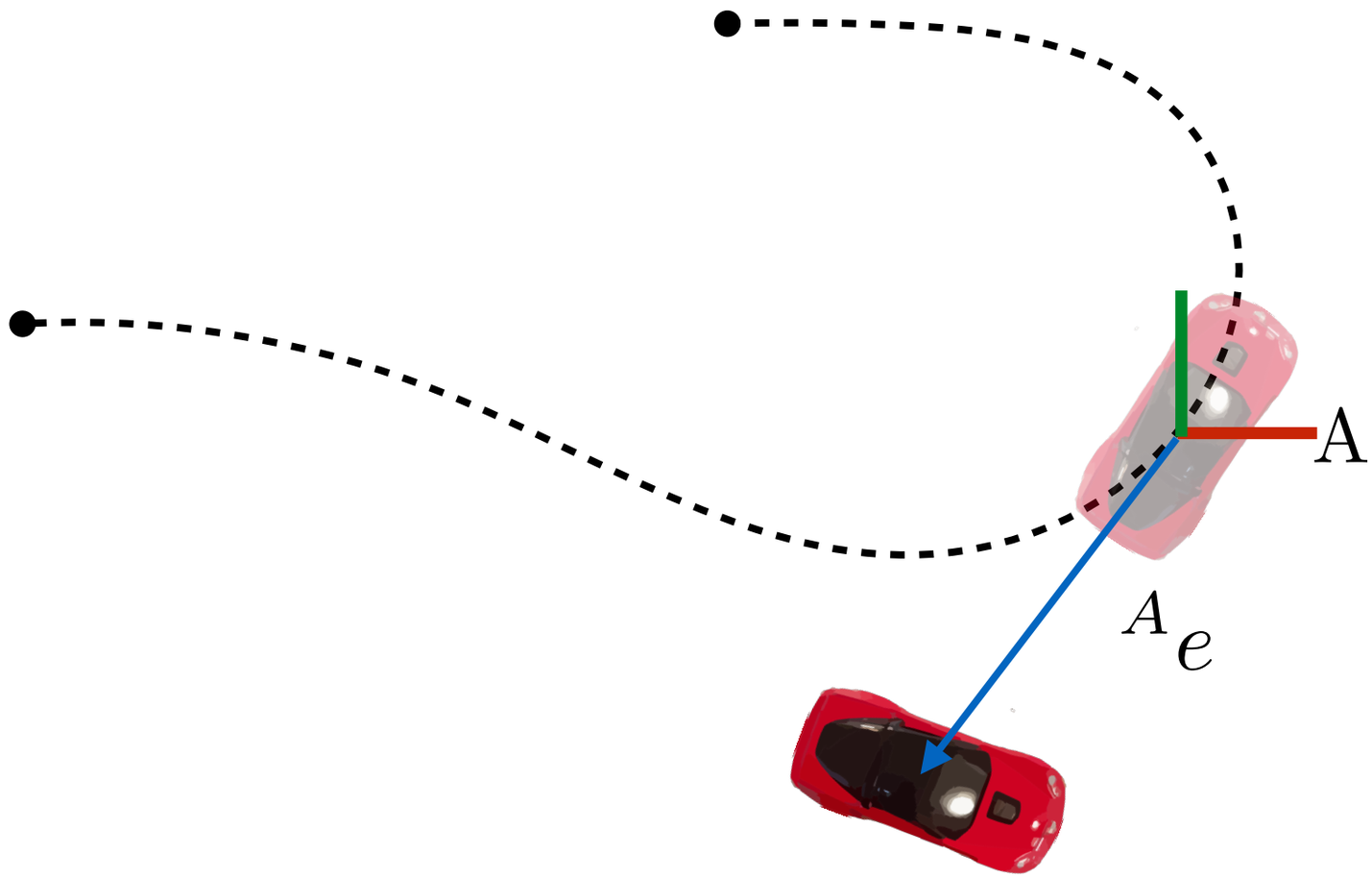


Step 3: Compute error to this state





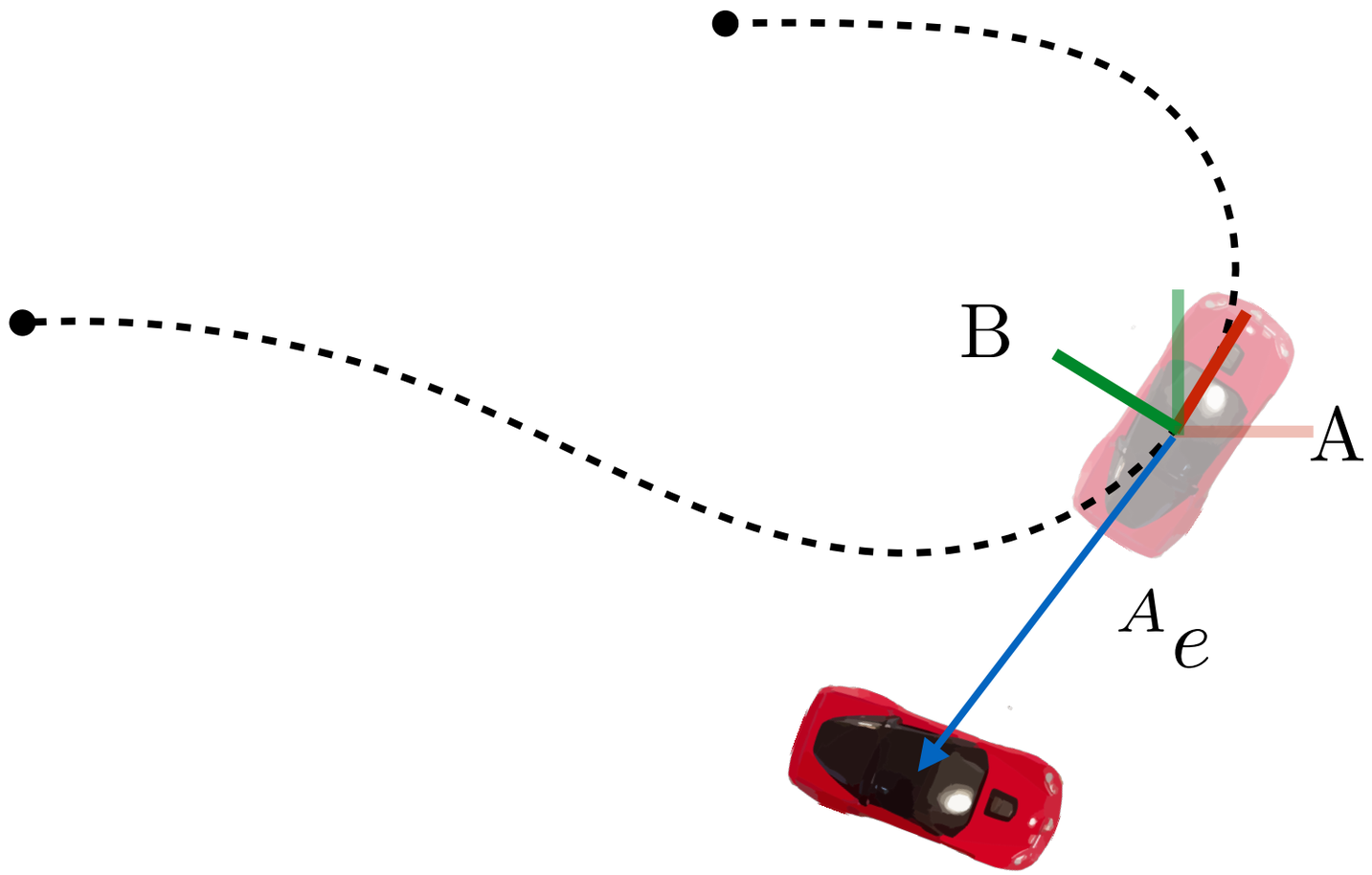
Step 3: Compute error to this state



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



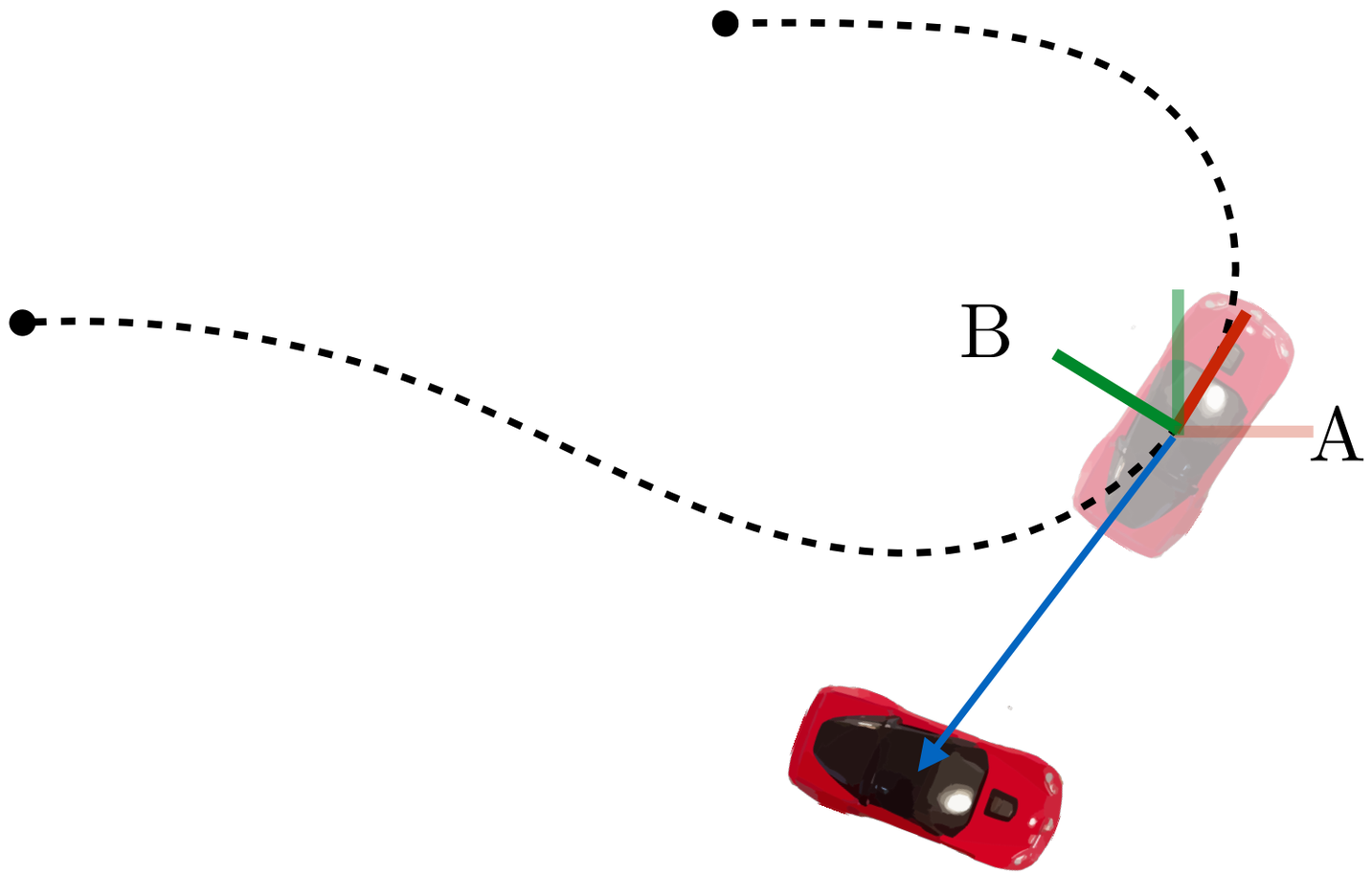
We want position in frame B

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

(rotation of A w.r.t B)

(rotation of A w.r.t B)

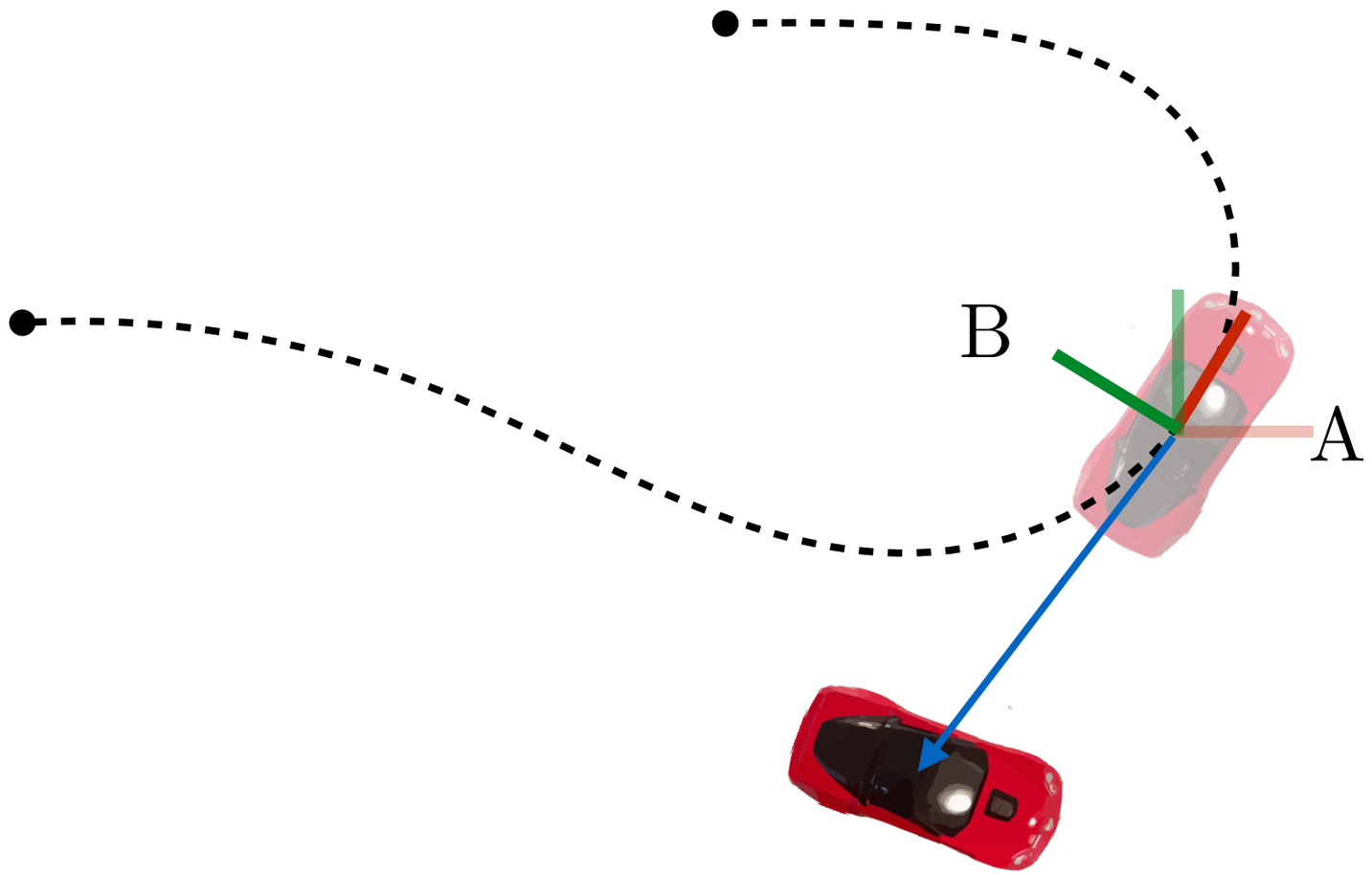
Step 3: Compute error to this state



We want position in frame B

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

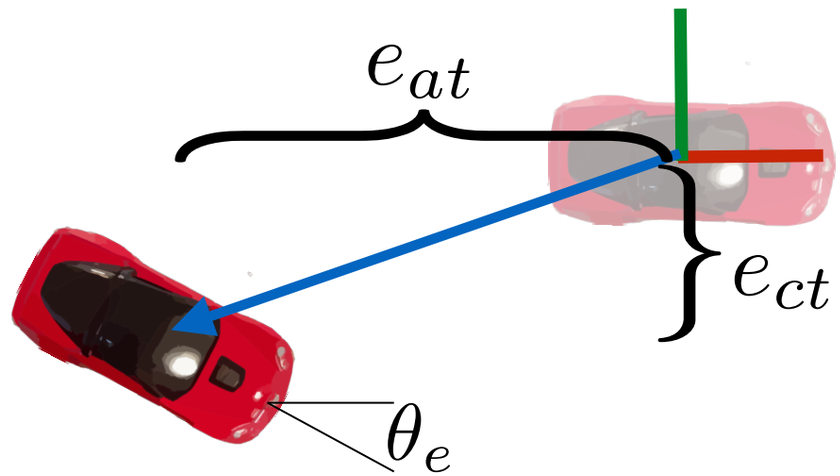
Step 3: Compute error to this state



Heading error

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

Step 3: Compute error to this state

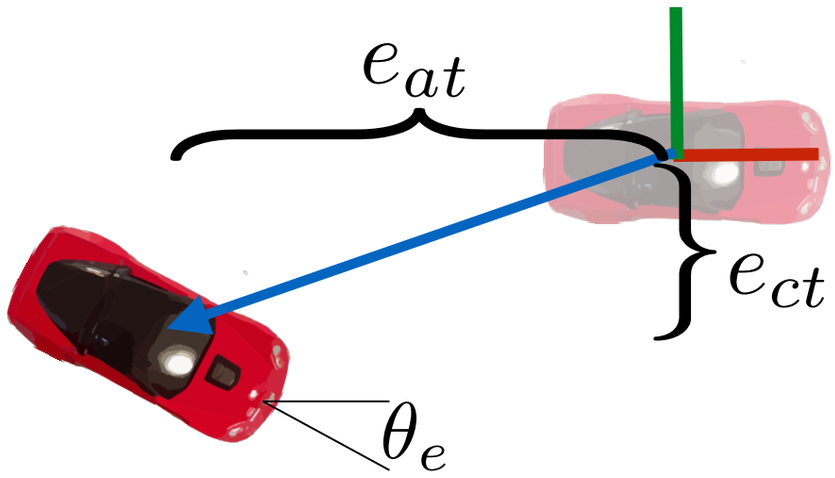


$$\text{(Along-track)} \quad e_{at} = \cos(\theta_{ref})(x - x_{ref}) + \sin(\theta_{ref})(y - y_{ref})$$

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

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

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

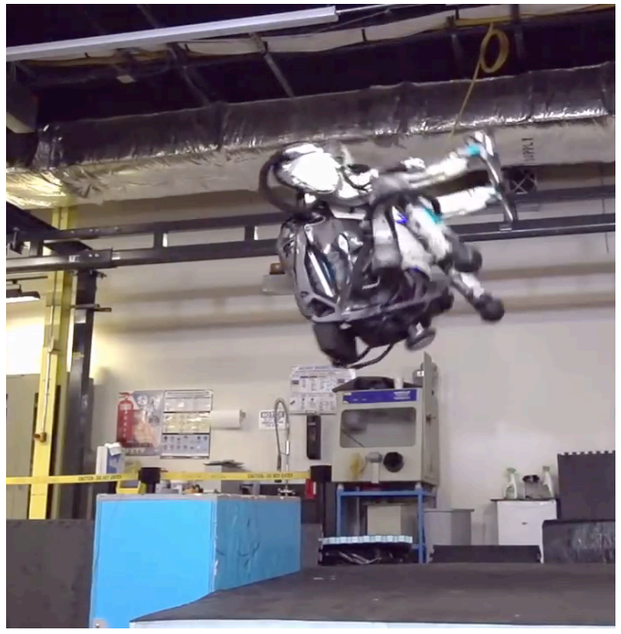
4. LQR

5. MPC

Proportional–integral–derivative (PID) controller

Used widely in industrial control from 1900s

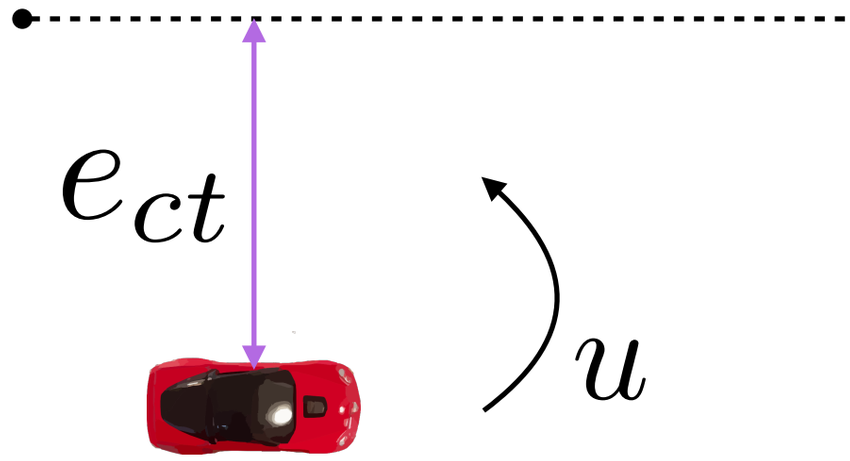
Regulate temp, press, speed etc



Do not try this with PID!!!

PID control overview

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



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

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

Some intuition ...

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

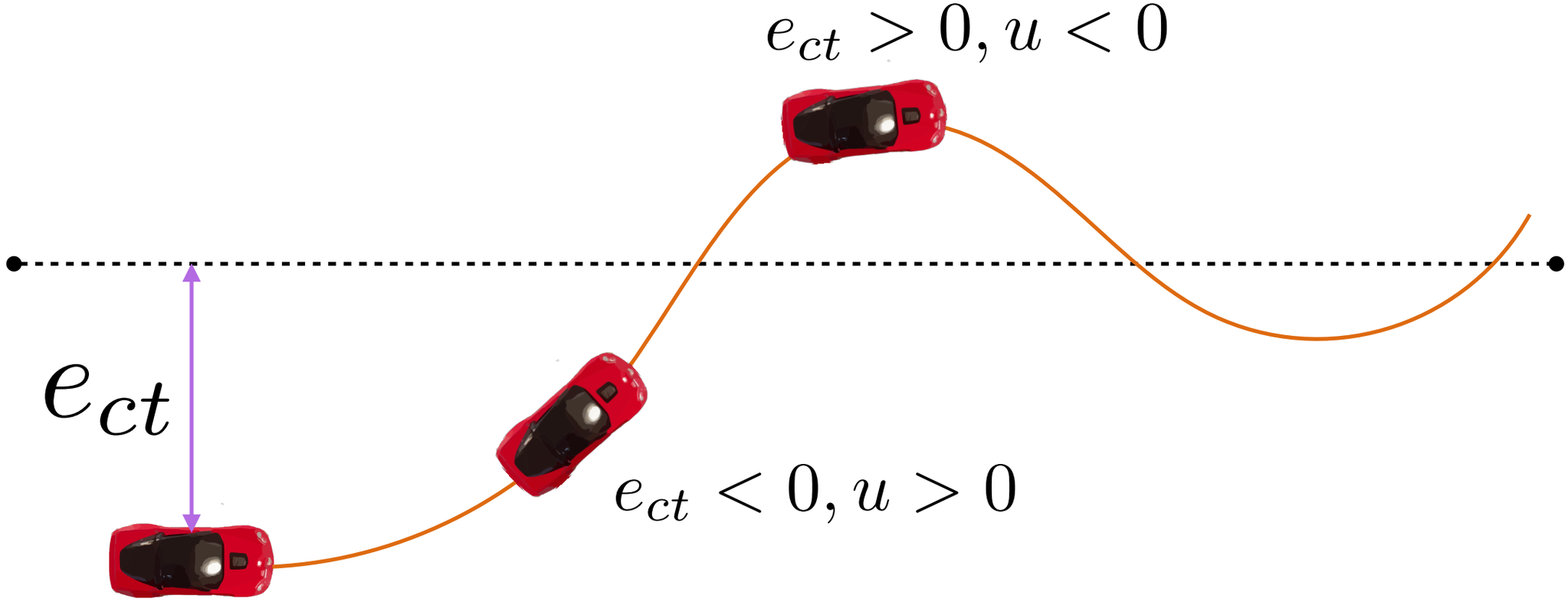
Proportional (current) | Integral (past) | Derivative (future)

Proportional - get rid of the current error!

Integral - if I am accumulating error, try harder!

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

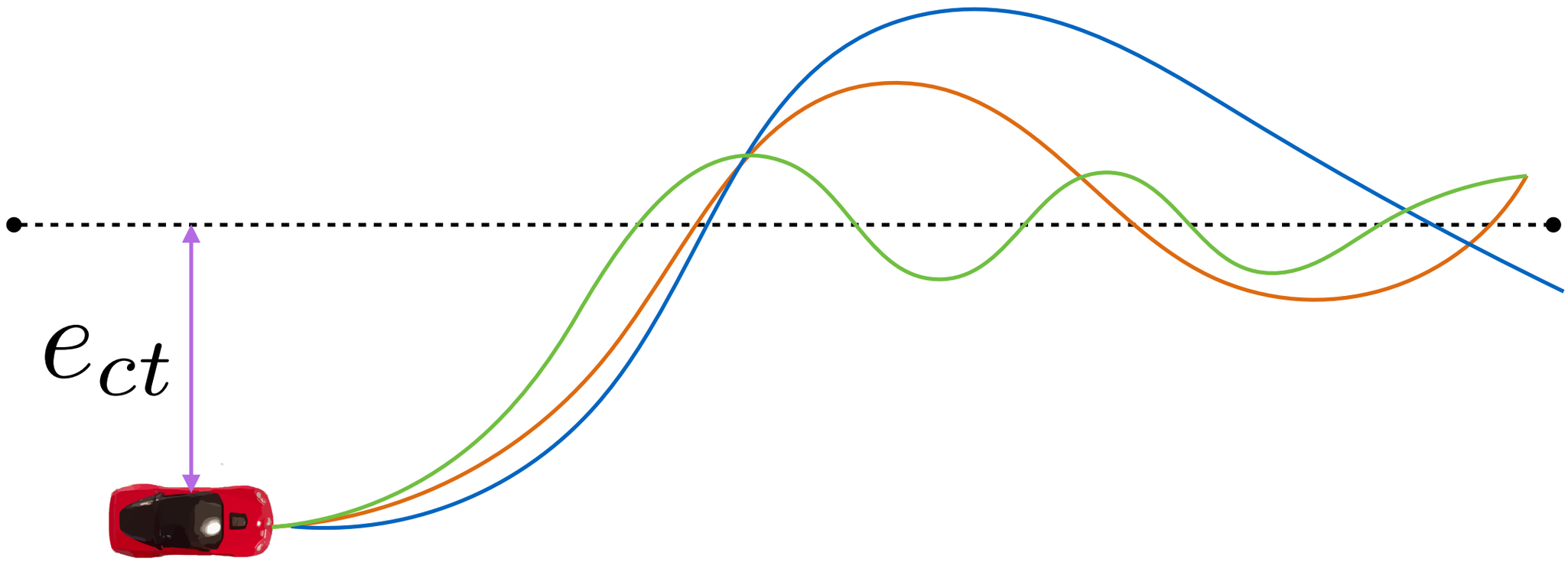
Proportional control



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

(Gain)

The proportional gain matters!

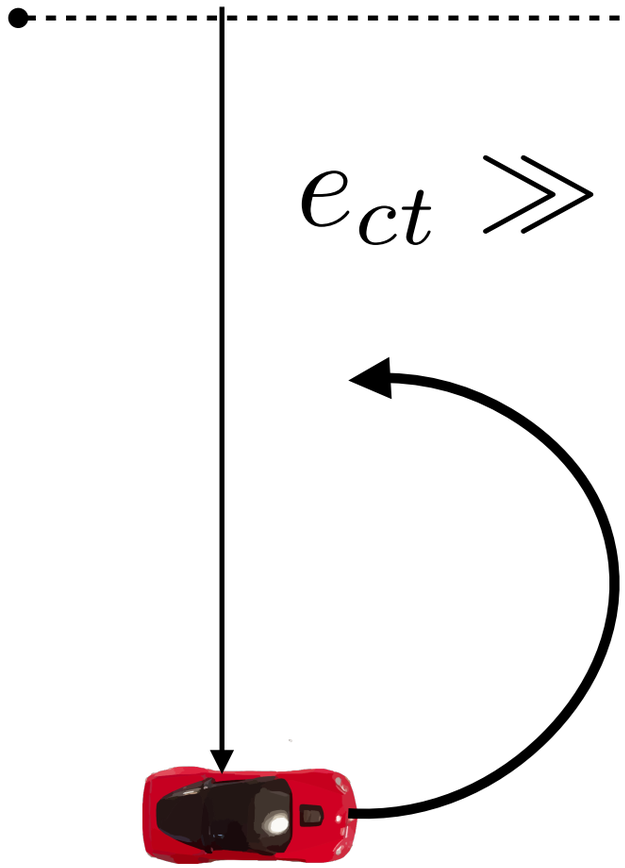


What happens when gain is low?

What happens when gain is high?

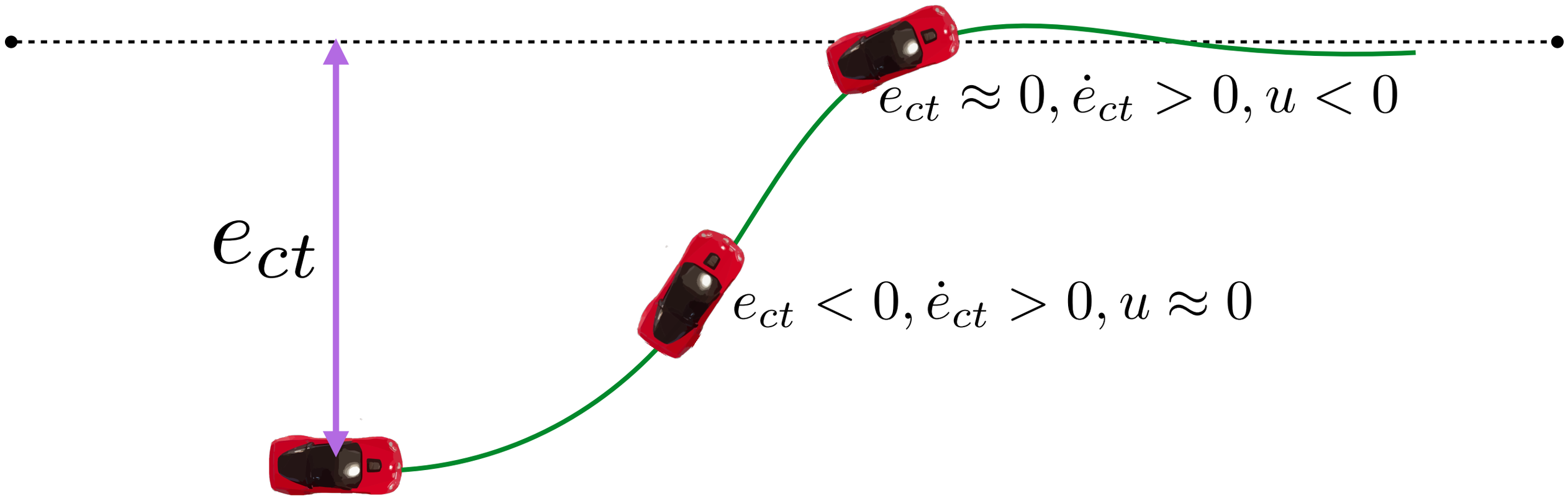
Proportional term

What happens when gain is too high?



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

Proportional derivative control



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

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

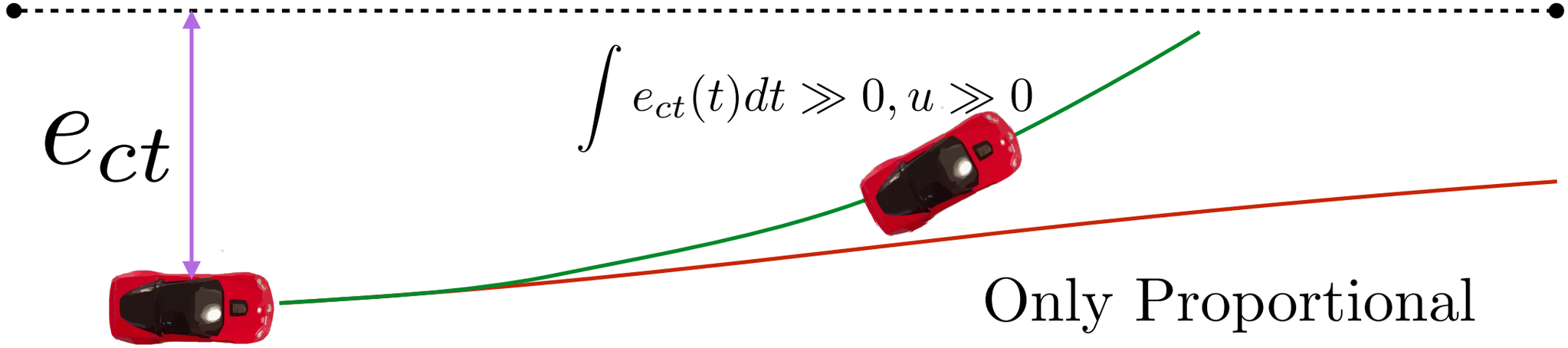
$$e_{ct} = -\sin(\theta_{ref})(x - x_{ref}) + \cos(\theta_{ref})(y - y_{ref})$$

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

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

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

Proportional integral control



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

Different control laws

1. PID control

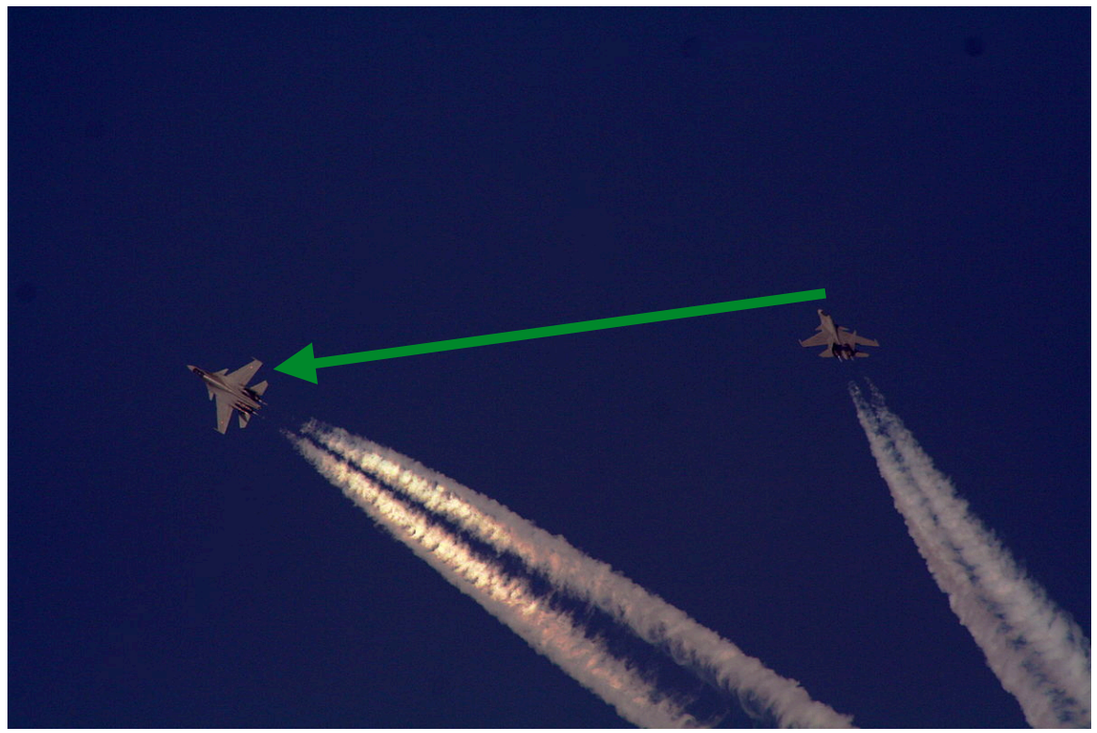
2. Pure-pursuit control

3. Lyapunov control

4. LQR

5. MPC

Pure Pursuit Control



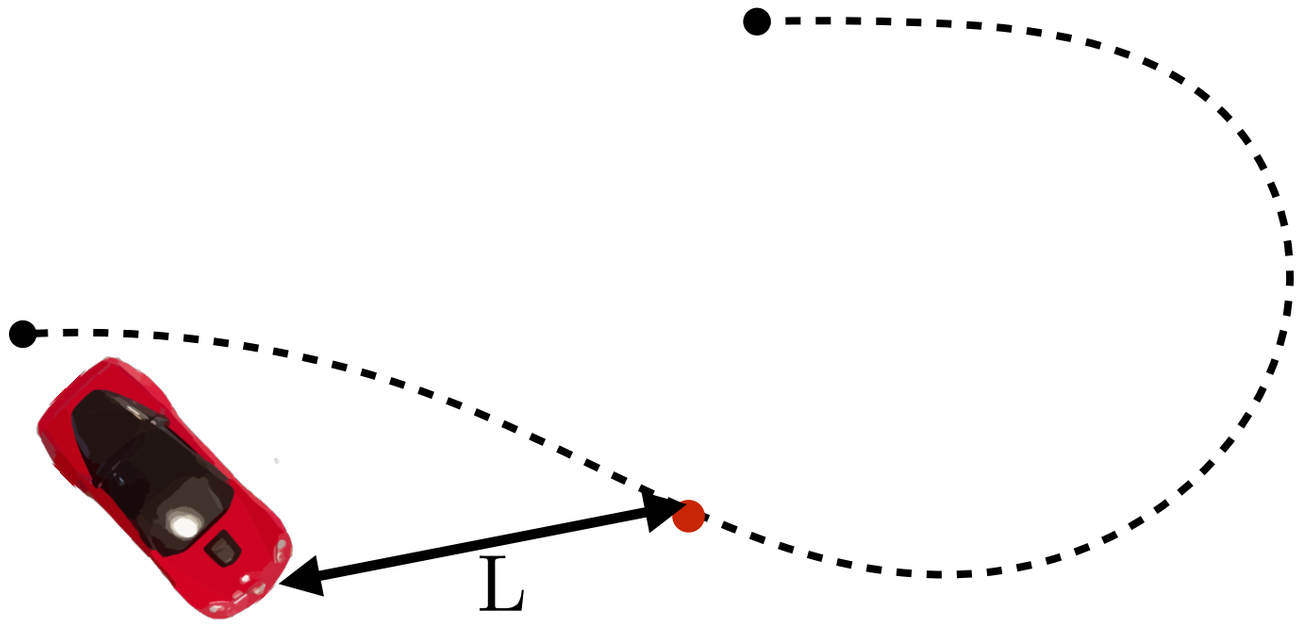
Aerial combat in which aircraft pursues another aircraft by pointing its nose directly towards it

Similar to carrot on a stick!

Key Idea:

The car is always moving in a circular arc

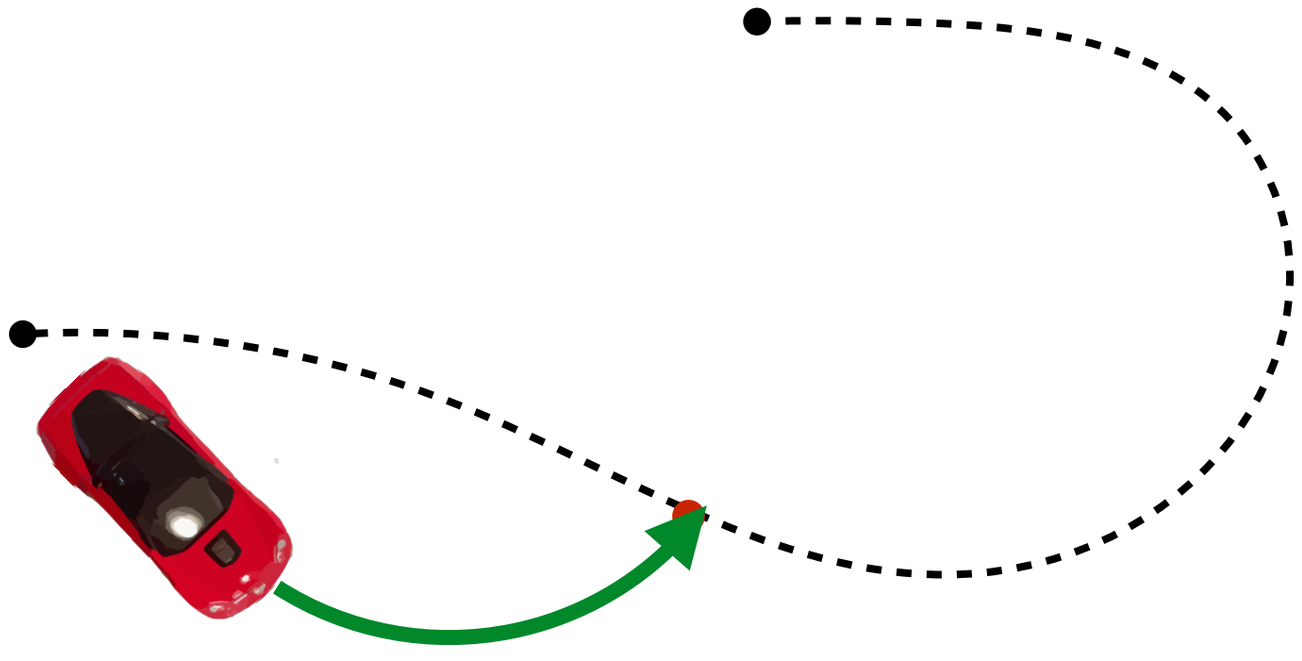
Consider a reference at a lookahead distance



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

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

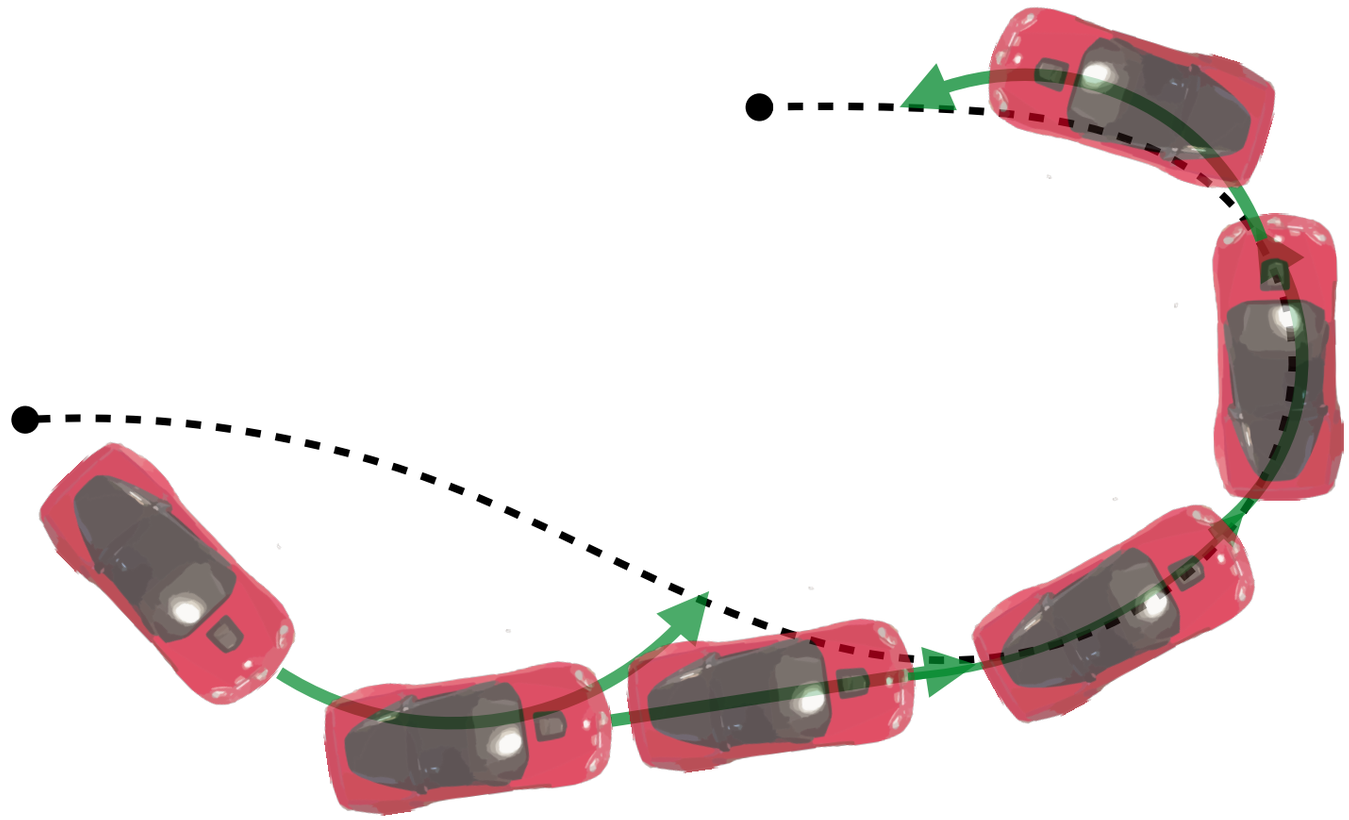
Solution: Compute a circular arc



We can always solve for an 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

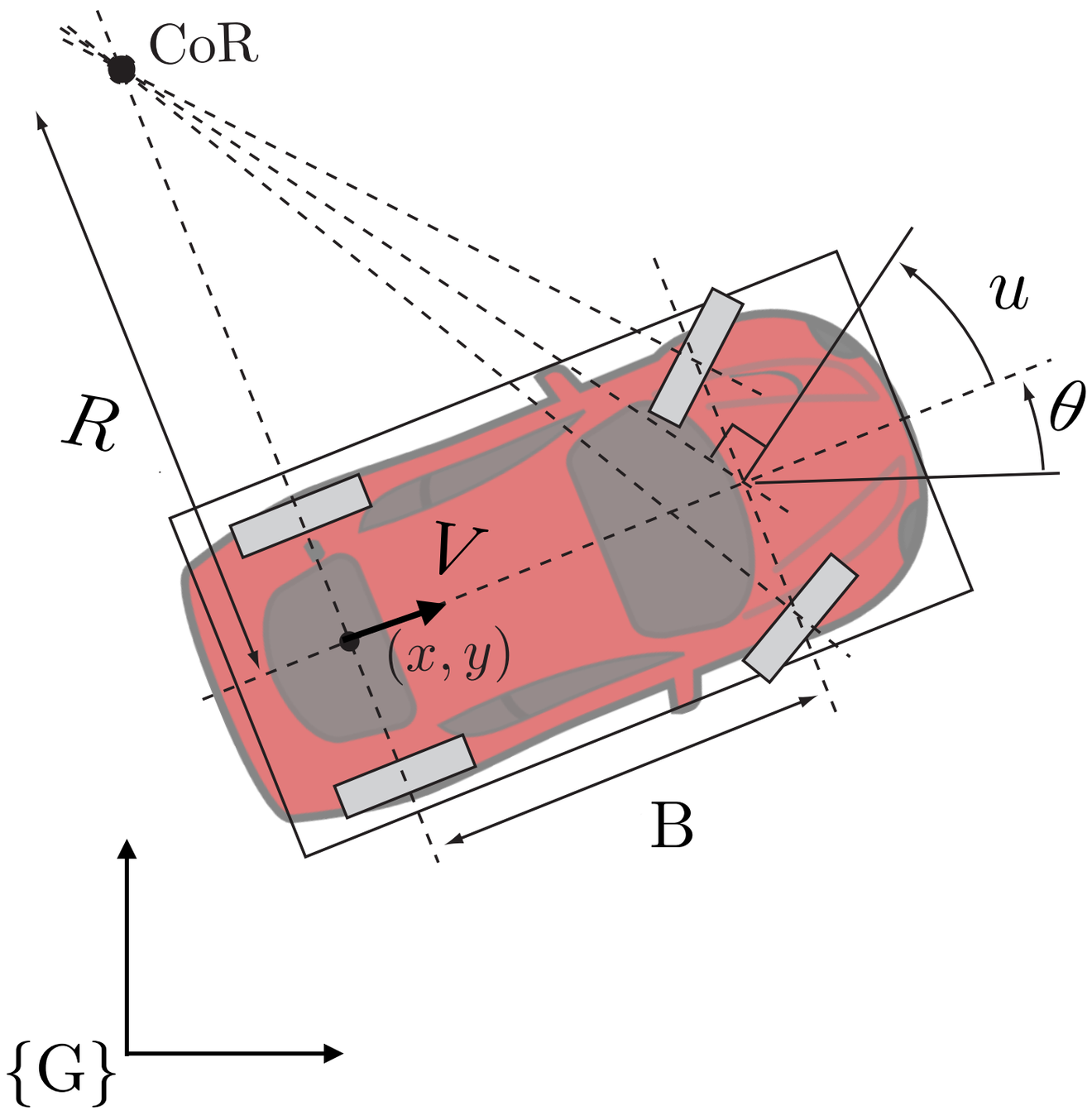
2. Move along the arc

3. Go to step 1

Remember Simple Car Kinematics

$$\tan u = \frac{B}{R}$$

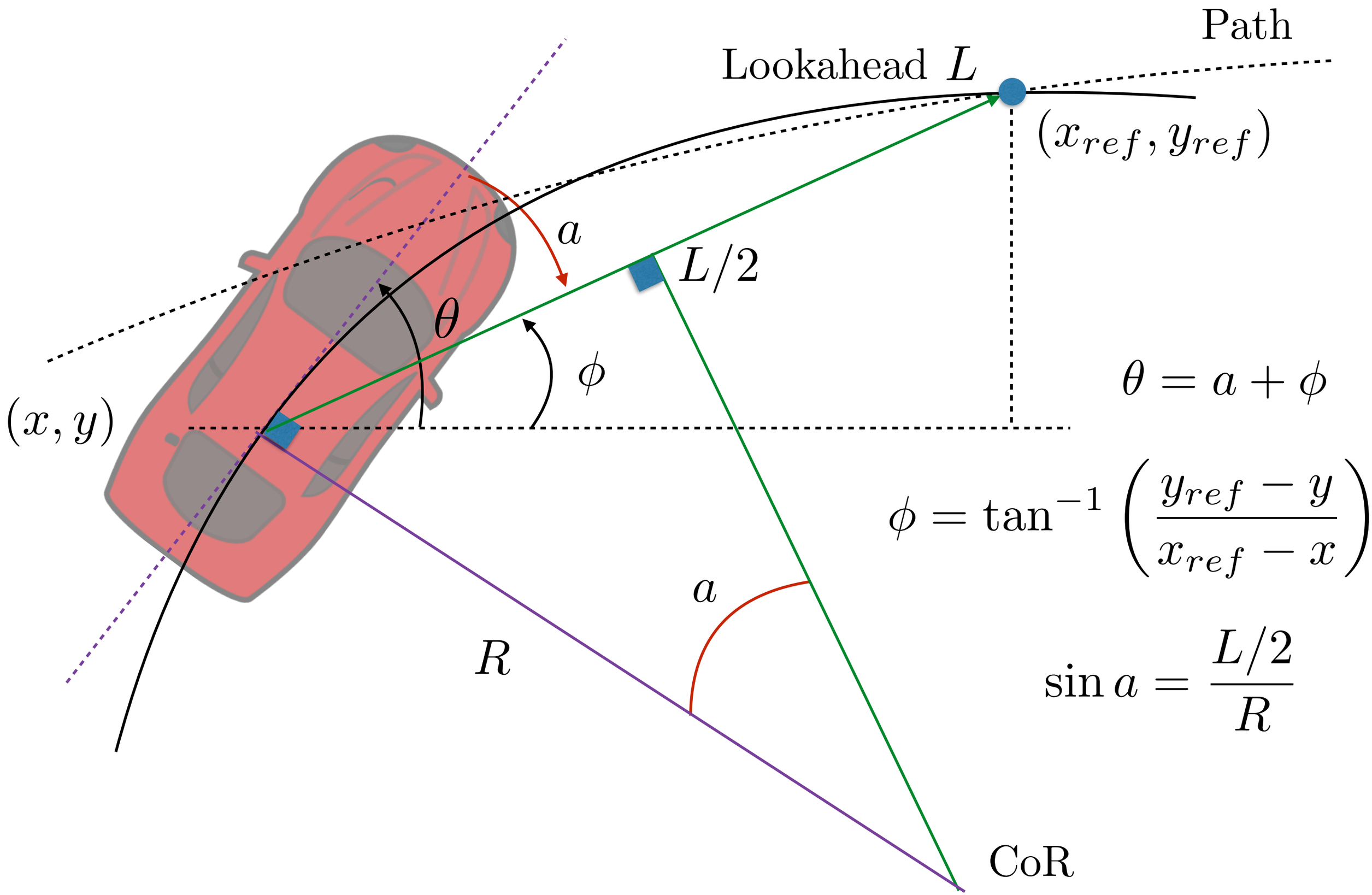
$$R = \frac{B}{\tan u}$$



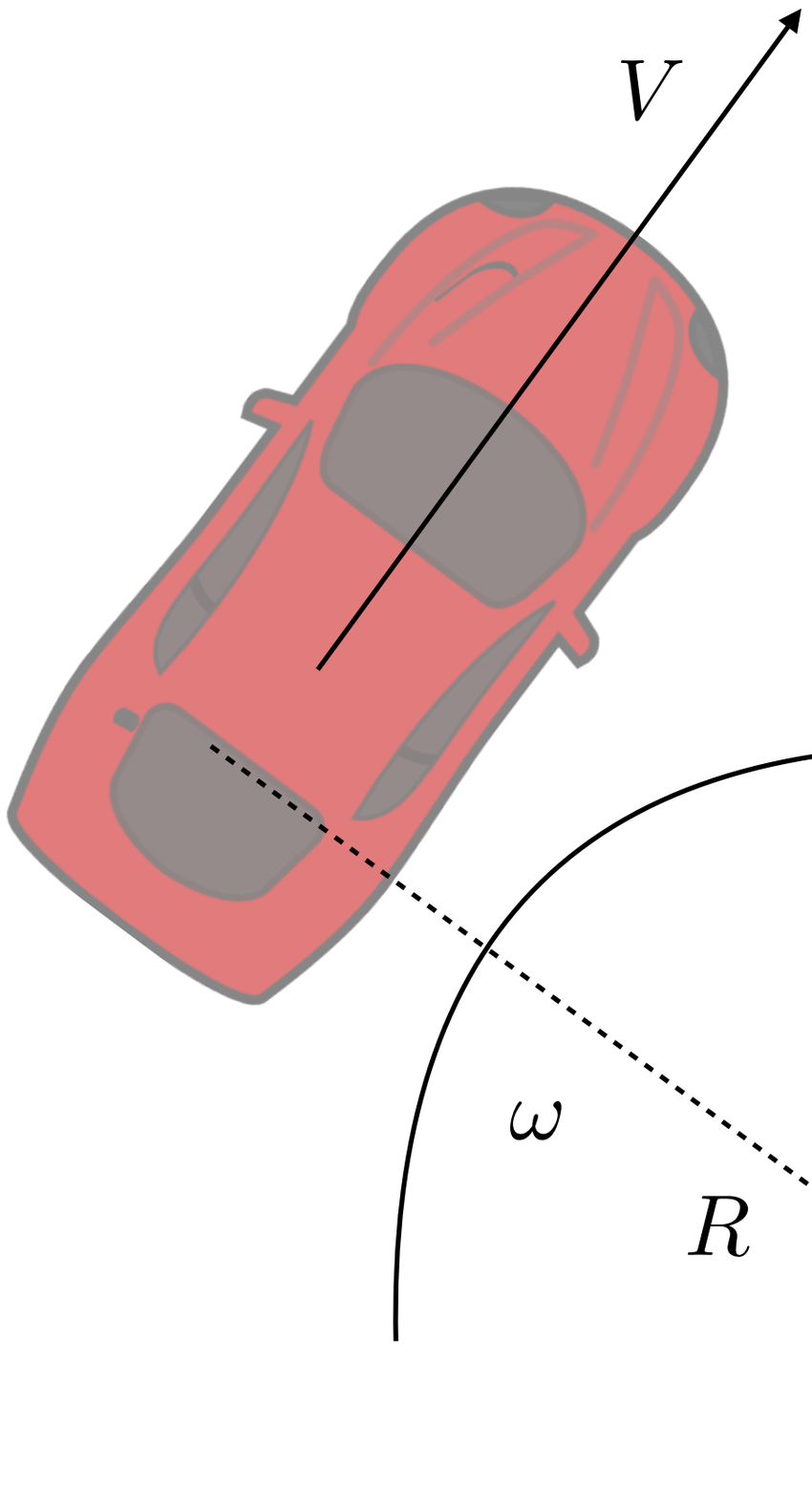
$$\omega = \frac{V}{R} = \frac{V \tan u}{B}$$

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

Chasing the Lookahead

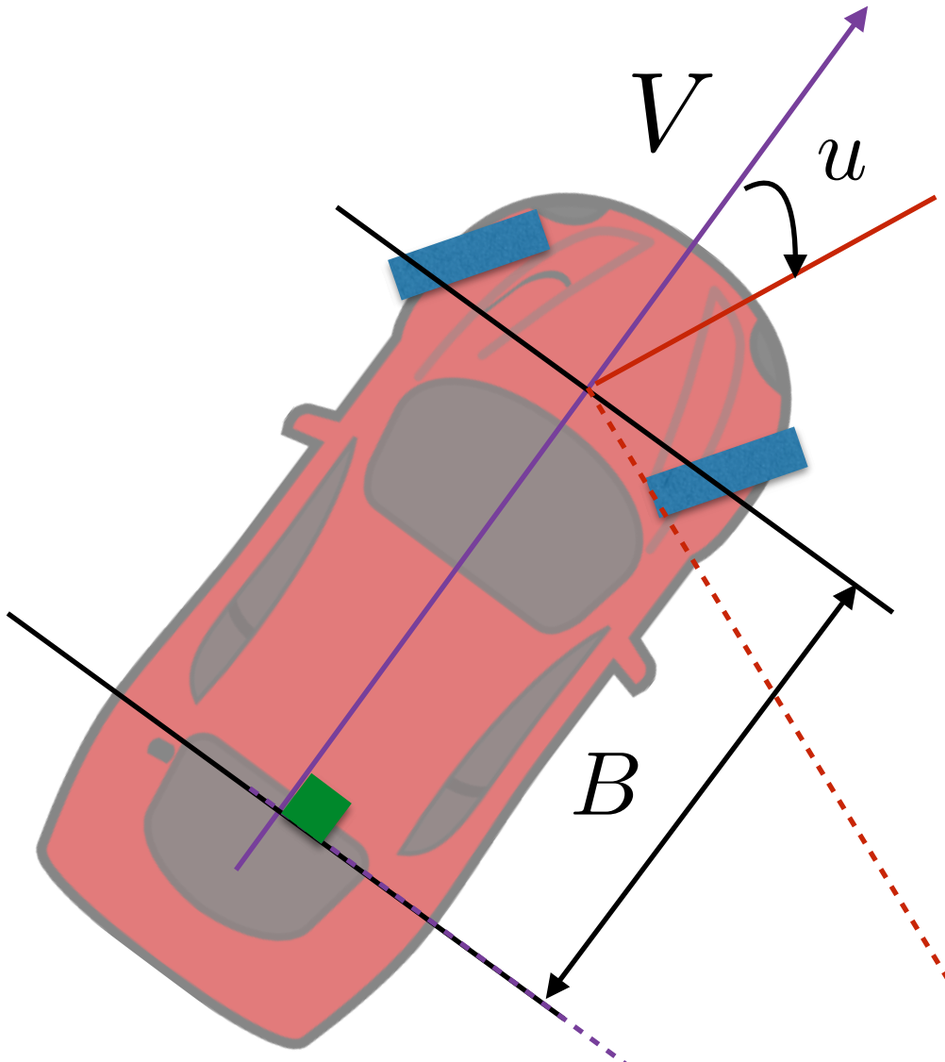


Rigid Body Rotation



$$\left.\begin{aligned} \dot{\theta} = \omega = \frac{V}{R} \\ \sin a = \frac{L/2}{R} \end{aligned}\right\} \dot{\theta} = \omega = \frac{2V \sin a}{L}$$

Rear-Axle Car Kinematics



$$\left.\begin{aligned}\tan u &= \frac{B}{R} \\ \sin a &= \frac{L}{2R}\end{aligned}\right\} \tan u = \frac{2B \sin a}{L}$$

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

Question: How do I choose L?