RRT MP

ECE/CS498

Pulkit Katdare

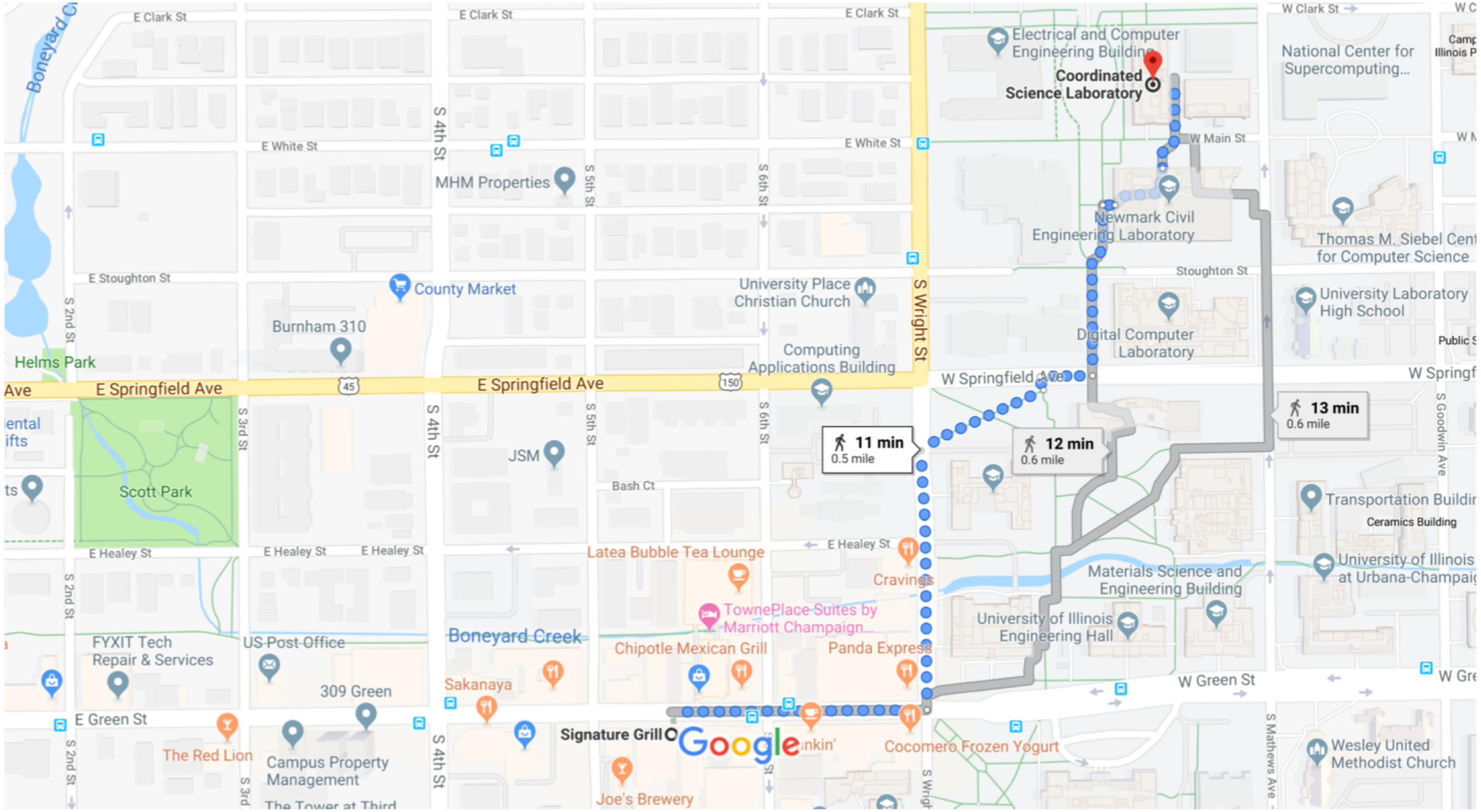
8th April, 2019

Outline

- Introduction
- Rapidly-Exploring Random Tree
- Controller Design
- Putting it all together

What is Planning?

Google Maps Signature Grill to Coordinated Science Laboratory Walk 0.5 mile, 11 min



Map data ©2019 Google 200 ft

Motion Planning

Given initial state x_{init} and a goal X_G, what is the path or sequence of control inputs that will lead us from start to goal?

Possible Issues:

- Obstacle avoidance
- Nonholonomic systems
- Computationally intensive
  - Nonconvex optimization
  - Large number of samples required in real-time

Two approaches: optimization-based & sampling-based techniques

Rapidly exploring Random Tree (RRT)

Build tree by generating next states through the dynamics by randomly selecting inputs

Rapidly exploring Random Tree (RRT)

Build tree by generating next states through the dynamics by randomly selecting inputs

Generate_RRT($x_{init}, K, \Delta t$)

$\mathcal{T}$.init(x_{init})

for k = 1 to K

x_{rand} ← RANDOM_STATE()

x_{near} ← NEAREST_NEIGHBOR($x_{rand}, \mathcal{T}$)

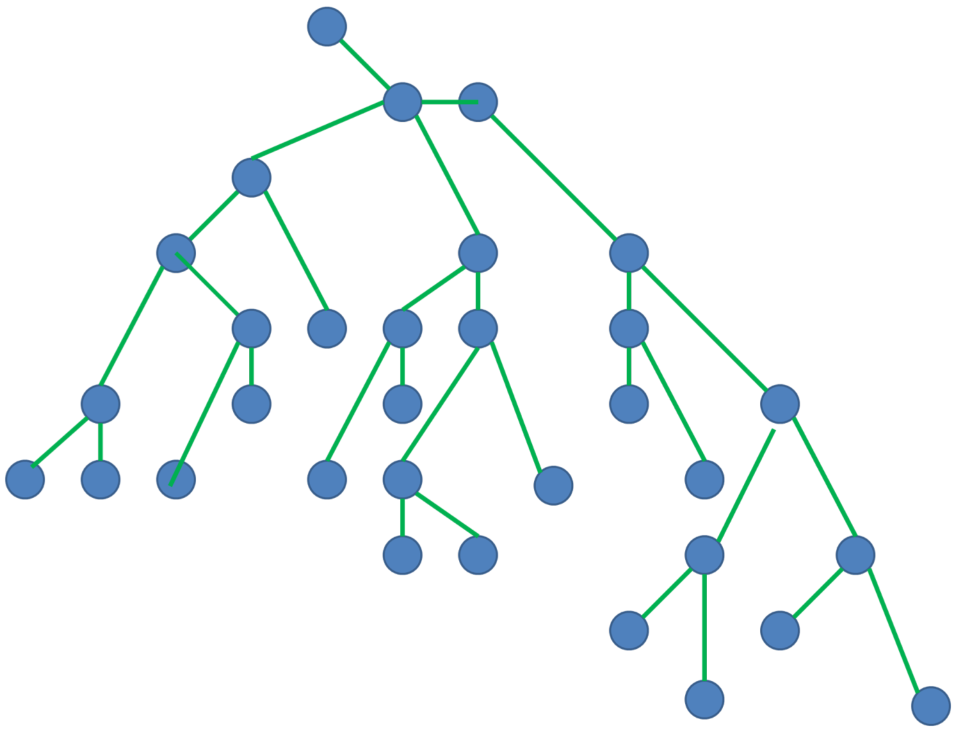
u ← SELECT_INPUT(x_{rand}, x_{near})

x_{new} ← NEW_STATE($x_{near}, u, \Delta t$)

$\mathcal{T}$.add_vertex(x_{new})

$\mathcal{T}$.add_edge(x_{near}, x_{new}, u)

Return $\mathcal{T}$



Rapidly exploring Random Tree (RRT)

Build tree by generating next states through the dynamics by randomly selecting inputs

Generate_RRT($x_{init}, K, \Delta t$)

$\mathcal{T}$.init(x_{init})

for k = 1 to K

  x_{rand} ← RANDOM_STATE()

  x_{near} ← NEAREST_NEIGHBOR($x_{rand}, \mathcal{T}$)

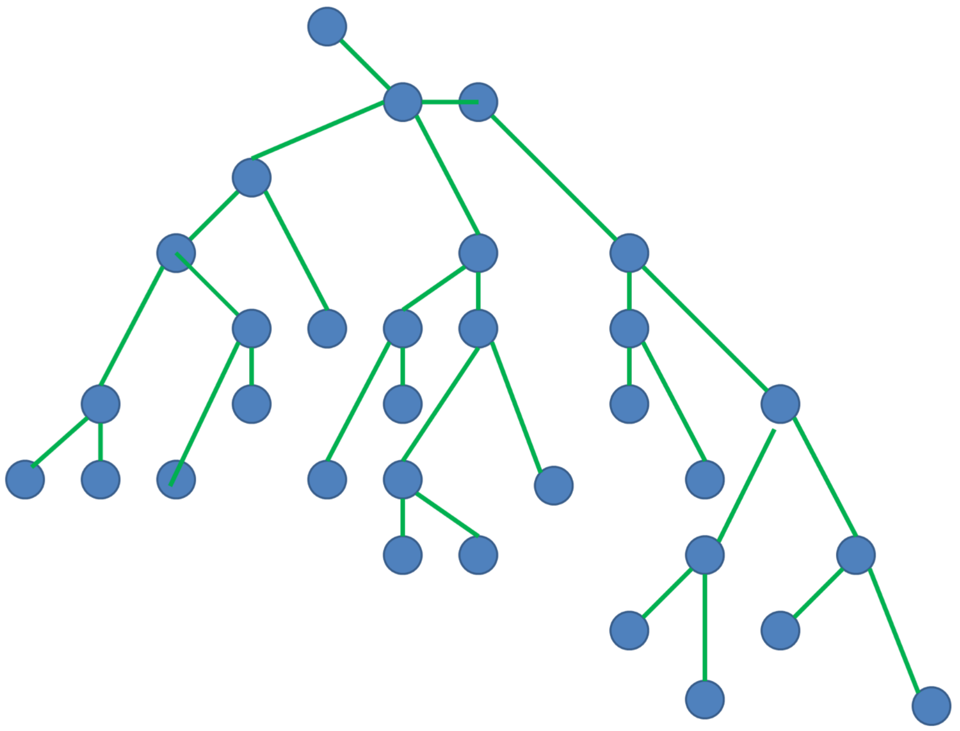
  u ← SELECT_INPUT(x_{rand}, x_{near}) How???

  x_{new} ← NEW_STATE($x_{near}, u, \Delta t$)

  $\mathcal{T}$.add_vertex(x_{new})

  $\mathcal{T}$.add_edge(x_{near}, x_{new}, u)

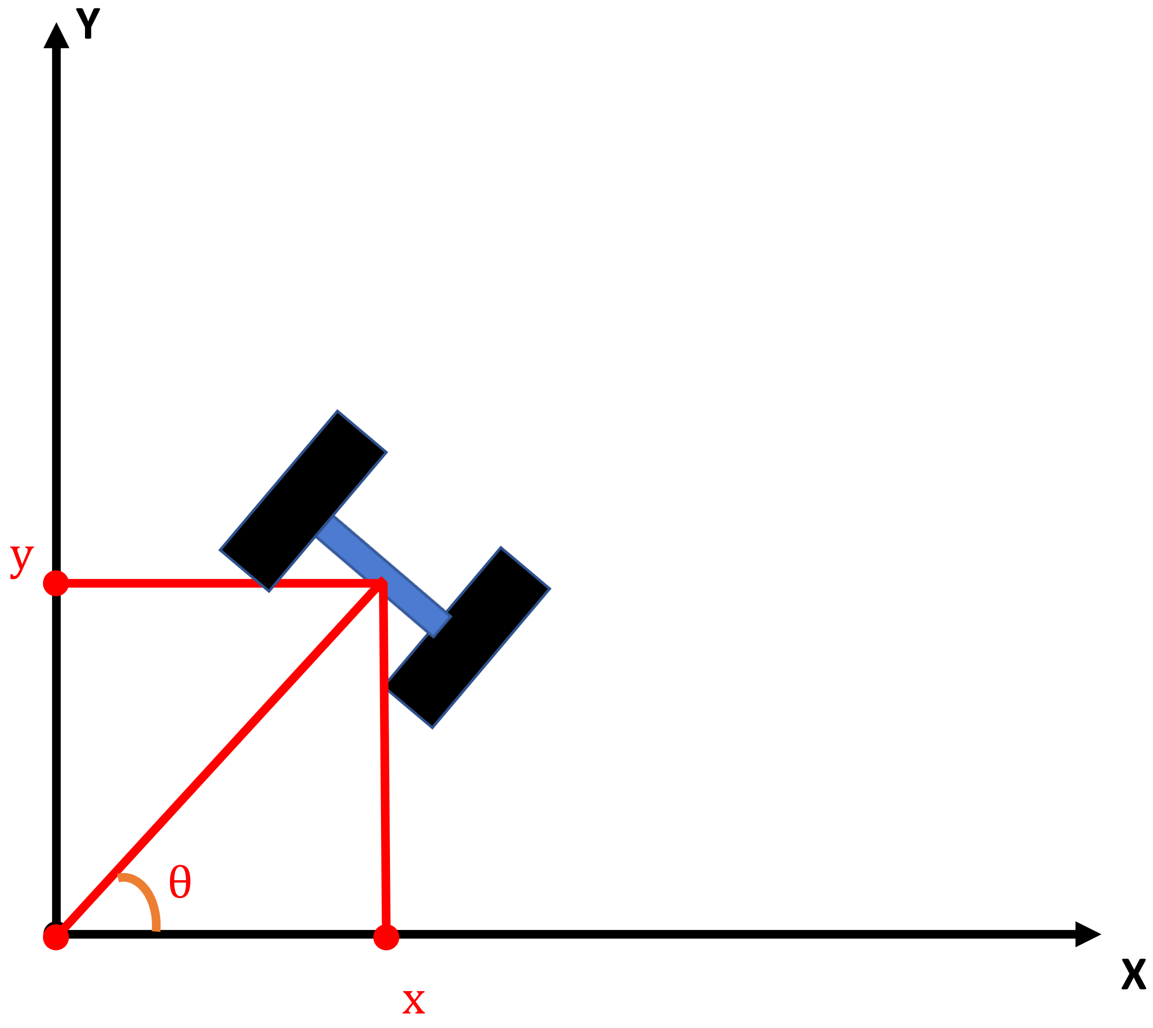
Return $\mathcal{T}$



Controller Design?

- How to move a robot from point **A** to point **B**
- We need a model first

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} V_R \cos(\theta) \\ V_R \sin(\theta) \\ \delta \end{bmatrix}$$

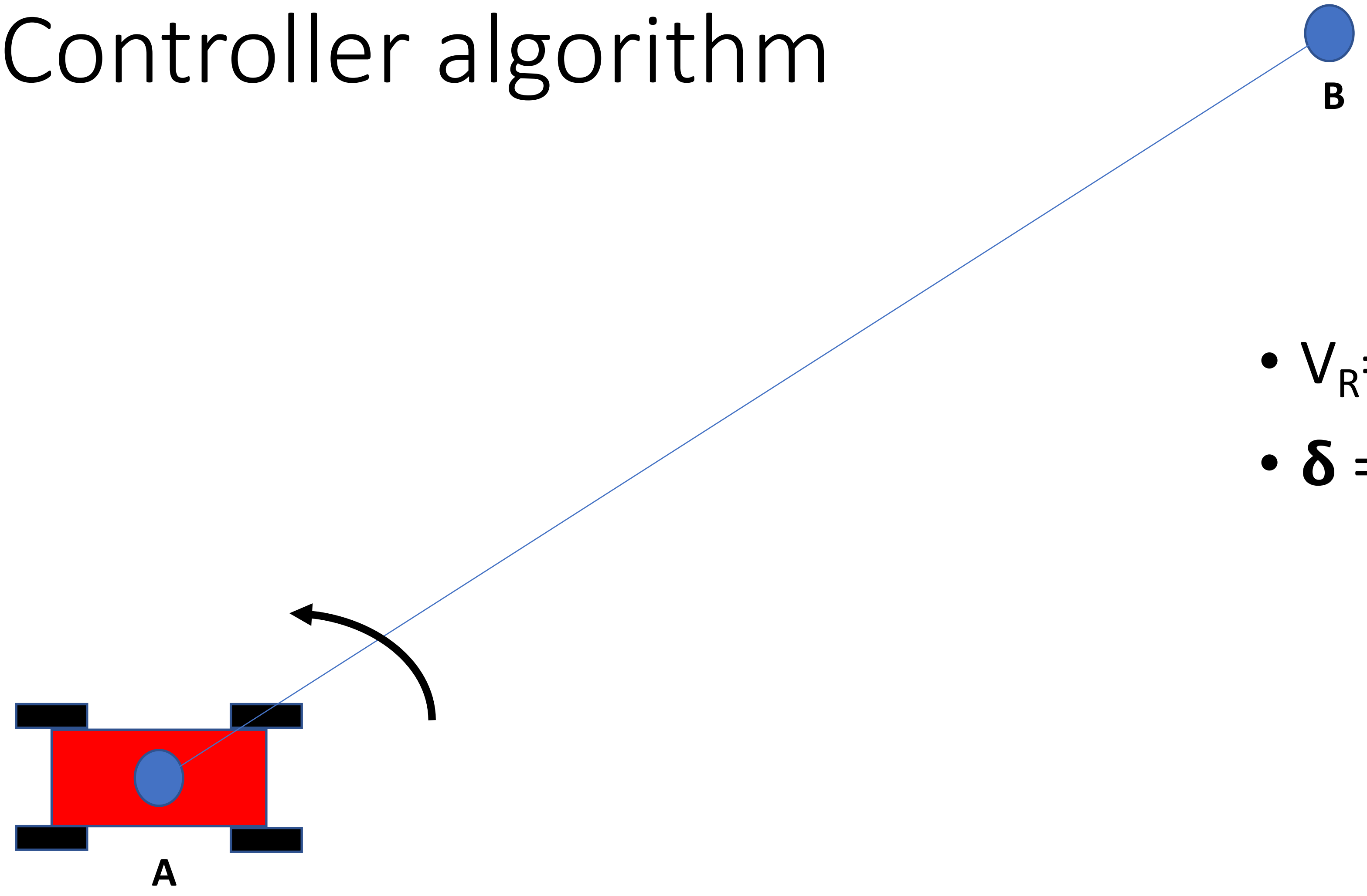


$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} V_R \cos(\theta) \\ V_R \sin(\theta) \\ \delta \end{bmatrix}$$

- V_R : speed of the car
- **δ** : steering of the car

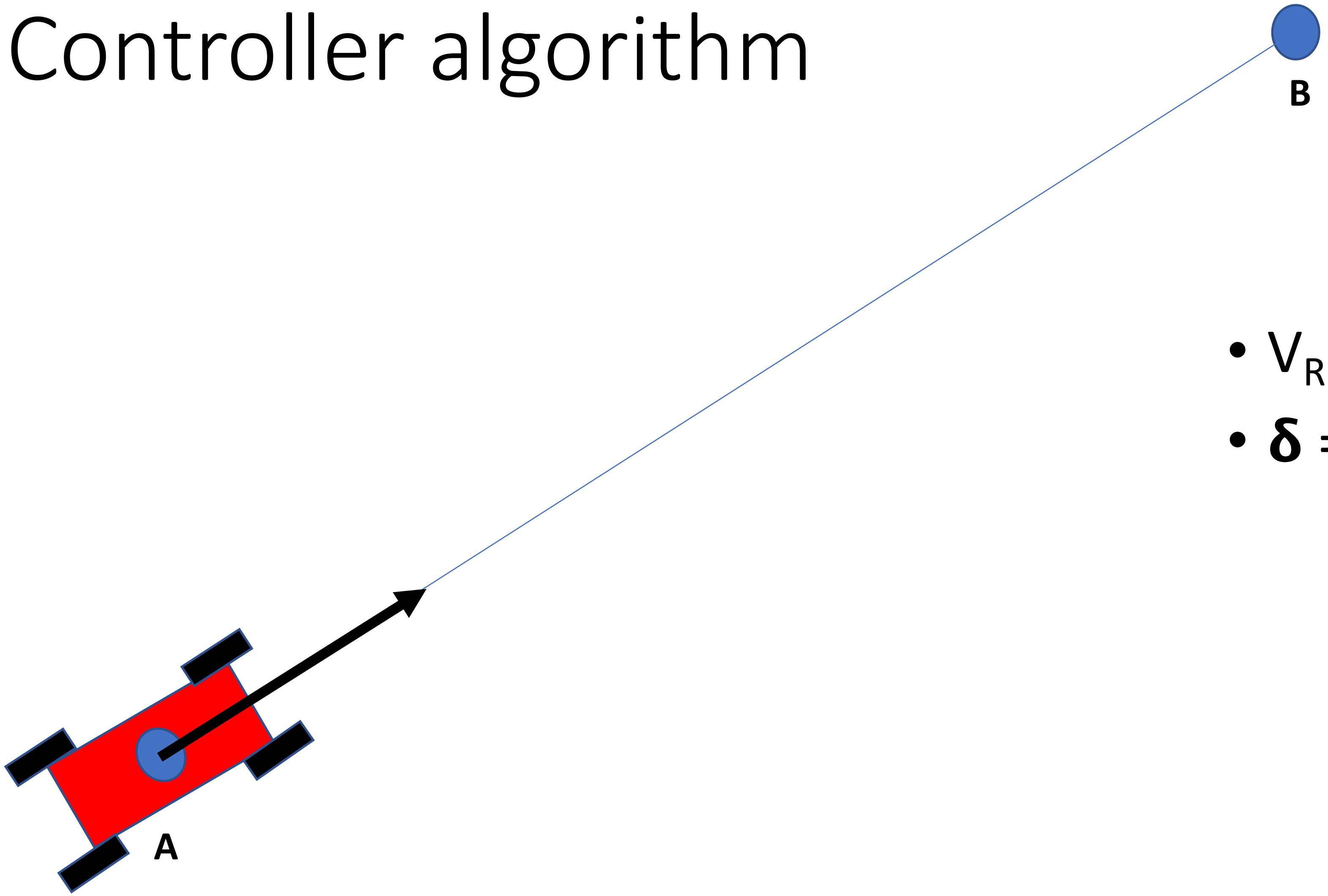
How to figure out this V_R and **δ**?

Controller algorithm



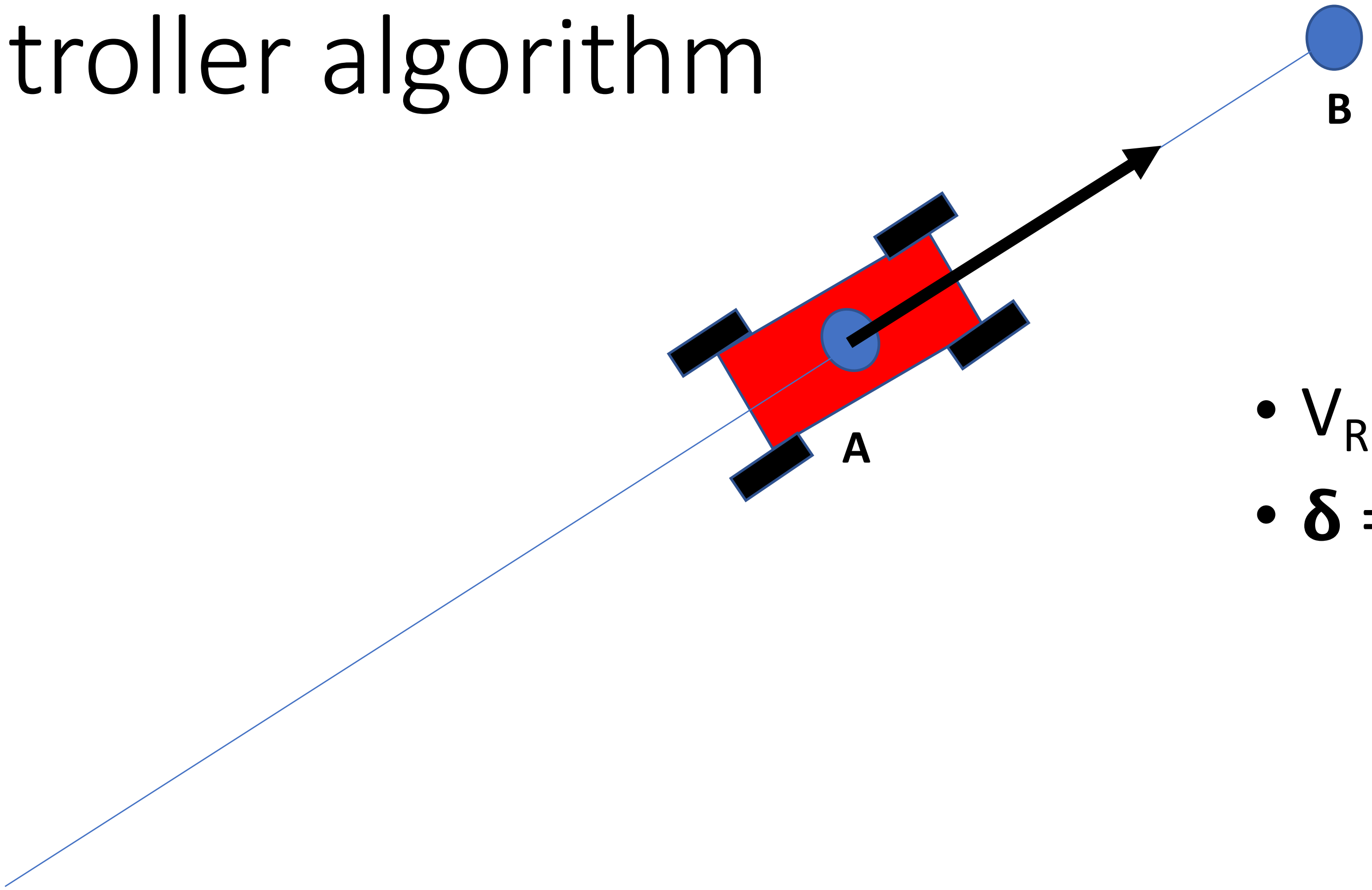
- $V_R = 0$
- $\boldsymbol{\delta} = K(\arctan(\overrightarrow{AB}) - \theta)$

Controller algorithm



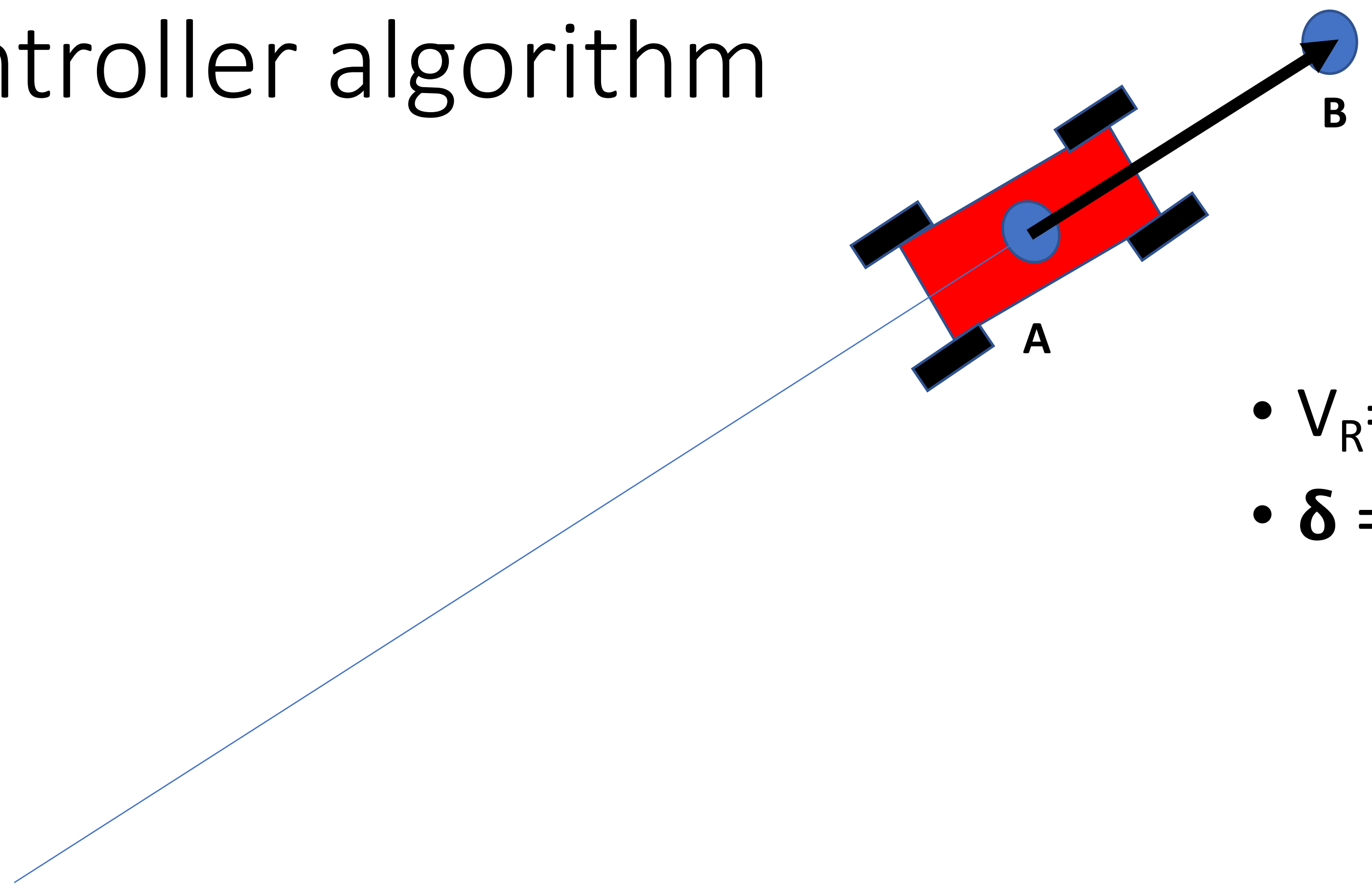
- V_R= Constant
- $\boldsymbol{\delta}$ = 0.0

Controller algorithm



- V_R= Constant
- $\boldsymbol{\delta}$ = 0.0

Controller algorithm



- V_R= Constant
- $\boldsymbol{\delta}$ = 0.0

This MP