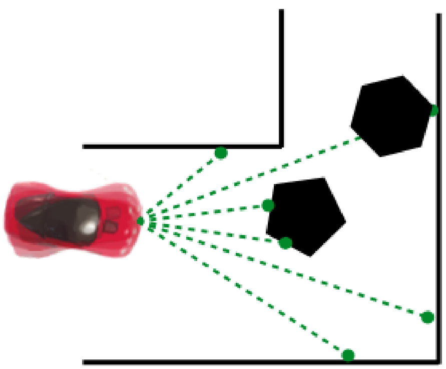
CMPE 185 Autonomous Mobile Robots

Navigation and Control Part 2

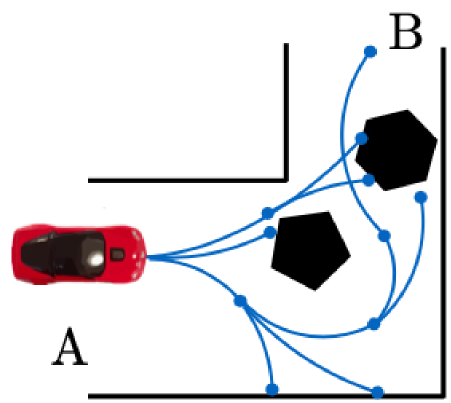
Dr. Wencen Wu
Computer Engineering Department
San Jose State University

Control for Mobile Robots

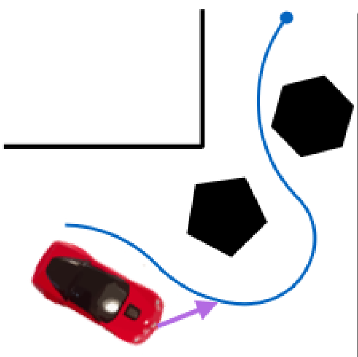
Estimate state



Plan a sequence of motions



Control robot to follow path

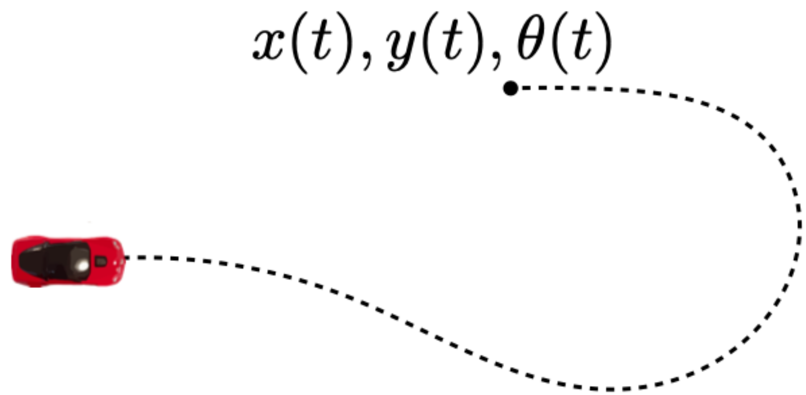


- Robot pose known
- Path is given

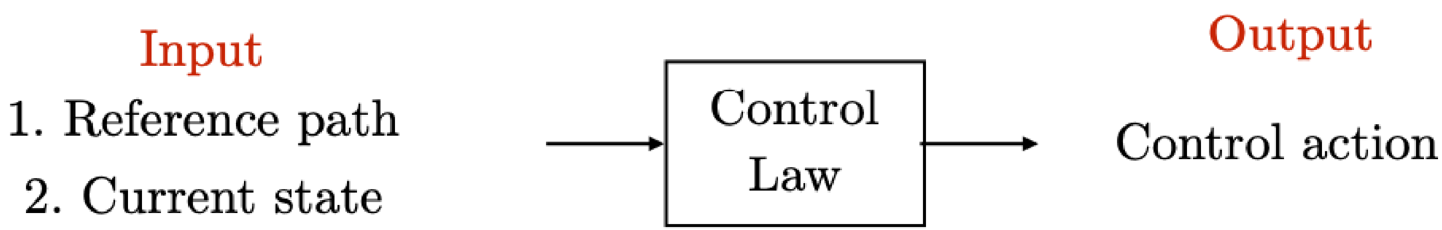
Some slides are adopted from Washington CS478

The Control Framework

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

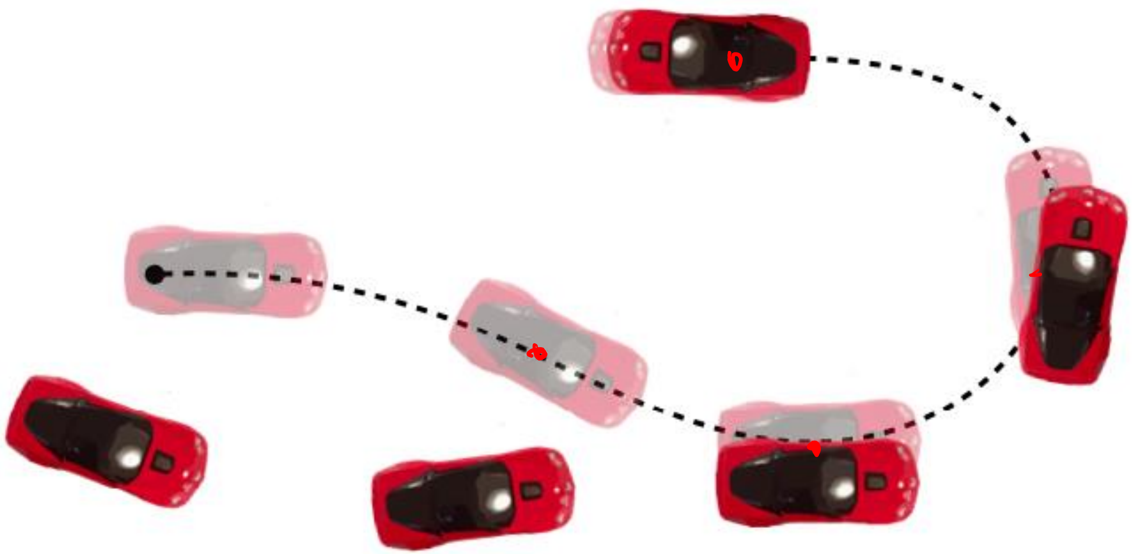


- Objective: Figure out a control trajectory u(t) to achieve this



- 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

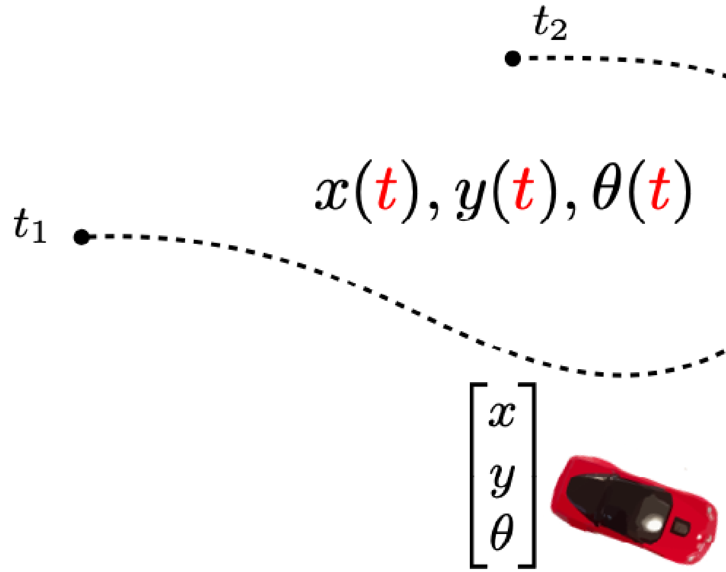
Steps to Designing a Controller

- 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*
  - A time-parameterized trajectory is a trajectory where the position and orientation of the robot are expressed as functions of time
  - It is used for controlling the motion of a robot w.r.t. time, ensuring the robot follows a specific path at a specific speed

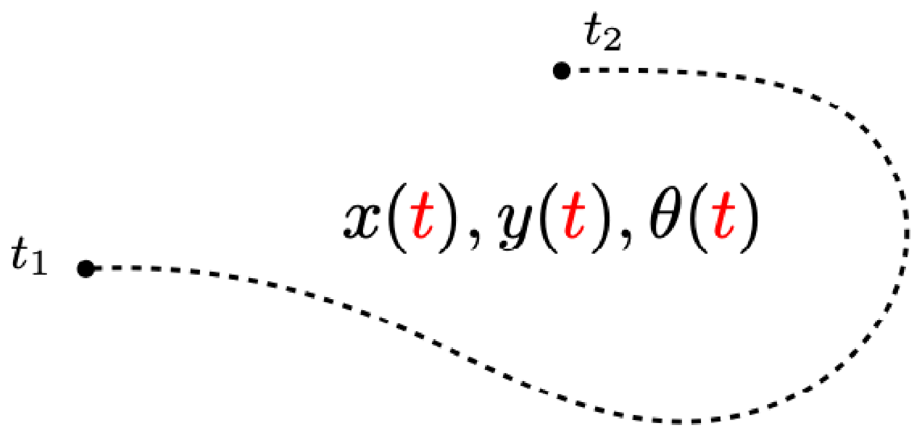


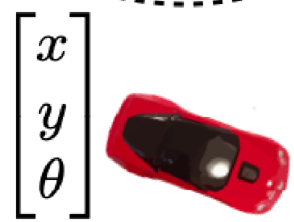
$$\boldsymbol{q}(t) = (x(t), y(t), \theta(t))$$

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



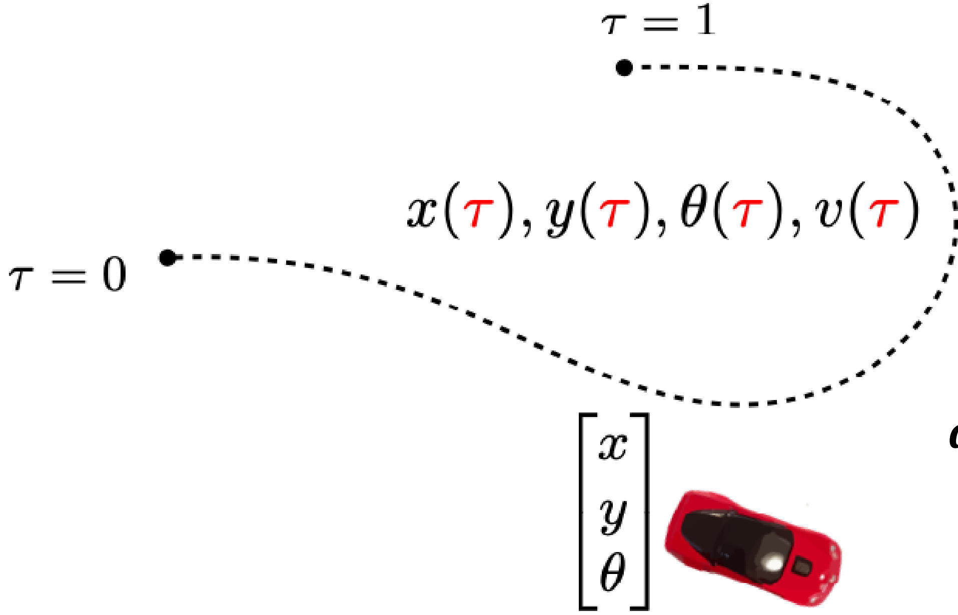


http://www.osrobotics.org/osr/planning/time_parameterization.html

Step 1: Get a Reference Path

How do we define a reference?

- **Option 2:** Index-parameterized *path*
  - An index-parameterized trajectory is a trajectory where the position and orientation of the robot are expressed as a functions of an abstract parameter (often called an index), rather than time
  - The index indicates a position along the path, but does not specify the time

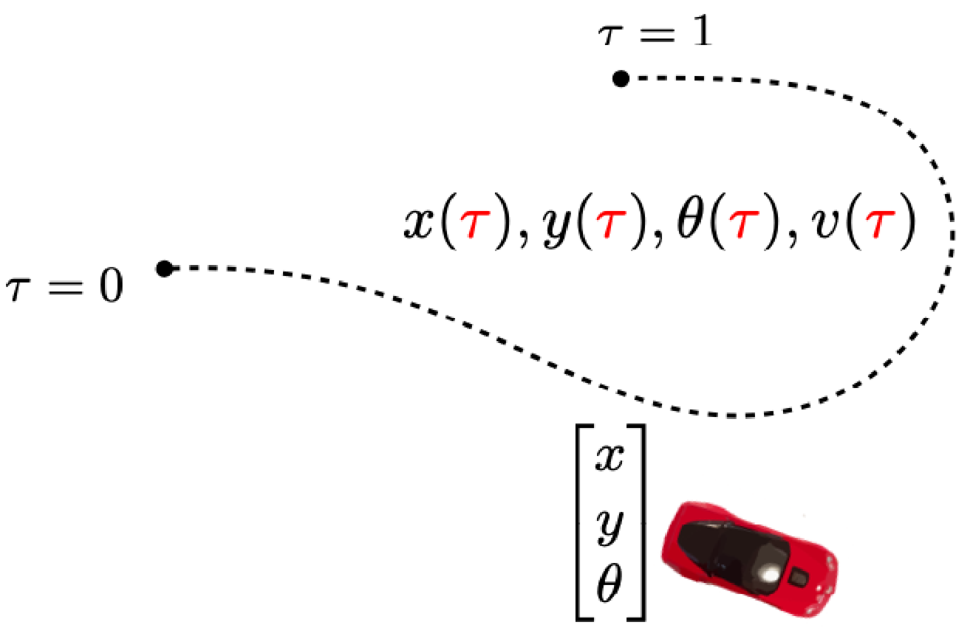


$$\boldsymbol{q}(t) = (x(\tau), y(\tau), \theta(\tau))$$

Step 1: Get a Reference Path

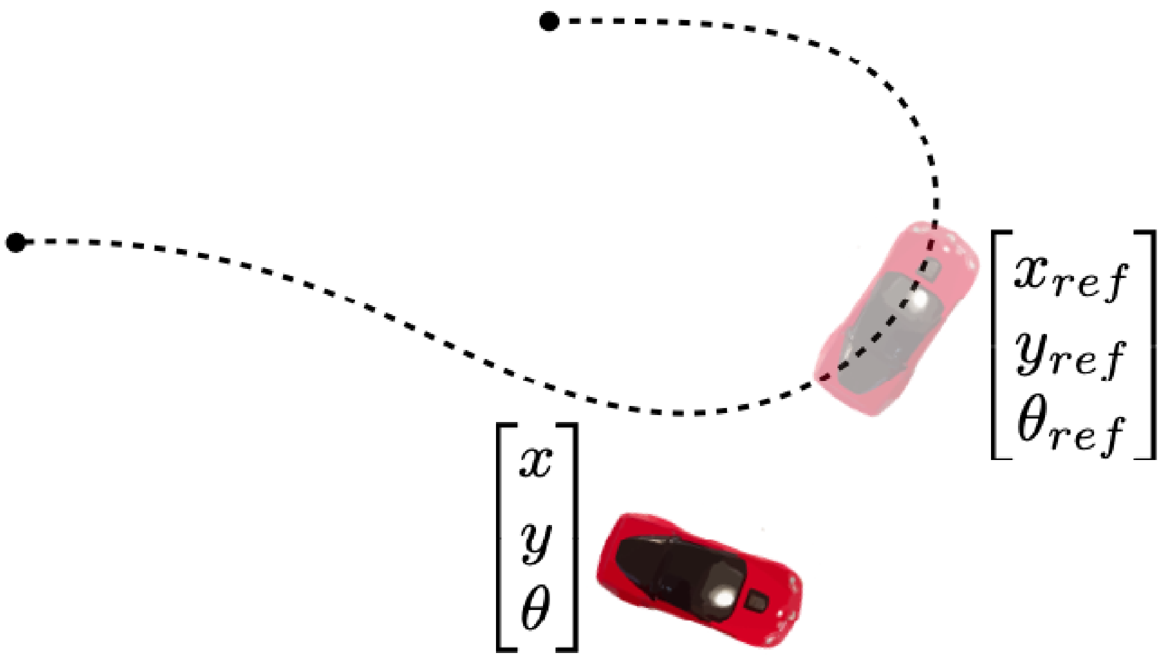
How do we define a reference?

- **Option 2:** Index-parameterized *path*
  - Pro: Useful for conveying the shape you want the robot to follow, focusing on spatial optimization first
  - Con: Can't control **when** robot will reach a point



Step 2: Pick a Reference (desired) State

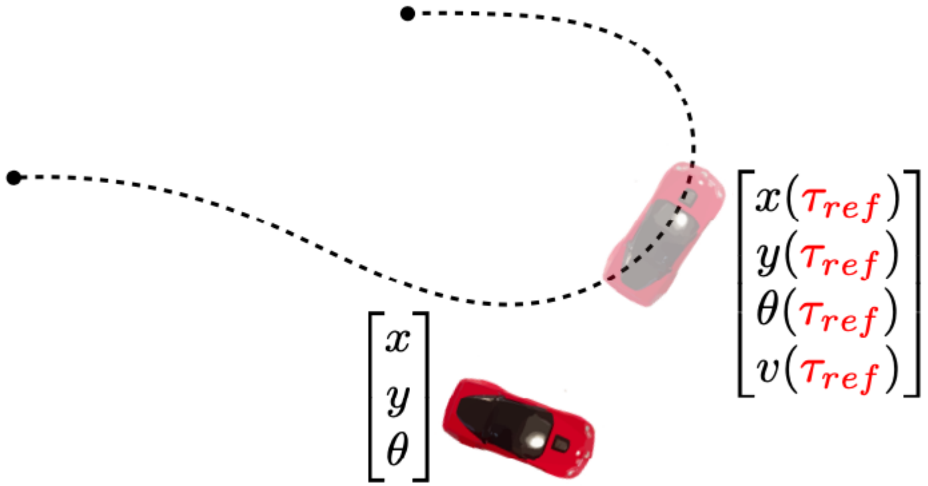
How do we pick a reference?



Step 2: Pick a Reference (desired) State

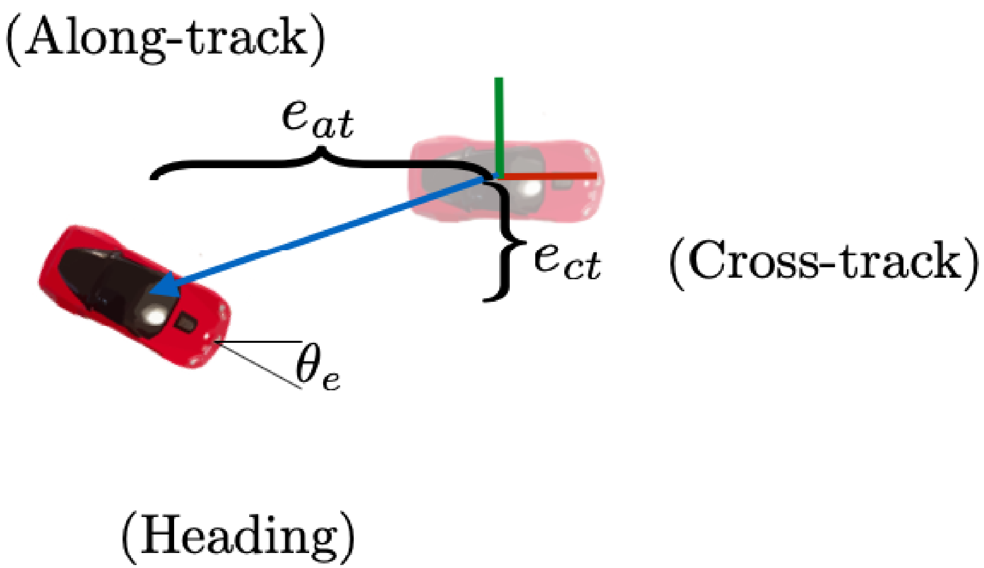
How do we pick a reference?

- Example: 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

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

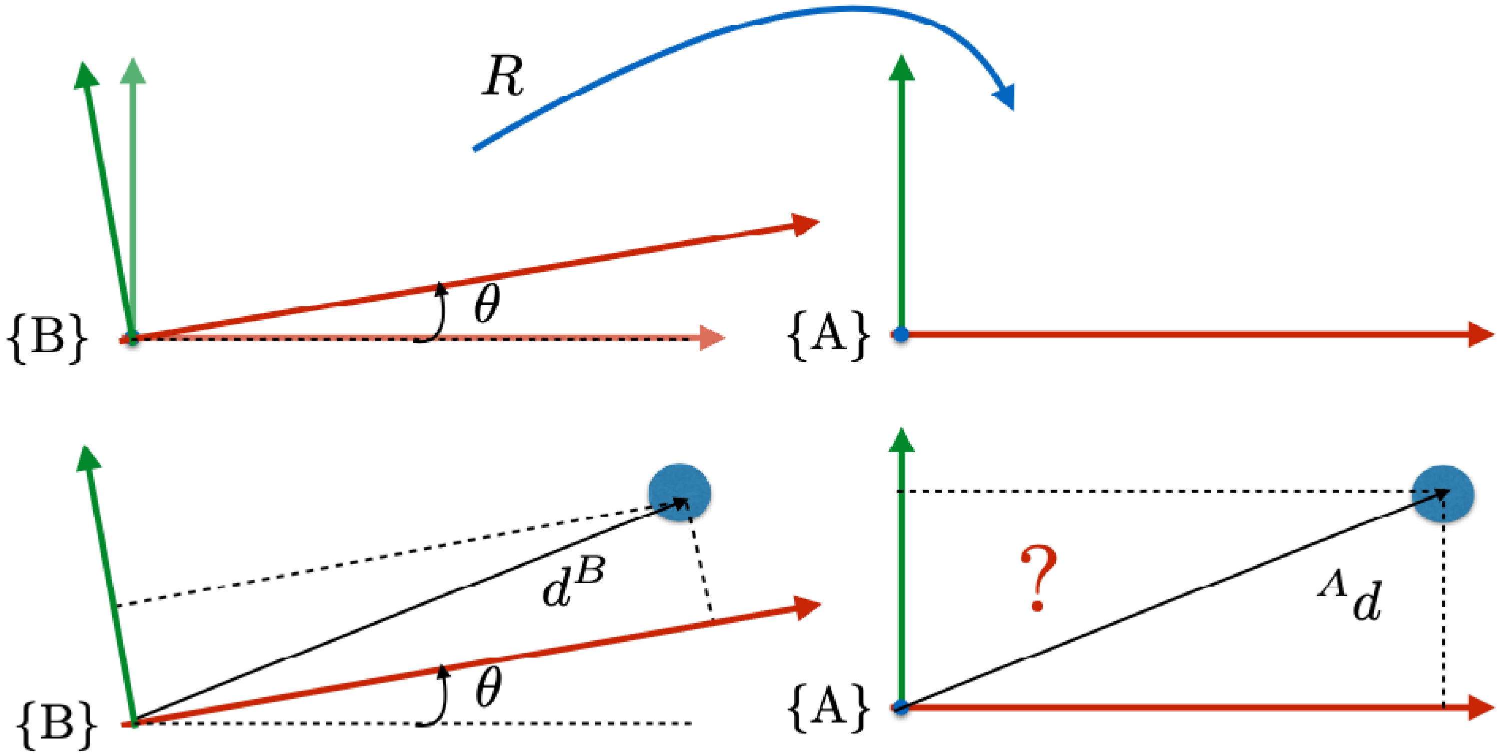
Recall: Rotation Matrix

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

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

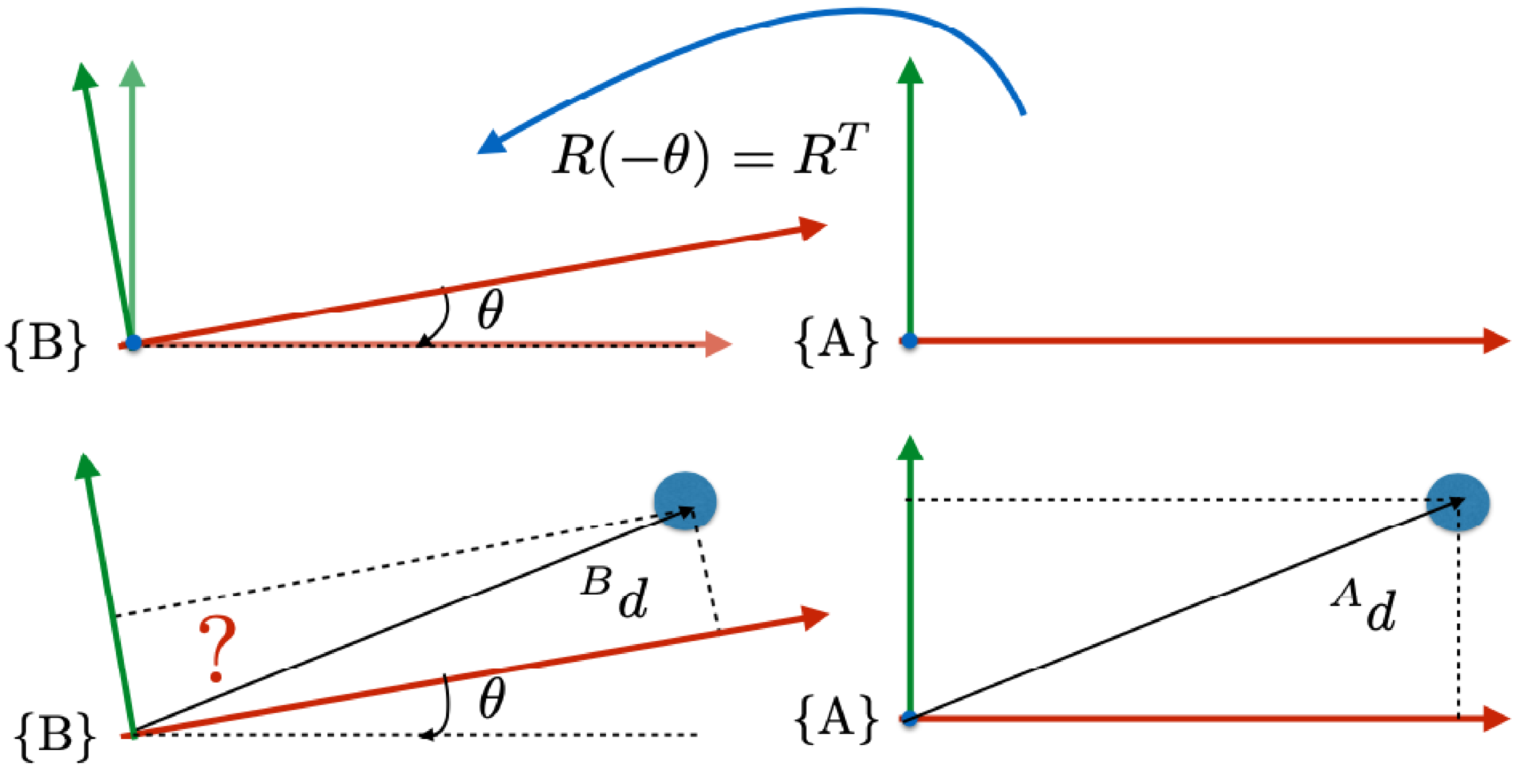
$$R = R_z(\theta) = \begin{bmatrix} \boxed{\begin{matrix} \cos\theta \\ \sin\theta \end{matrix}} & \boxed{\begin{matrix} -\sin\theta \\ \cos\theta \end{matrix}} \end{bmatrix}$$

Express Position in Desired Frame



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

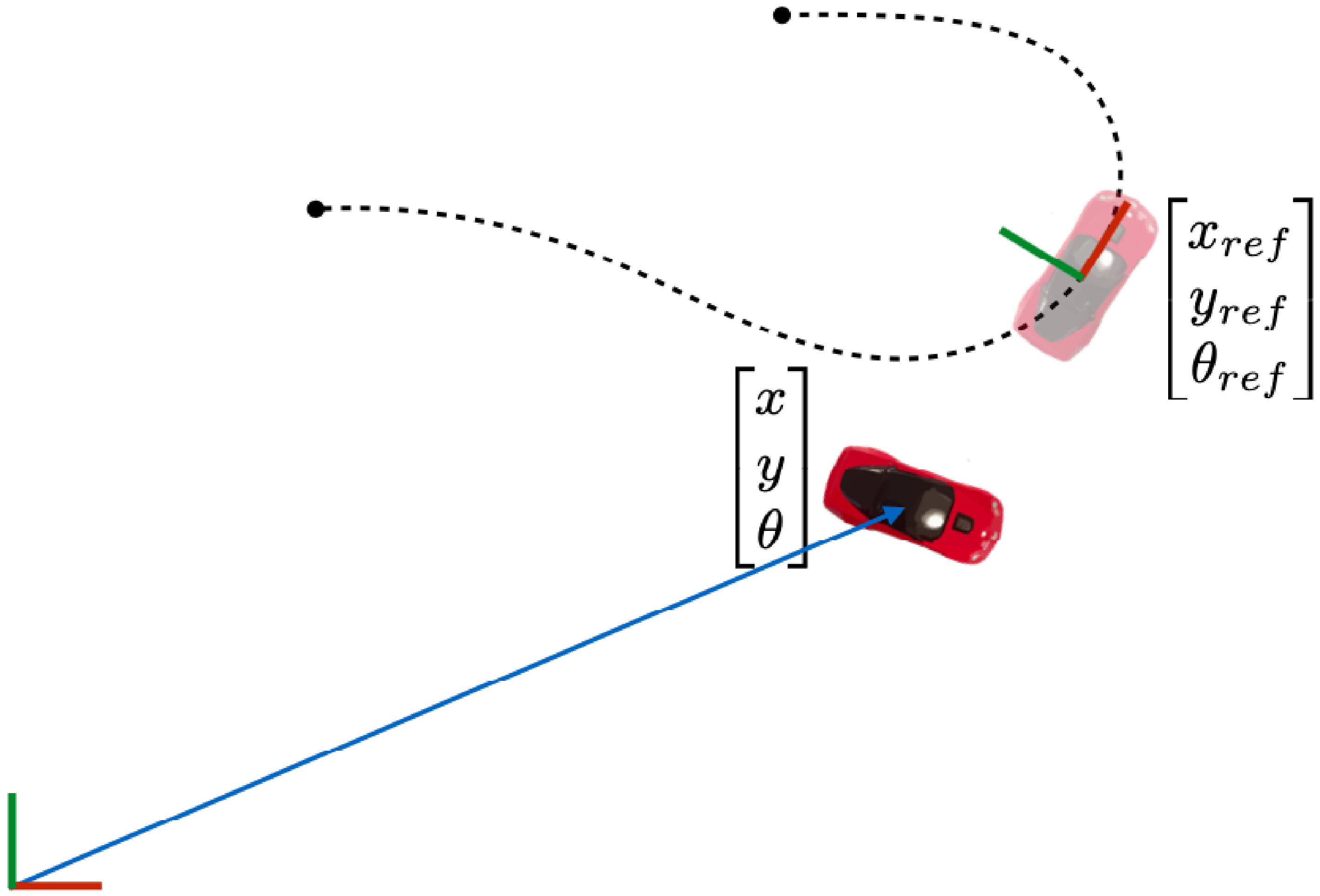
Inverse Transformation



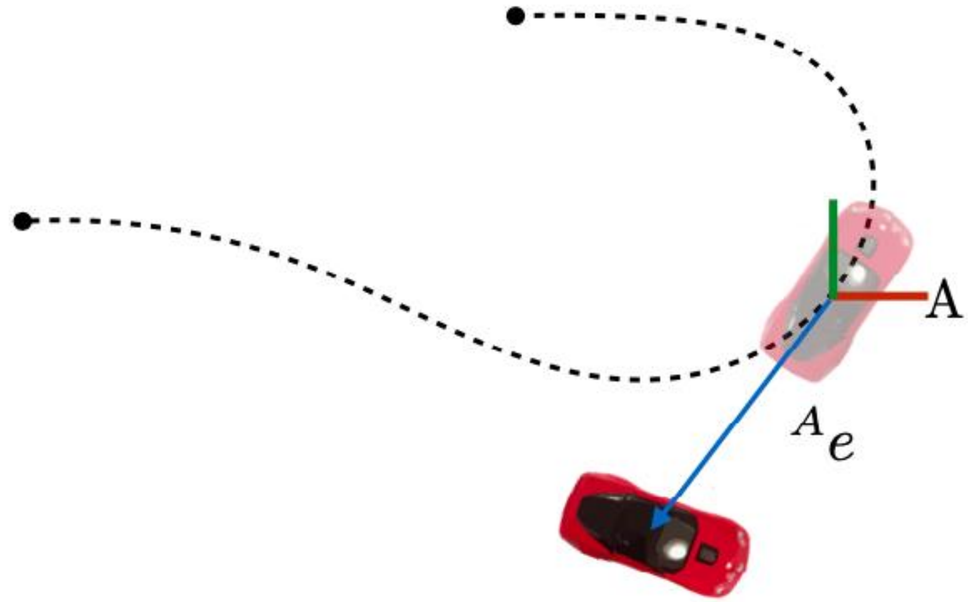
$$R^T = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

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

Step 3: Compute Error to This State



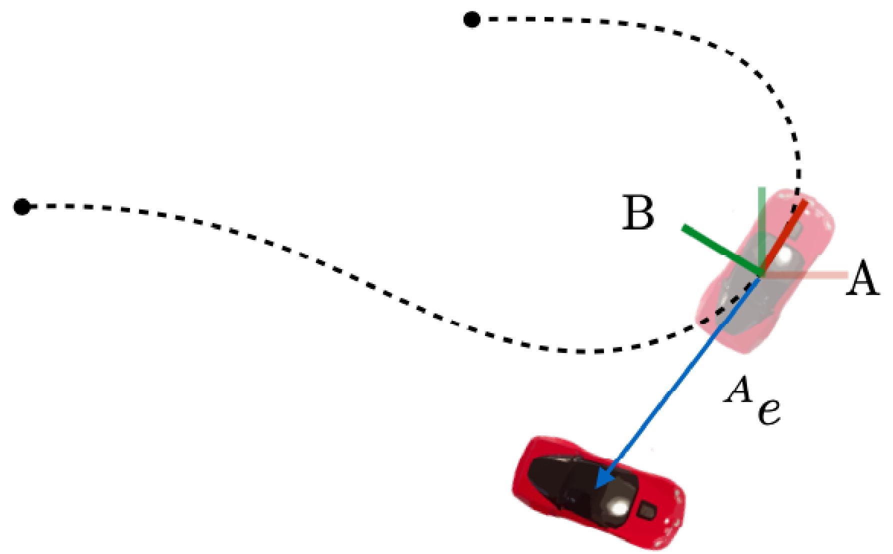
Step 3: Compute Error to This State



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

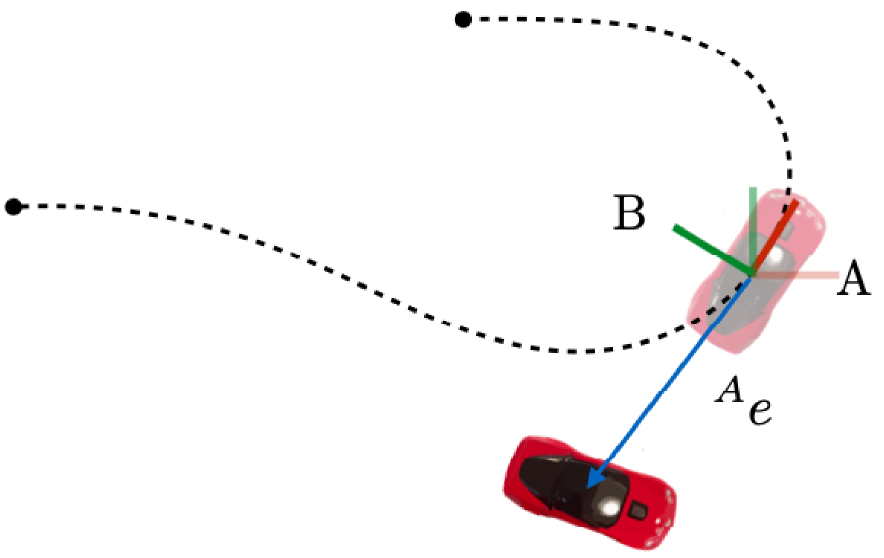


- We want position in frame B

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

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

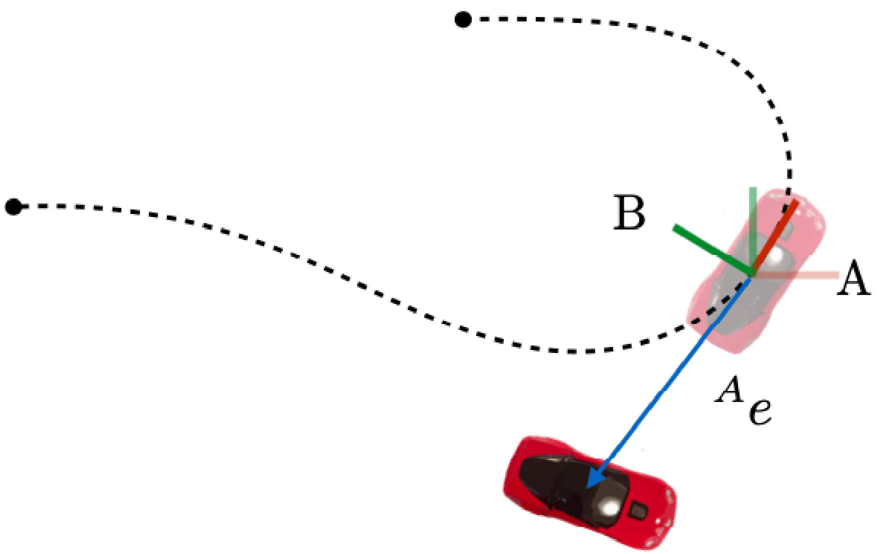
Step 3: Compute Error to This State



- We want position in frame B

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

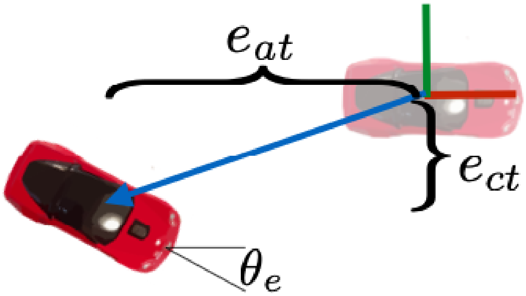
Step 3: Compute Error to This State



- Heading error

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

Step 3: Compute Error to This State

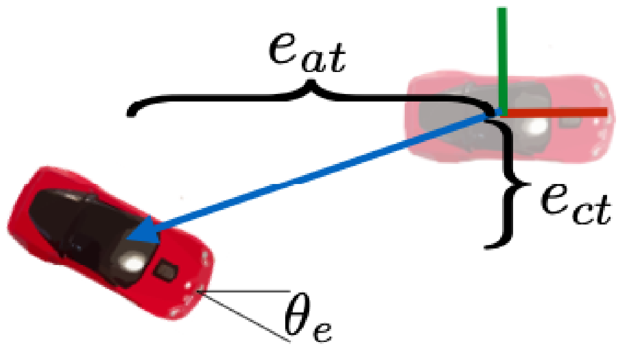


$$\text{(Along-track)} \quad e_{at} = \cos(\theta_{ref})(x - x_{ref}) + \sin(\theta_{ref})(y - y_{ref})$$

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

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

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

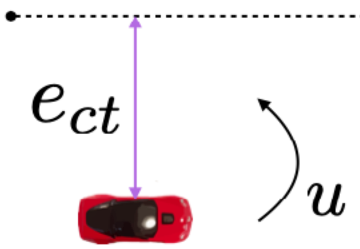
Step 4: Compute Control Law

- 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



$$u = -\left(\underset{\substack{\text{Proportional} \\ \text{(current)}}}{K_p e_{ct}} + \underset{\substack{\text{Integral} \\ \text{(past)}}}{K_i \int e_{ct}(t)dt} + \underset{\substack{\text{Derivative} \\ \text{(future)}}}{K_d \dot{e}_{ct}} \right)$$

How to 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!

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

Other Types of Controllers

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

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!