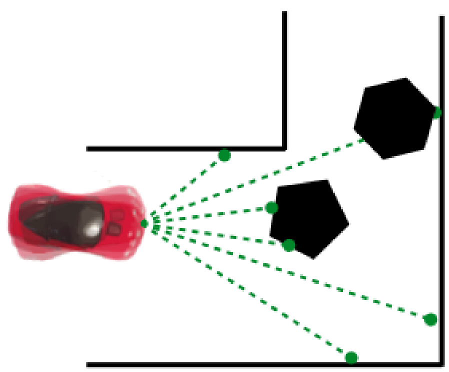
CMPE 185 Autonomous Mobile Robots

Navigation and Control Part 3

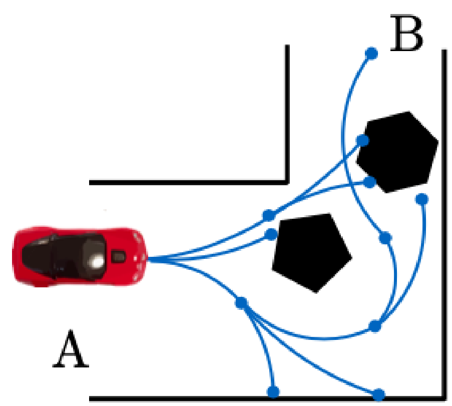
Dr. Wencen Wu
Computer Engineering Department
San Jose State University

Control for Mobile Robots

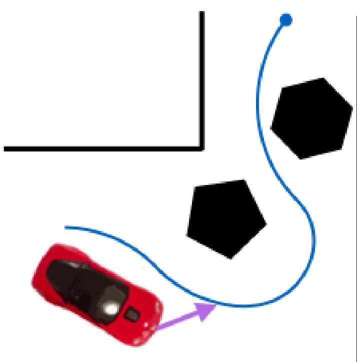
Estimate state



Plan a sequence of motions



Control robot to follow path

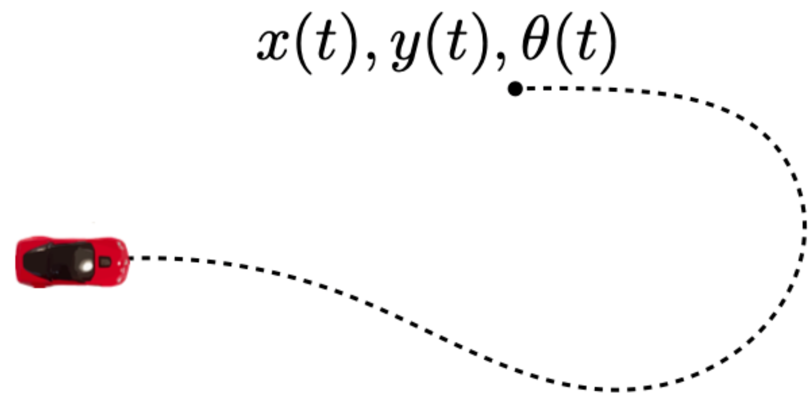


- Robot pose known
- Path is given

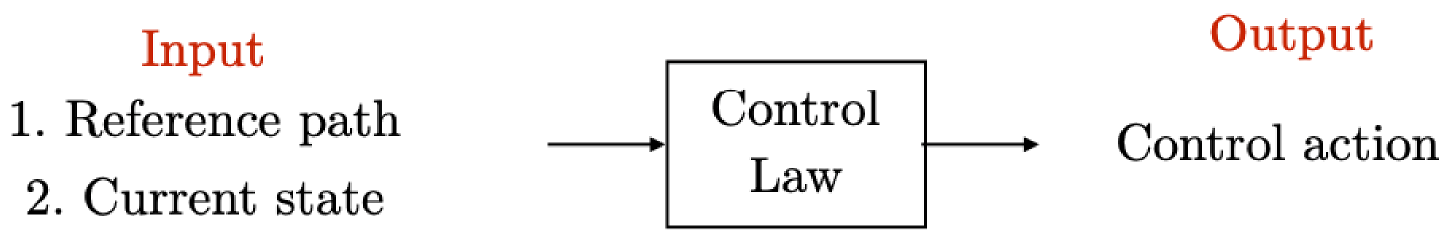
Some slides are adopted from Washington CS478

The Control Framework

- Let’s say we want to track a reference trajectory

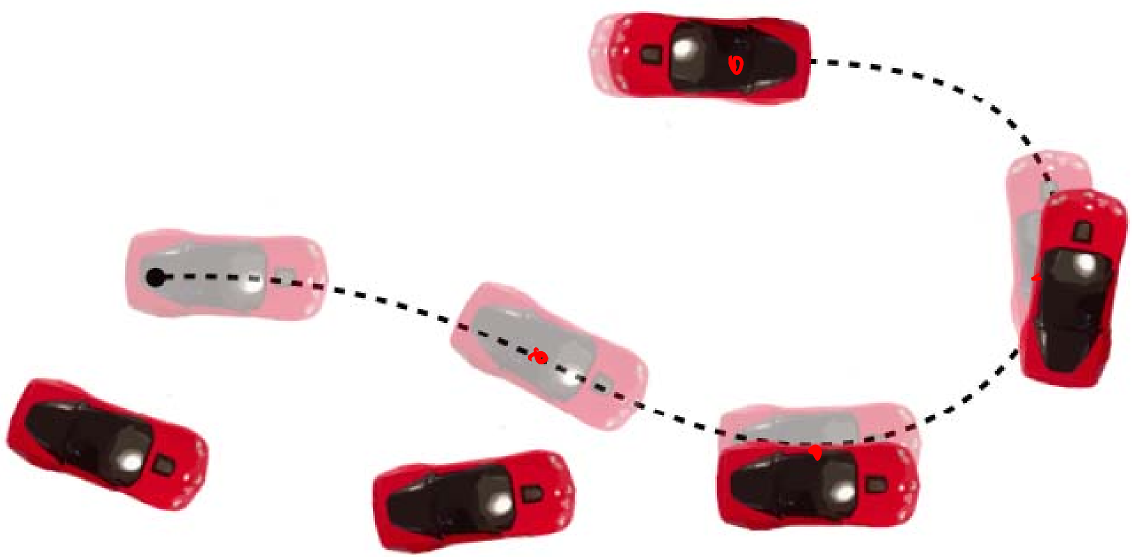


- Objective: Figure out a control trajectory u(t) to achieve this.



- We will focus on steering angle as the control input

Rough Idea of What Happens Across Timesteps



- Robot is trying to track a desired state on the reference path
  - i.e., take an action to drive down error between desired and current state

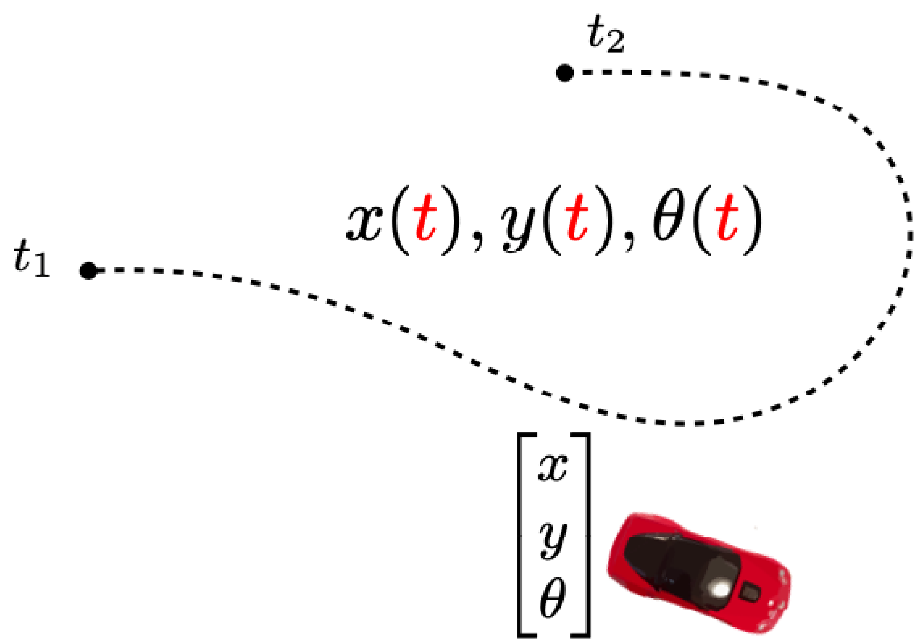
Steps to Designing a Controller

- Get a reference path / trajectory to track
- Pick a point on the reference
- Compute error to reference point
- Compute control law to minimize error

Step 1: Get a Reference Path

How do we define a reference?

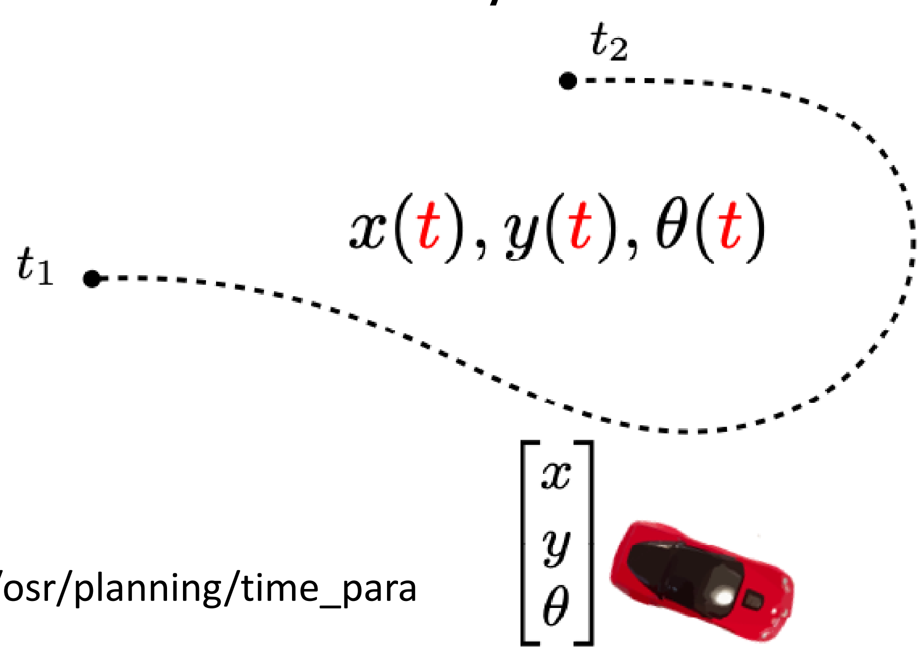
- **Option 1:** Time-parameterized *trajectory*
  - The time parameterization of a path is the problem of transforming this path into a trajectory which respects the physical limits of the robot, e.g. velocity, acceleration and torque limits, and minimizes a specific criterion, e.g. the execution time.

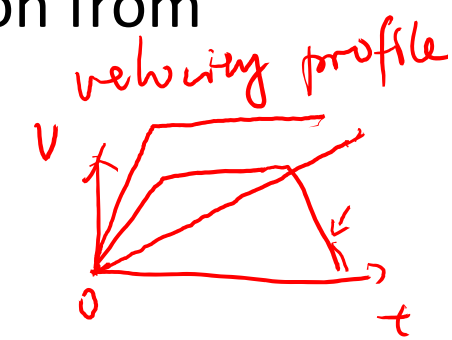


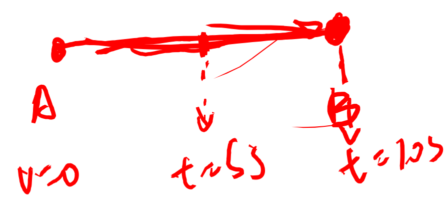
Step 1: Get a Reference Path

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





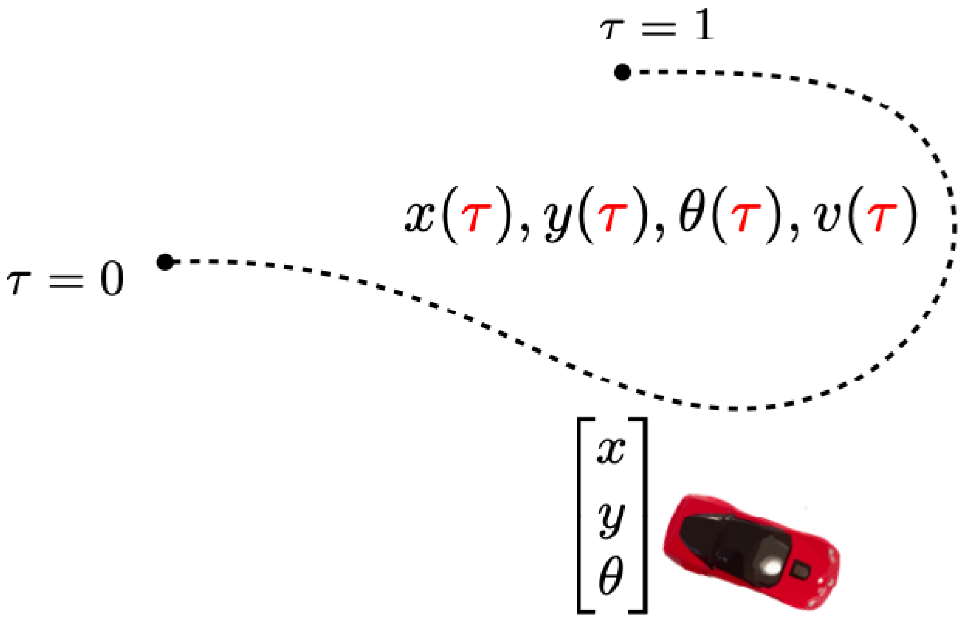


http://www.osrobotics.org/osr/planning/time_parameterization.html

Step 1: Get a Reference Path

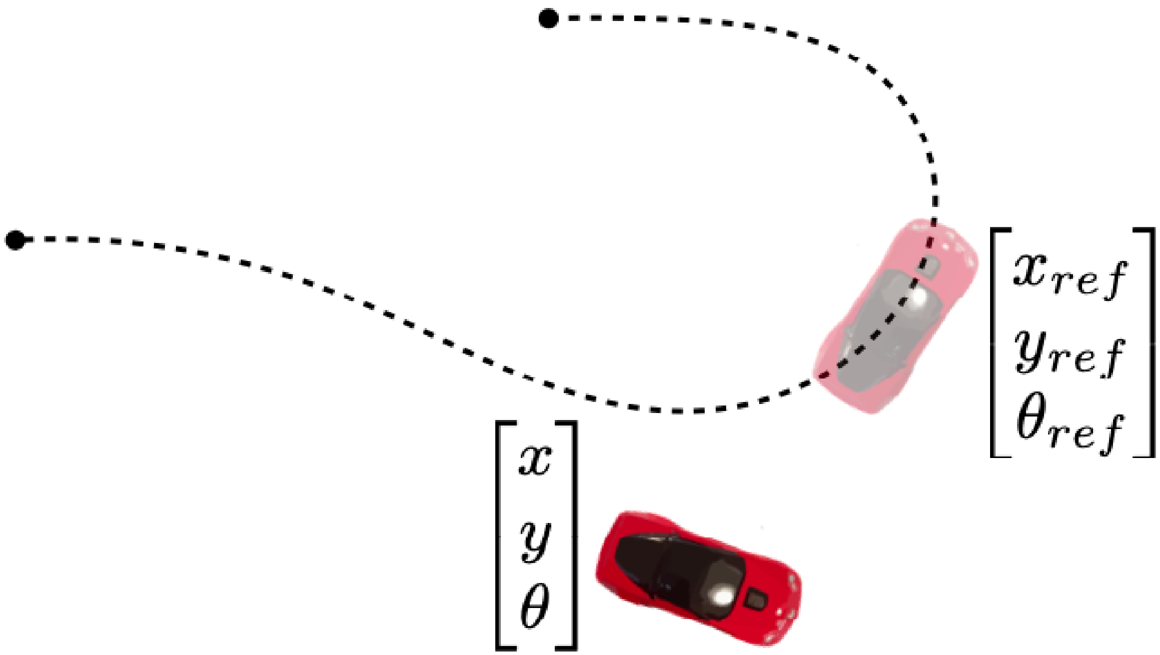
How do we define a reference?

- **Option 2:** Index-parameterized *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

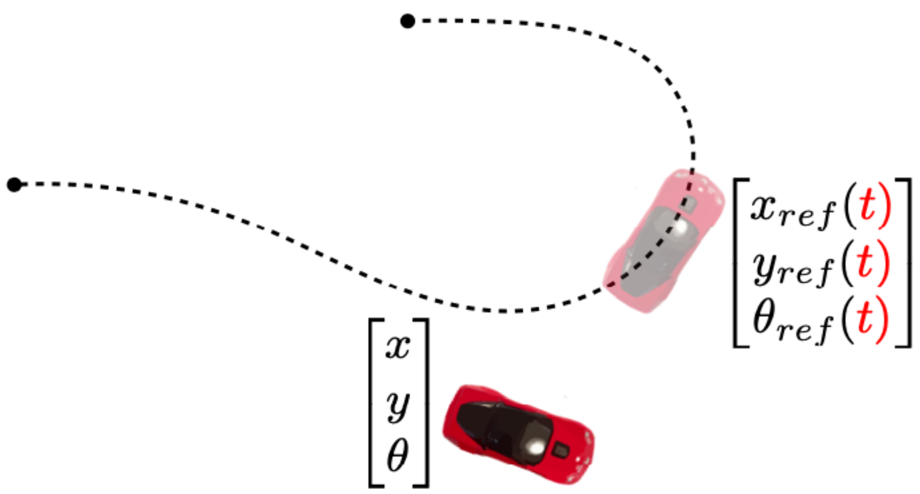
How do we pick a reference?



Step 2: Pick a Reference (desired) State

How do we pick a reference?

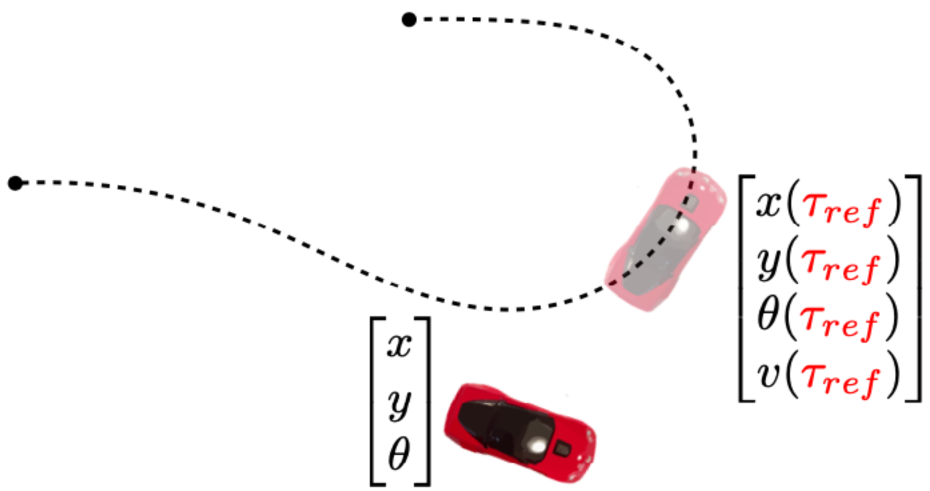
- Option 1: Time-parameterized trajectory



Step 2: Pick a Reference (desired) State

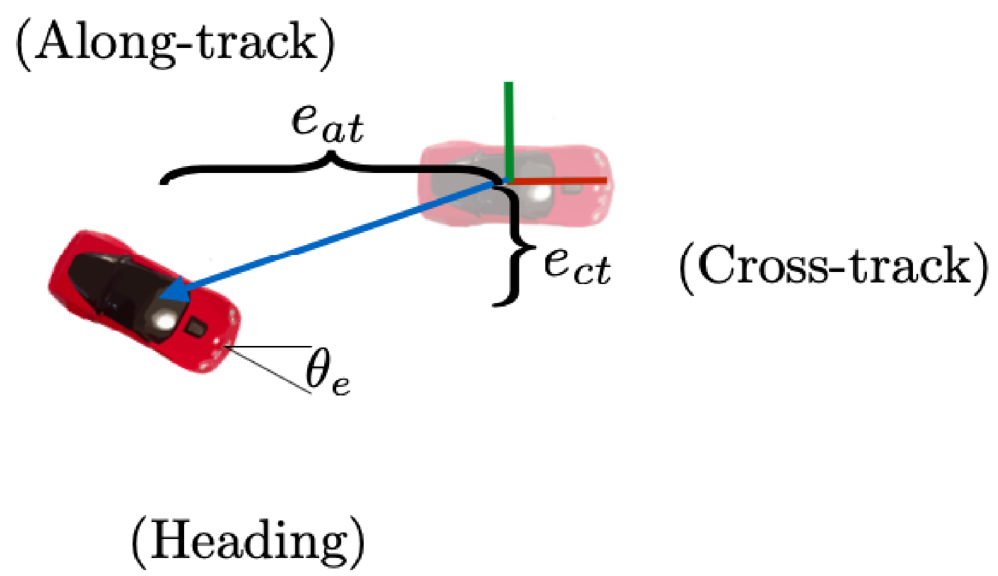
How do we pick a reference?

- Option 2: Index-parameterized path



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

Step 3: Compute Error to This State



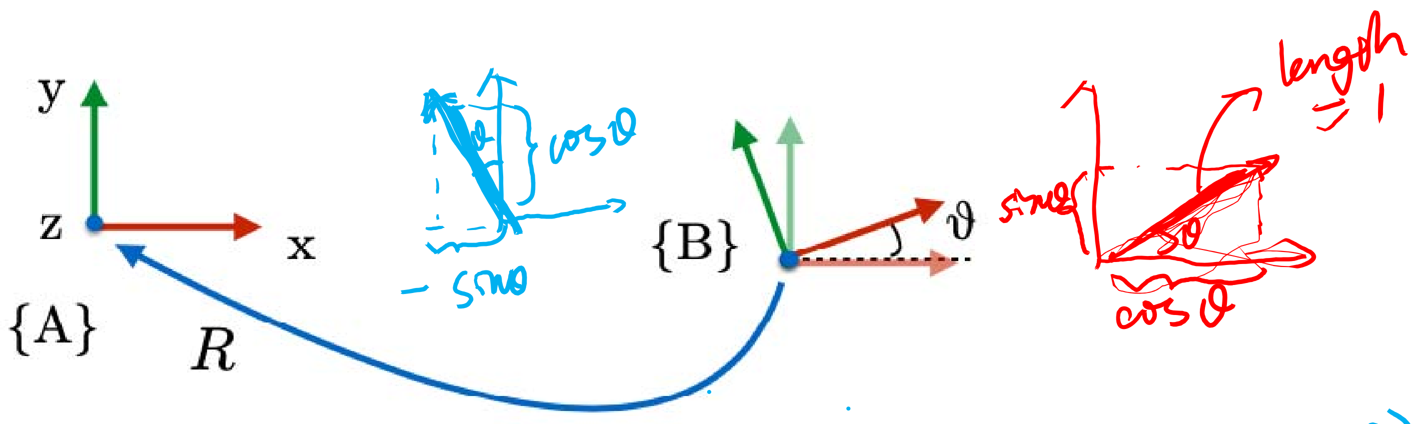
- Error is simply the state of the robot expressed in the frame of the reference (desired) state

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

Recall: Rotation Matrix

- The transformation between the two frames can be described by the rotation matrix ***R***

$$R = \begin{bmatrix} cos\theta & -\ sin\theta \\ sin\theta & con\theta \end{bmatrix} \quad R^{-1} = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix}$$

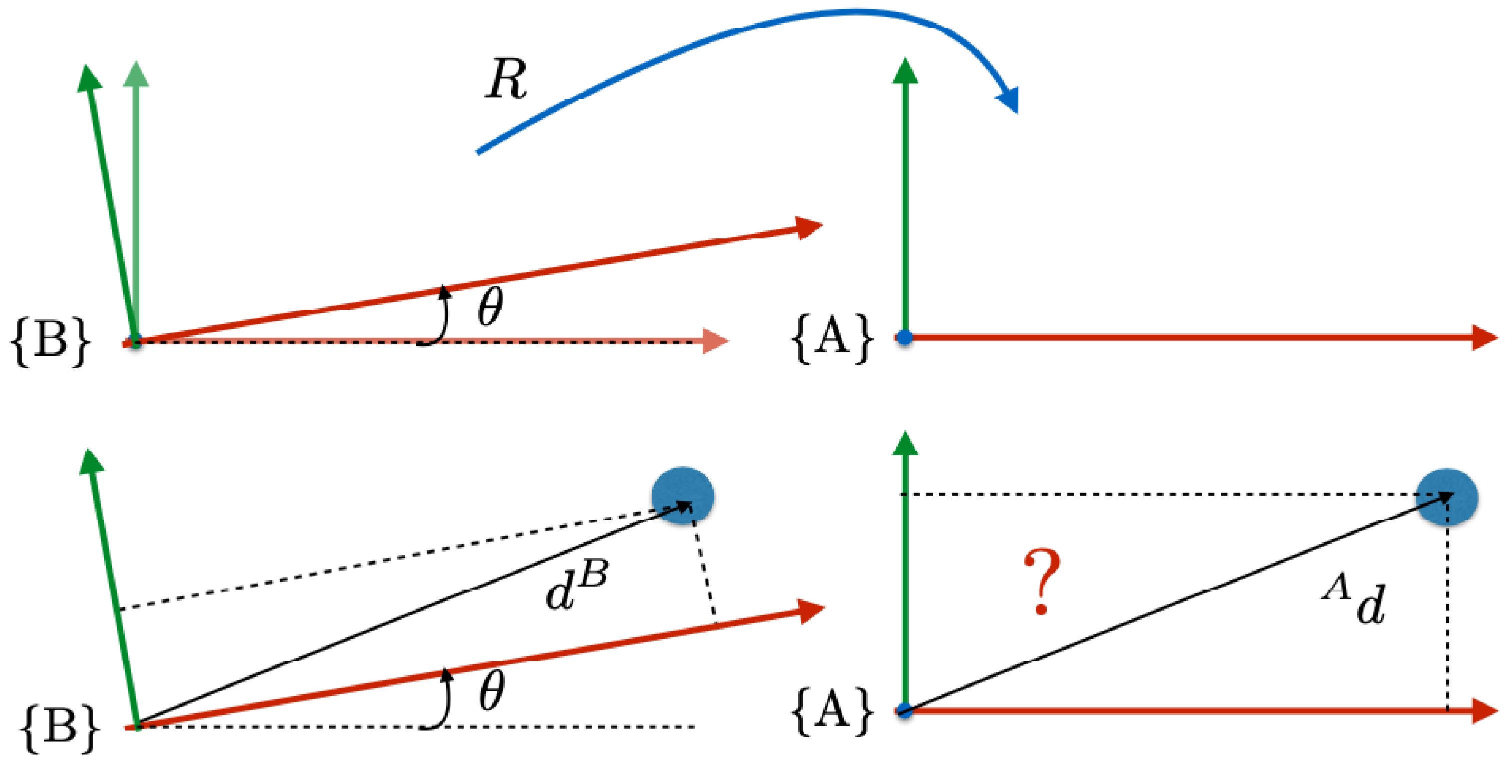


$$\begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix} \quad \begin{pmatrix} -\sin\theta \\ \cos\theta \end{pmatrix}$$

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

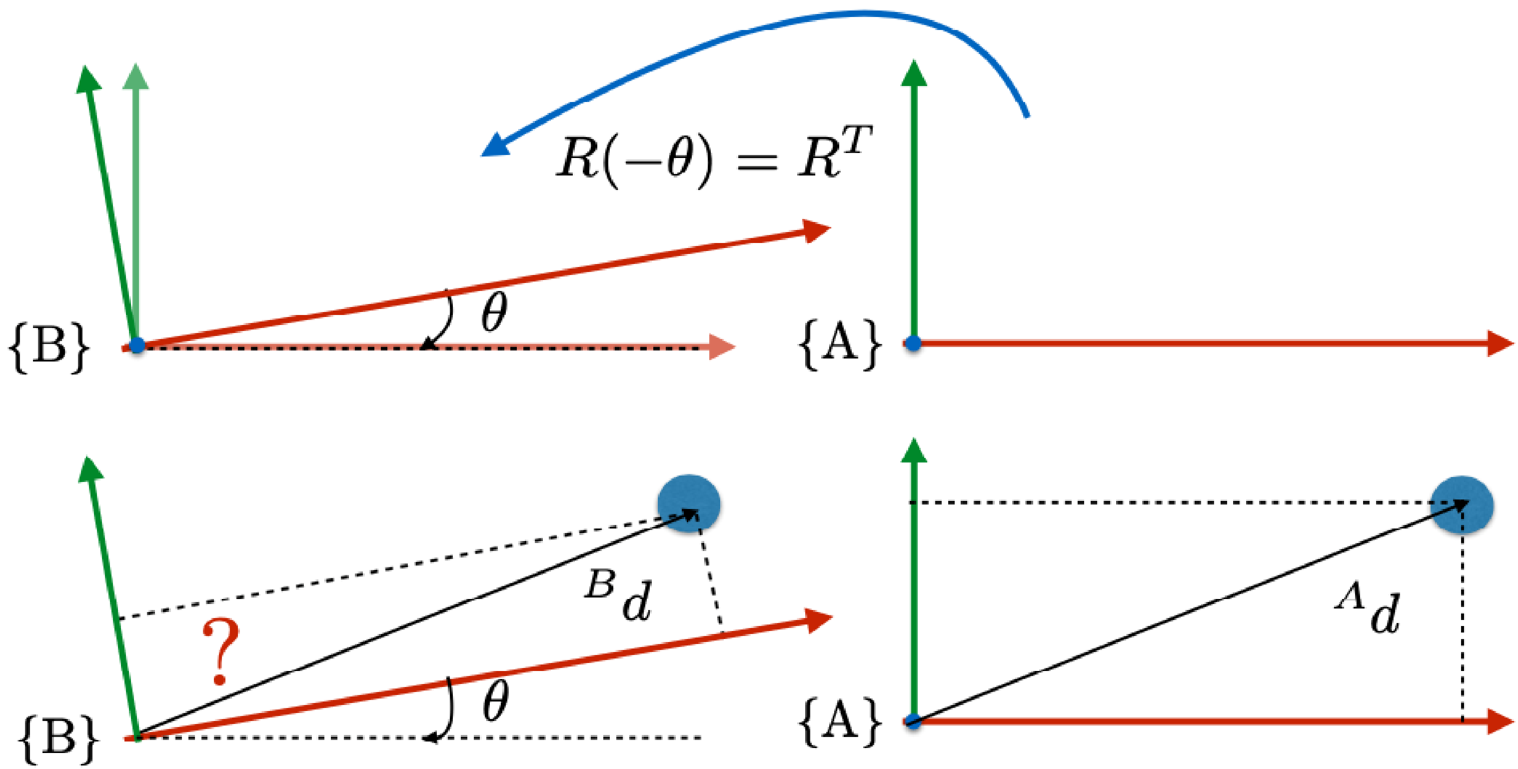
$[\ \vec{x} \quad \vec{y}\ ]$

Express Position in Desired Frame



$$R = R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \qquad {}^A d = R\,{}^B d$$

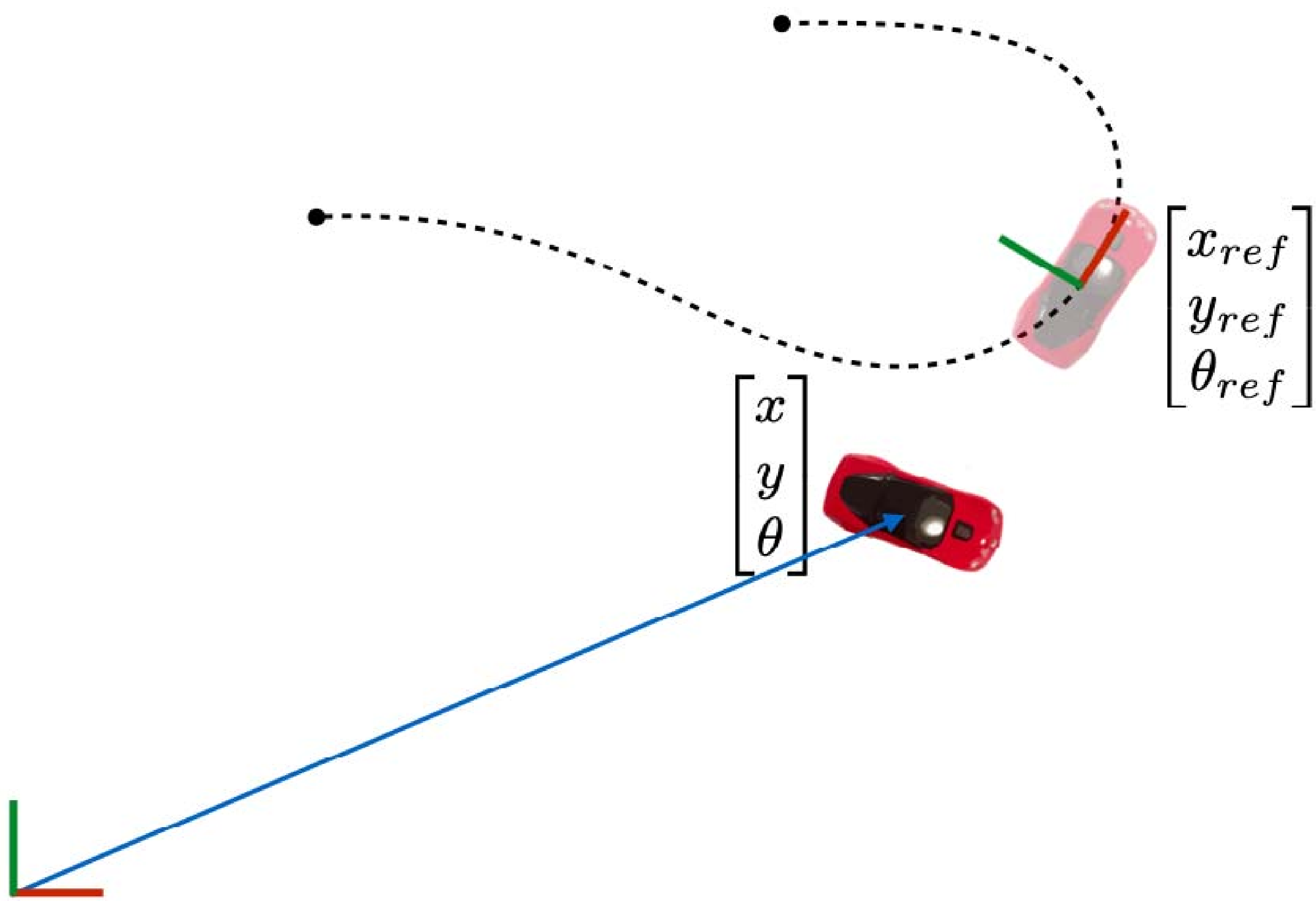
Inverse Transformation



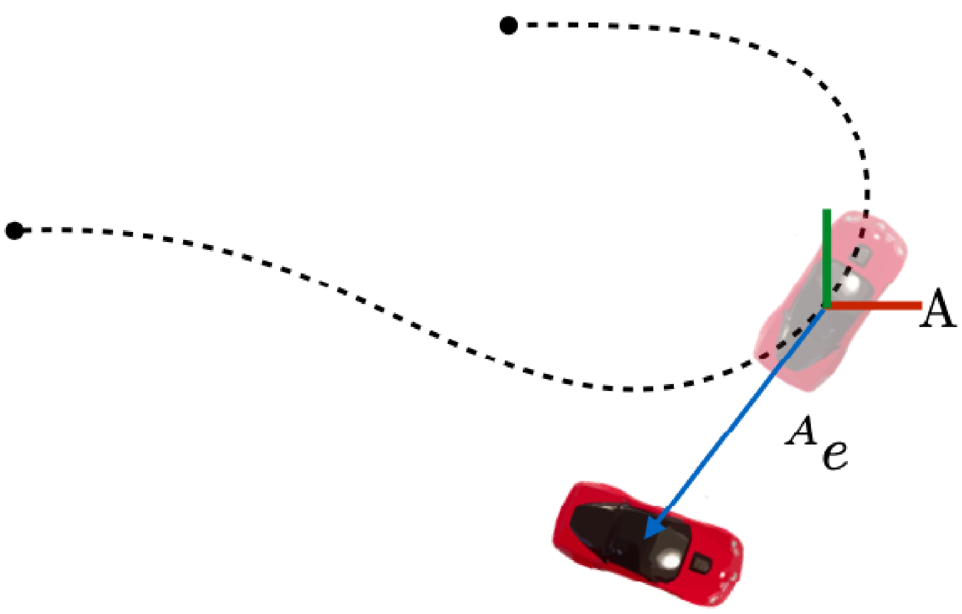
$$R^T = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

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

Step 3: Compute Error to This State



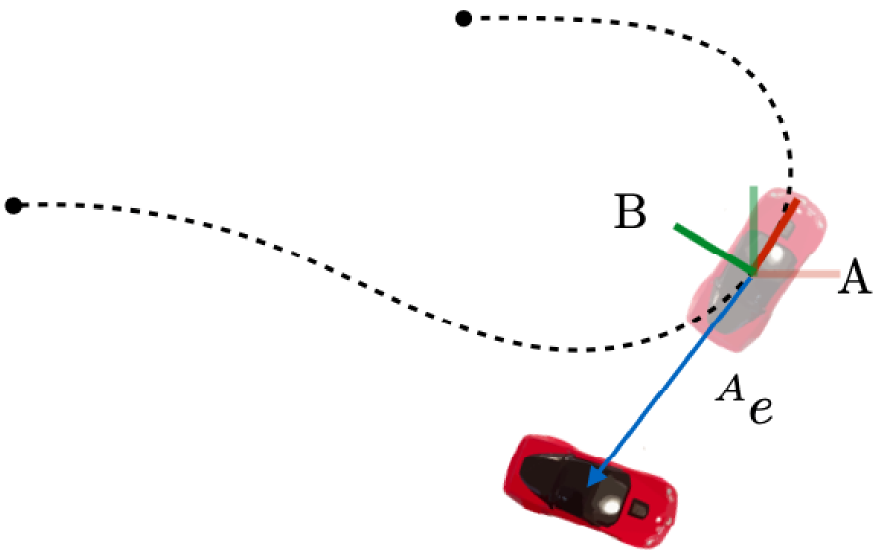
Step 3: Compute Error to This State



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

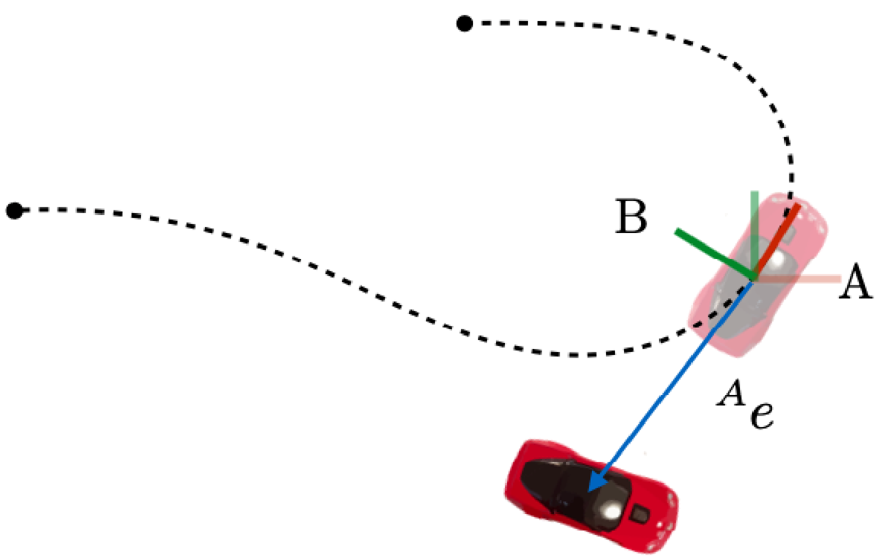


- We want position in frame B

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

(rotation of A w.r.t B) (rotation of A w.r.t B)

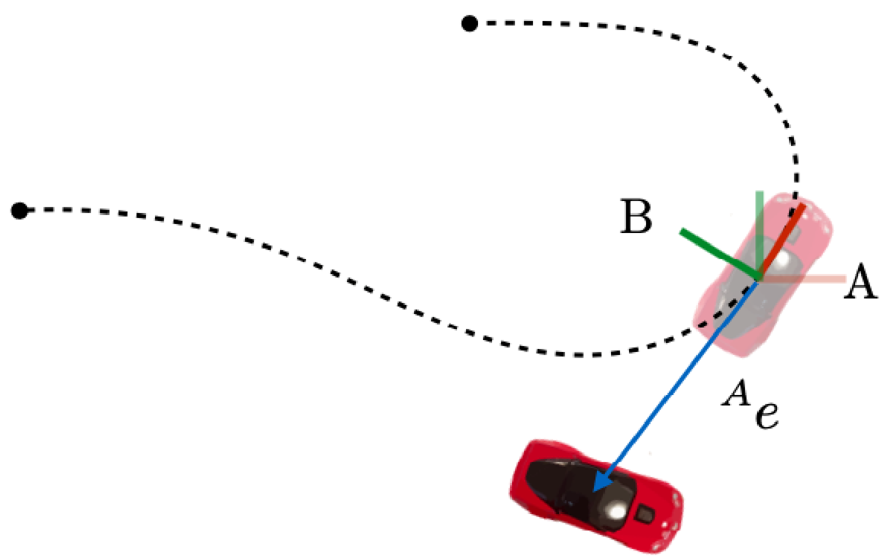
Step 3: Compute Error to This State



- We want position in frame B

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

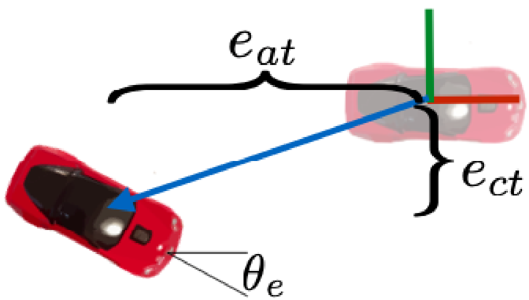
Step 3: Compute Error to This State



- Heading error

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

Step 3: Compute Error to This State

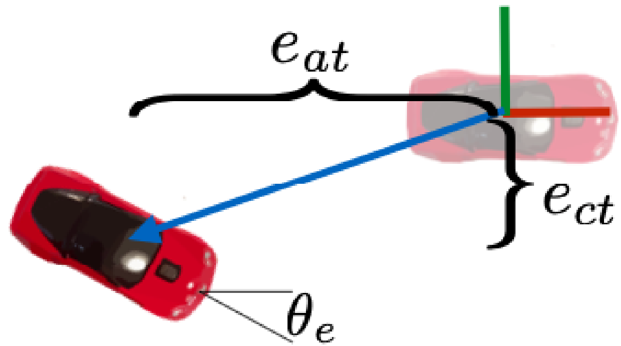


$$\text{(Along-track)} \quad e_{at} = \cos(\theta_{ref})(x - x_{ref}) + \sin(\theta_{ref})(y - y_{ref})$$

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

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

Some Things to Note



- We will only control the angular velocity; the linear velocity set to constant
- Hence, no real control on along-track error. Ignore for now
- Some control laws will only minimize cross-track error, others minimize both heading and cross-track error

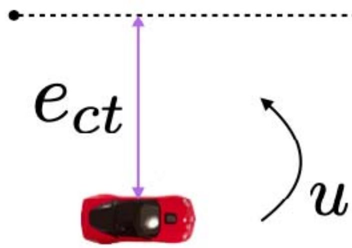
Step 4: Compute Control Law

- Compute the control signal based on instantaneous error

$$\underset{\text{control}}{u} = K(\underset{\text{error}}{e})$$

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

PID in Path Following Control



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

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

How to Evaluate the Derivative Term?

- Terrible way: Numerically differentiate error. Why is this a bad idea? $\dot{e} \approx \frac{e_{k+1} - e_k}{\delta t}$ sensitive to noise
- 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!

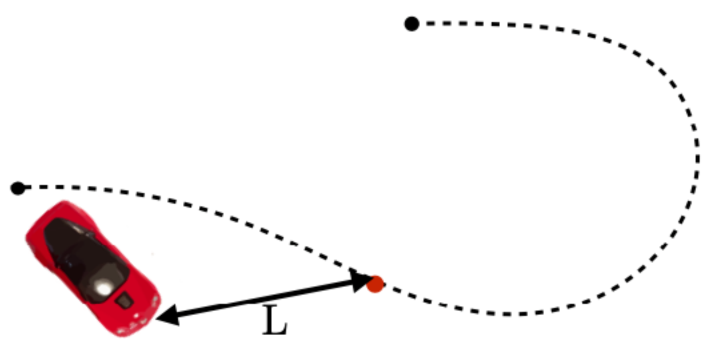
$$u = -\left(K_p e_{ct} + K_d V \sin\theta_e + K_i \int e_{ct}(t)dt\right)$$

Other Types of Controllers

- Pure-pursuit control
- Lyapunov control
- Linear Quadratic Regulator (LQR)
- Model Predictive Control (MPC)
- …

Pure-Pursuit Control

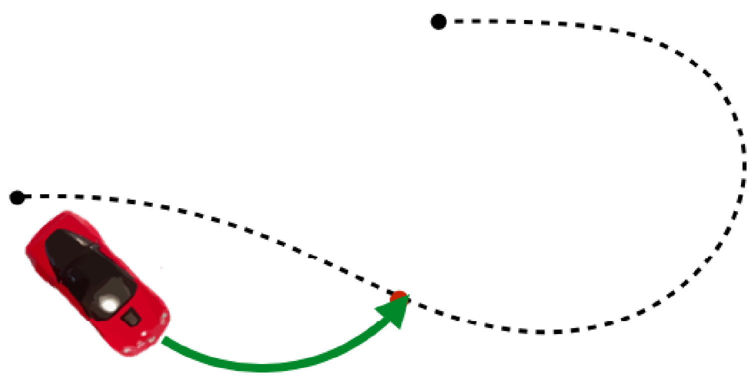
- Key idea: the robot is always moving in a circular arc
- Consider a reference at a lookahead distance



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

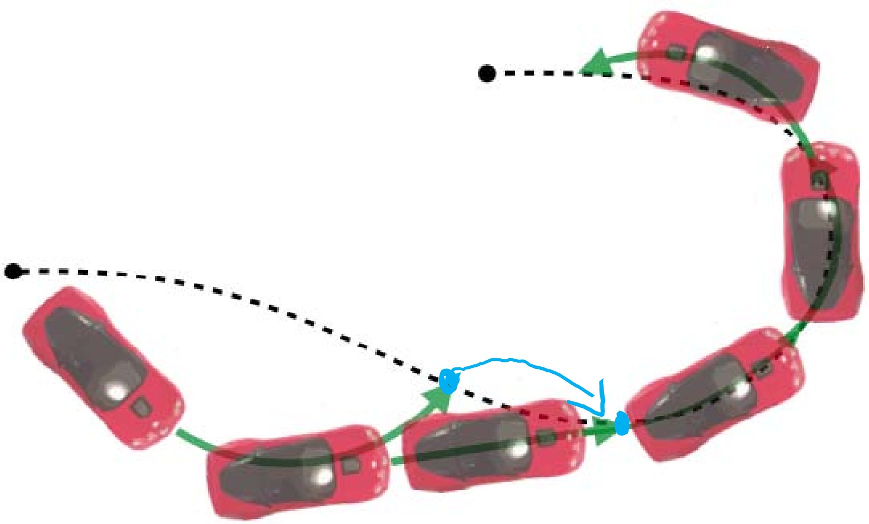
- Problem: Can we solve for a steering angle that guarantees that the car will pass through the reference?

Solution: Compute a Circular Arc



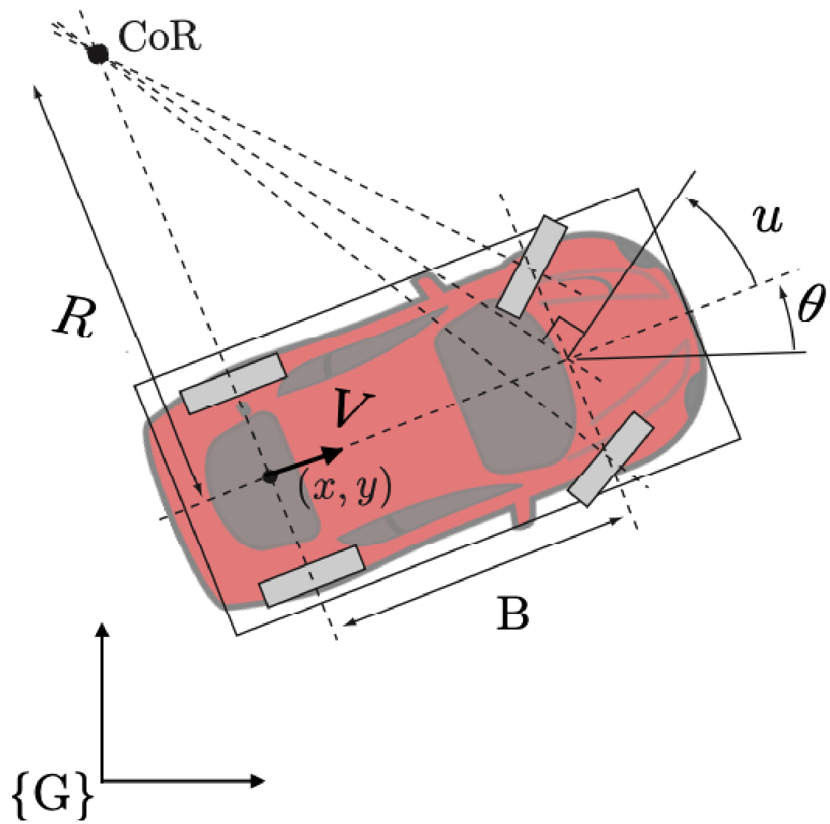
- We can always solve for an arc that passes through a lookahead point
- Note: as the robot moves forward, the point keeps moving

Pure Pursuit: Keep Chasing Lookahead



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

(Optional) Simple Car Kinematics



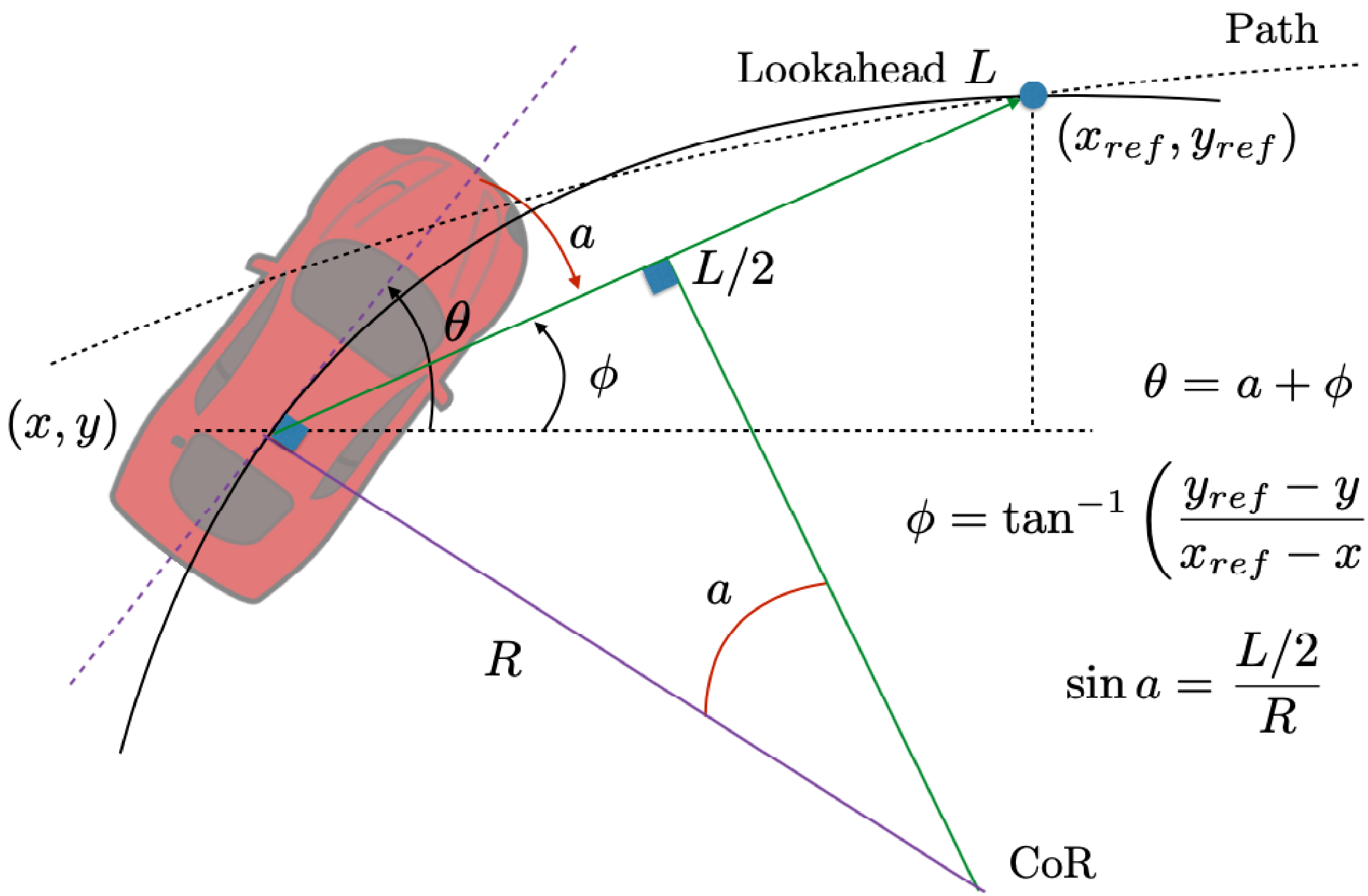
$$\tan u = \frac{B}{R}$$

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

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

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

(Optional) Chasing the Lookahead

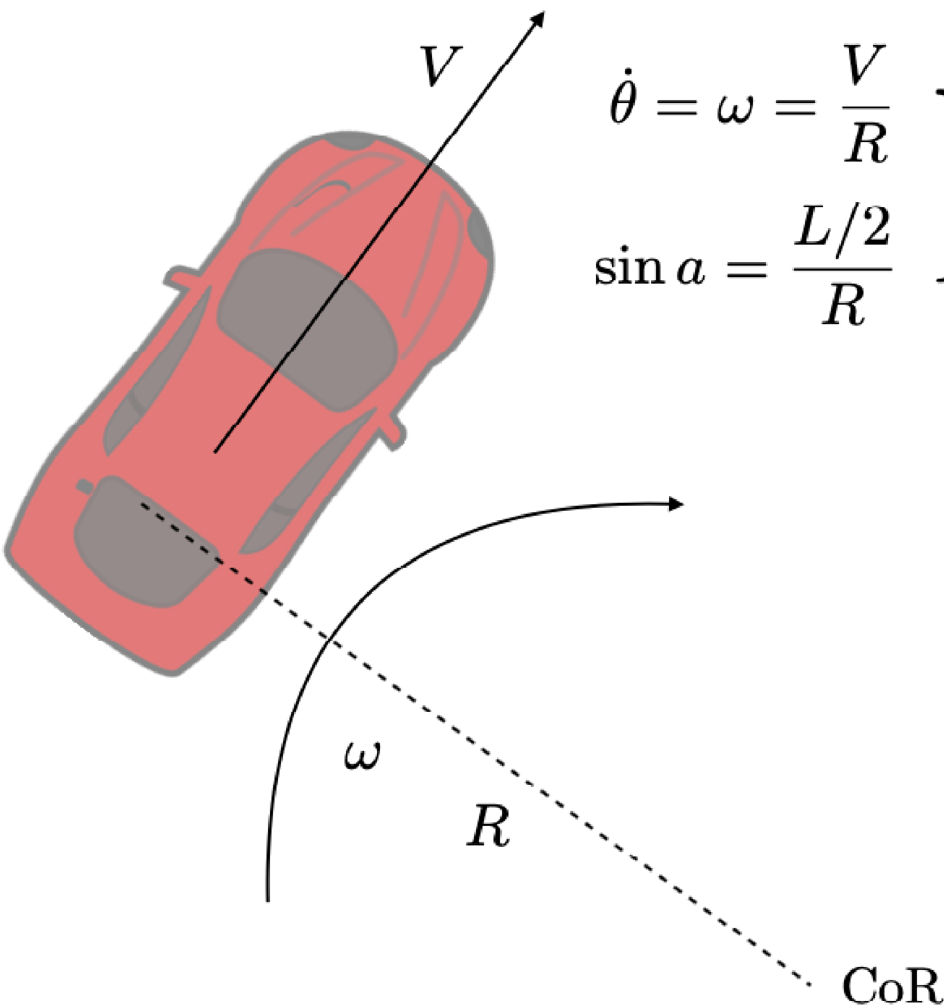


$$\theta = a + \phi$$

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

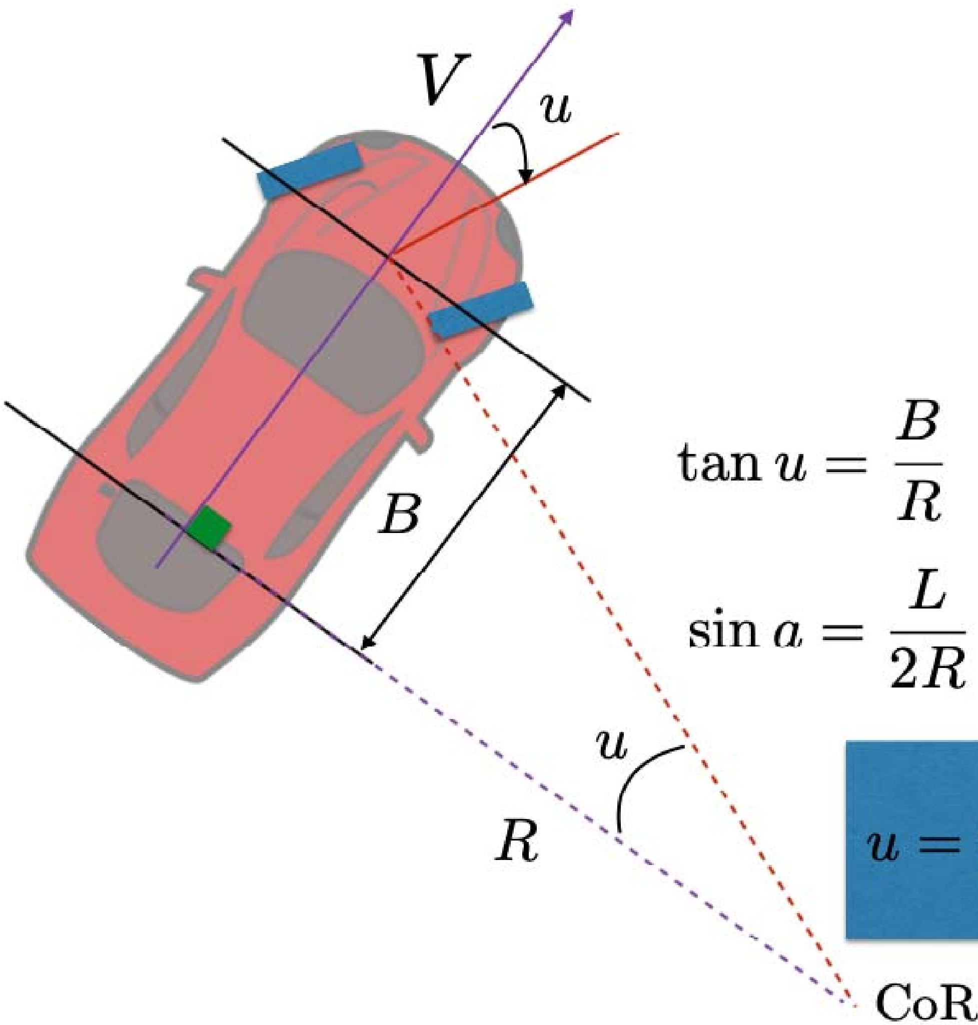
$$\sin a = \frac{L/2}{R}$$

(Optional) Rigid Body Rotation



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

(Optional) Rear-Axle Car Kinematics

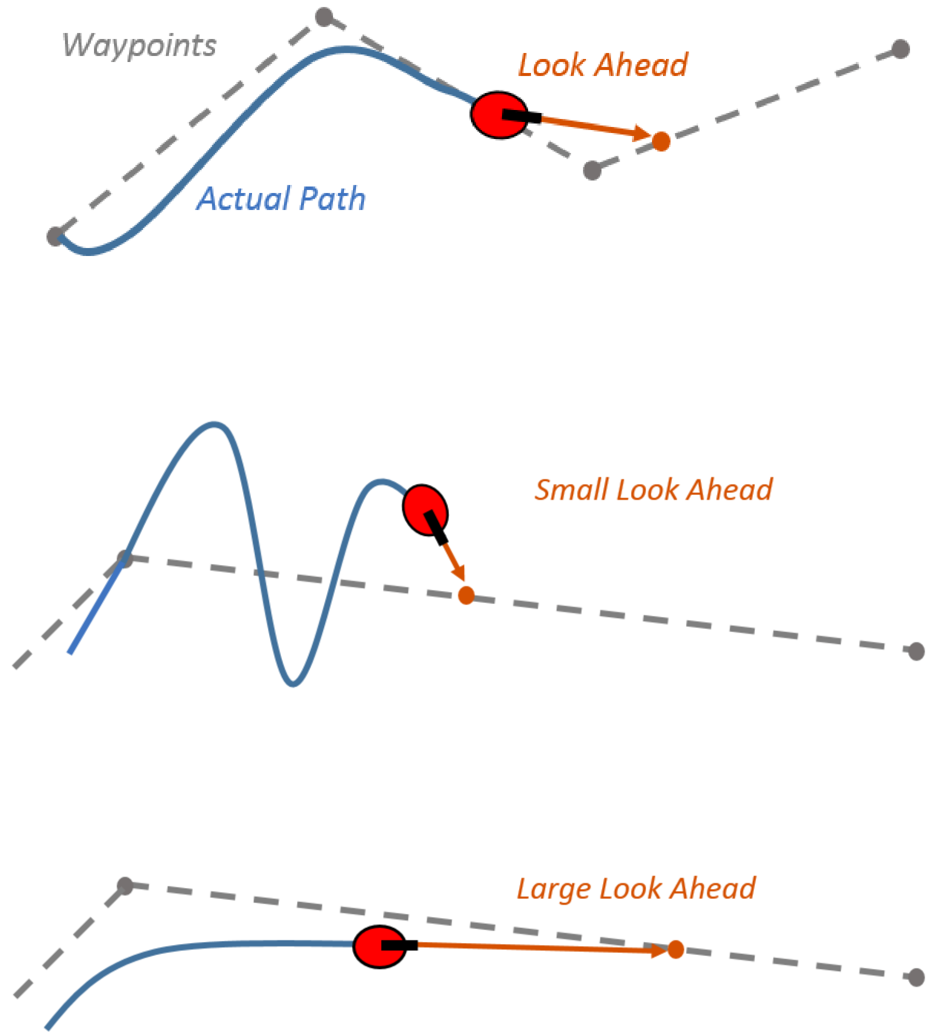


$$\left.\begin{array}{l} \tan u = \dfrac{B}{R} \\[2ex] \sin a = \dfrac{L}{2R} \end{array}\right\} \tan u = \frac{2B \sin a}{L}$$

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

Question: How do I Choose *L*?

- The lookahead distance *L* is the main tuning property for the controller
  - a small *L* moves the robot quickly towards the path, but may overshoots the path and oscillate along the path
  - a large *L* will reduce the oscillation but may result in larger curvatures near the corners



References

- https://www.mathworks.com/help/robotics/ug/pure-pursuit-controller.html
- https://www.ri.cmu.edu/pub_files/pub3/coulter_r_craig_1992_1/coulter_r_craig_1992_1.pdf
- https://www.ri.cmu.edu/pub_files/2009/2/Automatic_Steering_Methods_for_Autonomous_Automobile_Path_Tracking.pdf

- Thank You!